SIEMENS

SIMATIC HMI

WinCC (TIA Portal)
WinCC Engineering V18 – 编程参考

系统手册

在线文档

11/2022
在线文档

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 Siemens AG 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
Digital Industries
Postfach 48 48
90026 NÜRNBERG
德国

文件订购号: 在线文档
Ⓟ 11/2022 本公司保留更改的权利

目录

VB 脚本编写

1

1.1 系统函数

1.1.1 AcknowledgeAlarm

描述

确认选择的所有报警。

该系统函数用于 HMI 设备没有 ACK 键时或报警视图的集成键不能使用的情况。

此系统函数只能用于功能键。

在函数列表中使用

确认报警

在用户自定义函数中使用

AcknowledgeAlarm

如果组态的设备支持用户自定义函数，则可以使用。 更多信息，请参考“设备相关性”。

参数

--

1.1.2 ActivatePLCCodeViewer

描述

将画面切换到包含 PLC 代码视图的特定画面。PLC 代码视图显示了相关程序段的程序代码。

组态 GRAPH 概览的“单击 PLC 代码视图按钮”(Click PLC code view button) 事件或某个按钮的事件所触发的 激活 PLC 代码视图 系统函数。

GRAPH 概览中的画面切换

组态 GRAPH 概览的“单击 PLC 代码视图按钮”(Click PLC code view button) 事件所触发的此系统函数，以在 PLC 代码视图中显示 GRAPH 步顺控程序。如果没有错误处于未决状态，则会显示 GRAPH 步顺控程序的顺序。

如果出现一个错误，则在跳过故障步后，会在 PLC 代码视图中显示故障步。如果出现多个错误，则会在跳过相关步后显示顺控程序的第一个故障步。如果更正了该错误，则会自动在视图中显示下一个故障步。在详细视图中，会根据错误类型显示转换或互锁。

通过报警进行画面切换

组态某个按钮的事件所触发的该系统函数。如果在运行系统中按下该按钮，则此系统函数会检查在所组态报警视图中最后选择的报警是监控报警还是 GRAPH 报警。如果该报警可进行跳转，则会通过相应的程序代码打开组态的 PLC 代码视图。

对于以下监控报警，可从报警视图中的报警跳转至 PLC 代码视图：

- 对于全局监控，仅限互锁监控（互锁）
- 对于局部监控，则针对输入参数的所有基本监控

有关监控的更多信息，请参见“编程 PLC > 通过 ProDiag 监控机器和工厂”(Programming PLC > Supervising machinery and plants with ProDiag) 部分。

对于所有 GRAPH 报警，均可跳转至 PLC 代码视图。跳转后，GRAPH 顺控程序和故障步会显示在 PLC 代码视图中。

如果同一步的监控报警和互锁报警同时处于未决状态，则无论已选择监控报警还是互锁报警，触发 PLC 代码视图中的系统函数后，始终首先显示互锁网络。

从监控报警跳转到 PLC 代码视图时，如果在函数块中使用支持的本地操作符，则实例名称必须符合以下命名约定：<FB-Name>_DB。

仅在使用全局操作符时，才能跳转到函数块或组织块。

说明

不支持与报警缓冲区或报警日志相关的跳转。

在函数列表中使用

激活 PLC 代码视图（画面名称，画面对象）

在用户自定义函数中使用

ActivatePLCCodeView Screen_name, Object_name

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

说明

要通过“激活 PLC 代码视图”系统函数跳转到 GRAPH 顺控程序，GRAPH 块的版本最低须为 V3.0。

说明

在 VB 脚本中使用系统函数“激活 PLC 代码视图”不会影响 GRAPH 概览。

参数

画面名称

包含 PLC 代码视图的画面的名称。

说明

不支持将画面切换到弹出画面或滑入画面。

画面对象

PLC 代码视图的对象名称。

1.1.3 ActivatePreviousScreen

描述

将画面切换到在当前画面之前激活的画面。 如果先前没有激活任何画面，则画面切换不执行。

最近调用的 10 个画面被保存。 当切换到不再保存的画面时，会输出系统报警。

说明

如果要使用系统函数，则必须在浏览结构中使用要切换到的画面。

在函数列表中使用

激活前一画面

在用户自定义函数中使用

ActivatePreviousScreen

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

--

1.1.4 ActivateScreen

描述

将画面切换到指定的画面。

使用“ActivateScreenByNumber”系统函数可以从根画面切换到永久性区域，反之亦然。

在函数列表中使用

激活画面（画面名称，对象编号）

在用户自定义函数中使用

ActivateScreen Screen_name, Object_number

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

画面名称

要切换到的画面的名称。

对象编号

画面切换后在指定画面中获得焦点的操作员控件元素。操作员控件元素的编号在组态期间使用 TAB 顺序确定。

在指定为"0"时：

- 如果调用该系统函数时焦点位于永久性区域，则永久性区域保留焦点。
- 如果调用该系统函数时焦点位于根画面，则指定画面中的第一个操作员控件元素获得焦点。

说明

如果将“到达边界”事件分配给"ActivateScreen"系统函数，则只有数值"0"对“对象编号”参数有效。活动对象不是由对象号定义的，而是由画面更改之前其 X 位置定义的。

示例

例如，单击按钮时，以下程序代码将使用 ActivateScreen 函数激活"Screen_2"。

```
Sub ActivateScreen_2()

'User-defined code
'' i. e. when pressing a button

ActivateScreen "Screen_2",0
```

1.1.5 ActivateScreenByNumber

描述

根据变量值将画面切换到另一画面。

画面由其画面号标识。

在函数列表中使用

根据编号激活画面（画面号，对象编号）

在用户自定义函数中使用

ActivateScreenByNumber Screen_number, Object_number

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

画面号

包含目标画面的画面编号的变量。

如果需要从根画面切换到永久性区域，则指定"0"或"-1"：

0 = 从根画面切换到永久性区域。

-1 = 从永久性区域切换到根画面

对象编号

画面切换后在指定画面中获得焦点的画面对象的编号。操作员控件元素的编号在组态期间使用 TAB 顺序确定。

在指定为"0"时：

- 如果调用该系统函数时焦点位于永久性区域，则永久性区域保留焦点。
- 如果调用该系统函数时焦点位于根画面，则指定画面中的第一个操作员控件元素获得焦点。

1.1.6 ActivateSystemDiagnosticsView

描述

激活系统诊断视图。系统诊断视图可显示相关设备的详细视图。

在函数列表中使用

激活系统诊断视图（画面名称，画面对象）

在用户自定义函数中使用

ActivateSystemDiagnosticsView Screen_name, Object_name

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

画面名称

系统诊断视图中包含的画面的名称。

画面对象

系统诊断视图的对象名称。

1.1.7 ArchiveLogFile

描述

此系统函数将日志移动或复制到其它存储位置，以便长期记录。

在使用“ArchiveLogFile”之前，务必先运行系统函数“CloseAllLogs”。

完成此系统函数后，再运行“OpenAllLogs”系统函数。

在“复制日志”模式下，只有在成功复制了日志或在复制过程中发生超时的情况下，才会重新打开日志。

在“移动日志”模式中，将重命名要移动的日志，并且会立即打开新日志。要移动重命名的日志，系统将会执行一个作业，在不能访问目标目录时该作业会尝试每 300 秒移动一次日志。在运行系统重新启动后，该作业仍将保留直到执行完毕。因此，在移动日志之前，应检查目标目录是否可访问。

进行审计跟踪时，请始终使用“移动”(hmiMove) 模式，否则将在复制存储的数据方面违反 FDA 准则。

在函数列表中使用

归档日志文件（日志类型，日志，目录名称，模式）

在用户自定义函数中使用

ArchiveLogFile Log_type, Log, Directory_name, Mode

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

日志类型

确定日志的类型：

0 (hmiTagArchive) = 变量日志

1 (hmiAlarmArchive) = 报警日志

2 (hmiAudittrailArchive) = 可用于符合 GMP 的项目的审计跟踪日志。更多信息，请参见“激活符合 GMP 的组态”。

日志

被归档的日志的名称。

目录名称

保存日志的路径。

模式

0 (hmiCopy) = 复制日志

1 (hmiMove) = 移动日志

应用实例

希望将日志文件从本地存储介质移动到服务器，以定期生成此文件的备份副本。

有关组态的注意事项

在调度程序中设置一个每天定时执行的任务。为此任务组态以下函数列表：

HMI 设备上的操作步骤

- 将关闭所有日志文件。
- 将指定的日志文件移动到服务器上。
- 将再次打开所有关闭的日志文件。

1.1.8 备份 RAM 文件系统

描述

将 RAM 文件系统备份到 HMI 设备的存储介质中。

重启动 HMI 设备后，数据被自动重新装载到 RAM 文件系统中。

诸如 Internet Explorer 的应用程序会将将最近访问的网址数据临时保存到 HMI 设备的 DRAM 文件系统中。

在函数列表中使用

备份 RAM 文件系统

在用户自定义函数中使用

BackupRAMFileSystem

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

--

1.1.9 CalibrateTouchScreen

描述

调用用于校准触摸屏幕的程序。

在校准过程中，将会提示触摸屏幕显示上的五个位置。在 30 秒内触摸屏幕显示以确认校准过程。如果在该时间间隔内没有完成校准，校准设置被放弃。用户提示为英语。

首次启动 HMI 设备时使用该系统函数。

说明

系统函数将重置方向键。

在函数列表中使用

校准触摸屏

在用户自定义函数中使用

CalibrateTouchScreen

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

--

说明

系统函数“校准触摸屏”无法仿真。

1.1.10 ChangeConnection

说明

断开与当前所用 PLC 的连接，并其它地址的 PLC 建立连接。新连接的 PLC 必须属于同一种设备类别（S7-300、S7-400 等）。

说明

更改为其它地址时，需确保该地址未被其它 HMI 设备使用。

支持以下地址类型：

- IP 地址
- MPI 地址

支持以下 PLC 类型：

- SIMATIC S7 300/400
- SIMATIC S7 200
- SIMATIC S7 1200（设备版本最高为 V11）
- SIMATIC S7 LOGO!

- SIMATIC S7-NC
- SIMOTION

在函数列表中使用

更改连接（连接，地址，插槽，机架）

在用户自定义函数中使用

ChangeConnection Connection, Address, Slot, Rack

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

连接

断开的连接的名称。该名称在组态时设置，例如，在“连接”编辑器中设置。

地址

要建立连接的 PLC 的 MPI/PROFIBUS 或 IP 地址。

说明

通过变量设置地址。对象列表显示所有数据类型的变量。仅选择下列数据类型的变量：

- 以太网连接：“String”数据类型
- MPI 连接：“Int”数据类型

插槽

要建立连接的 PLC 的插槽。

机架

要建立连接的 PLC 的机架。

应用示例

要在多台机器上操作一台 HMI 设备，则需在项目中组态一个控制器。更换 PLC 时，需要先断开与在用 PLC 的连接。然后重新建立与具有其它地址参数的新的 PLC 的连接。此时，从新控制器将更新该连接中的所有变量。

默认情况下，使用创建项目时指示的 PLC。

1. 在“连接”编辑器中输入 PLC 的名称和地址。
2. 在过程画面中组态一个按钮。
3. 对“按下”事件组态系统函数“更改连接”。
4. 输入连接名称和 PLC 地址作为参数。

1.1.11 ChangeConnectionEIP

描述

断开与当前使用的 Allan Bradley 控制器的连接，并与运行系统中的另一个 Allan Bradley 控制器建立连接。“EIP”代表“Ethernet/IP”。

新连接的控制器必须属于同一设备类别 (Allen Bradley)。

对于要建立的连接，要连接的控制器的 CPU 类型必须与已连接的控制器 CPU 类型匹配，例如，两个控制器的 CPU 类型均为“SLC/Micrologix PLC”。

说明

更改为另一个地址时，确保该地址尚未被其它 HMI 设备使用。

支持以下地址类型：

- IP 地址

在函数列表中使用

更改连接 EIP（连接，地址，通信路径）

在用户自定义函数中使用

ChangeConnectionEIP Connection, Address, Communication_path

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

连接

断开的连接的名称。该名称在组态时设置，例如，在“连接”编辑器中设置。

地址

正在连接的控制器的 IP 地址。

说明

通过变量设置地址。对象列表显示所有数据类型的变量。仅选择数据类型为“字符串”的变量。

说明

存储的 IP 地址必须由 4 个十进制数字组成，取值范围介于 0 到 255 之间。

通信路径

从以太网模块到正在连接的控制器的 CIP 路径。

可使用字符串变量或“字符串”数据类型的 HMI 变量来指定通信路径。

应用示例

想要在多台机器上操作一台 HMI 设备。为此，请在项目中组态一个 PLC。当切换 PLC 时，断开与正在使用的 PLC 的连接。然后重新建立与具有其它地址参数的新的 PLC 的连接。此时，该连接的所有变量均从新的 PLC 更新。

1. 在“连接”编辑器中输入 PLC 的名称和地址。
2. 在过程画面中组态一个按钮。
3. 对“按下”事件组态系统函数“ChangeConnectionEIP”。
4. 输入连接名称和 PLC 地址作为参数。

1.1.12 ClearAlarmBuffer

描述

删除 HMI 设备报警缓冲区中的报警。

说明

尚未确认的报警也被删除。

在函数列表中使用

清除报警缓冲区（报警类别编号）

在用户自定义函数中使用

ClearAlarmBuffer Alarm_class_number

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

报警类别编号

确定要从报警缓冲区中删除的报警：

0 (hmiAll) = 所有报警

1 (hmiAlarms) = 报警类别“错误”的报警

2 (hmiEvents) = 报警类别“警告”的报警

3 (hmiSystem) = 报警类别“系统事件”的报警

4 (hmiS7Diagnosis) = 报警类别"S7 诊断报警”的报警

说明

设备相关性

报警类别"Diagnosis Events"的报警在基本面板上不可用。

1.1.13 ClearAlarmBufferProtool

描述

该系统函数用来确保兼容性。

它具有与系统函数“清除报警缓冲区”相同的功能，但使用旧的 ProTool 编号方式。

在函数列表中使用

清除报警缓存 Protool（报警类别编号）

在用户自定义函数中使用

ClearAlarmBufferProtoolLegacy Alarm_class_number

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

报警类别编号

将要删除其消息的报警类别号：

-1 (hmiAllProtoolLegacy) = 所有报警

0 (hmiAlarmsProtoolLegacy) = 报警类别的报警“Errors”

1 (hmiEventsProtoolLegacy) = 报警类别的报警“Warnings”

2 (hmiSystemProtoolLegacy) = 报警类别的报警“System”

3 (hmiS7DiagnosisProtoolLegacy) = 报警类别的报警“Diagnosis Events”

说明

设备相关性

报警类别“Diagnosis Events”的报警在基本面板上不可用。

1.1.14 ClearDataRecord

描述

删除配方数据记录。

可以从一个或多个配方中删除若干个数据记录。

在函数列表中使用

清除数据记录（配方编号/名称，数据记录编号/名称，确认，输出状态消息，处理状态）

在用户自定义函数中使用

ClearDataRecord

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

配方编号/名称

要删除配方数据记录的配方的编号或名称。如果想要从所有可用配方中删除配方数据记录，则指定“0”。

数据记录编号/名称

要删除的配方数据记录的编号或名称。如果想要删除所有的配方数据记录，则指定"0"。

确认

确定是否需要操作员确认删除操作：

0 (hmiOff) = 关：无需确认便开始删除。

1 (hmiOn) = 开：必须经过确认后才能开始删除。

输出状态消息

确定在删除后是否输出状态消息：

0 (hmiOff) = 关：不输出状态消息。

1 (hmiOn) = 开：输出状态消息。

处理状态

返回系统函数的处理状态。例如，可以使用返回值延迟执行其它系统函数，直到本系统函数已经成功完成：

2 = 系统函数正在执行。

4 = 系统函数已经成功完成。

12 = 因为出现了错误，系统函数未执行。

只能为参数使用 HMI 变量。

1.1.15 ClearDataRecordMemory

描述

删除指定的存储介质中的所有配方数据记录。

在函数列表中使用

ClearDataRecordMemory（存储位置，确认后，输出状态消息，处理状态）

在用户自定义函数中使用

ClearDataRecordMemory Storage_location, Confirmation, Output_status_message, Processing_status

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

存储位置

确定存储位置：

0 (hmiFlashMemory) = 闪存：HMI 设备的内部闪存

1 (hmiStorageCard) = 存储卡

2 (hmiStorageCard2) = 存储卡 2

3 (hmiStorageCard3) = 多媒体存储卡

4 (hmiStorageCard4) = 存储卡

需要确认

确定是否需要操作员确认删除操作：

0 (hmiOff) = 关：无需确认便开始删除。

1 (hmiOn) = 开：必须经过确认后才能开始删除。

输出状态消息

确定在删除后是否输出状态消息：

0 (hmiOff) = 关：不输出状态消息。

1 (hmiOn) = 开：输出状态消息。

处理状态

返回系统函数的处理状态。例如，可以使用返回值延迟执行其它系统函数，直到本系统函数已经成功完成：

2 = 系统函数正在执行。

4 = 系统函数已经成功完成。

12 = 因为出现了错误，系统函数未执行。

只能为参数使用 HMI 变量。

1.1.16 ClearLog

描述

删除给定日志中的所有数据记录。

在函数列表中使用

清除日志（日志类型，日志）

在用户自定义函数中使用

ClearLog Log_type, Log

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

日志类型

确定日志的类型：

0 (hmiTagArchive) = 变量日志

1 (hmiAlarmArchive) = 报警日志

2 (hmiAudittrailArchive) = 审计跟踪日志。可用于符合 GMP 的项目的审计跟踪日志。更多信息，请参见“激活符合 GMP 的组态”。

日志

要删除所有条目的日志的名称。

1.1.17 CloseAllLogs

描述

断开 WinCC 与所有日志之间的连接。

在函数列表中使用

关闭所有日志

在用户自定义函数中使用

CloseAllLogs

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

--

应用实例

处于运行状态时，希望更改记录过程值的数据介质。

有关组态的注意事项

在"Close Archive"按钮上组态系统函数“关闭所有日志”。

在"Open Archive"按钮上组态系统函数“打开所有日志”和“开始记录”。

将要停止或开始的日志的相应名称作为参数进行传送。

HMI 设备上的操作步骤

按下"Close Archive"按钮后，打开的日志将关闭。可以改变数据介质。改变数据介质期间会继续执行记录操作。待记录的过程值会被缓存。用"Open Archive"按钮打开所有日志。继续在指定日志中进行记录。缓冲的过程值之后被添加到日志中。

1.1.18 ControlSmartServer

描述

启动或停止 Sm@rtServer。

在函数列表中使用

控制 SmartServer（模式）

在用户自定义函数中使用

ControlSmartServer Mode

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

模式

指定是启动还是停止 Sm@rtServer。

-1 (hmiToggle) = 切换：在两种模式之间切换

0 (hmiStop) = 停止：停止 Sm@rtServer

1 (hmiStart) = 启动：启动 Sm@rtServer

1.1.19 ControlWebServer

描述

启动或停止 Web 服务器。

在函数列表中使用

控制 Web 服务器（模式）

在用户自定义函数中使用

ControlWebServer Mode

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

模式

指定是启动还是停止 Web 服务器。

-1 (hmiToggle) = 切换：在两种模式之间切换

0 (hmiStop) = 停止：停止 Web 服务器。

1 (hmiStart) = 启动：启动 Web 服务器。

1.1.20 CopyLog

描述

将日志的内容复制到其它日志中。变量值只能复制到其它数据日志中，而报警只能复制到其它报警日志中。

说明

如果使用“复制日志”系统函数复制日志，则外部应用程序有可能无法读取日志副本的消息文本中某些特定国家/地区的特殊字符。这不适用于 WinCC Runtime。WinCC Runtime 可读取复制的日志文件，而不会出错。

说明

复制循环日志时，会复制 80% 的日志条目。其余 20% 的条目不会复制，因为该空间默认为缓冲区溢出预留。

在函数列表中使用

复制日志（日志类型，目标日志，源日志，模式，删除源日志）

在用户自定义函数中使用

CopyLog Log_type, Destination log, Source_log, Mode, Delete_source_log

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

日志类型

确定日志的类型：

0 (hmiTagArchive) = 变量日志

1 (hmiAlarmArchive) = 报警日志

目标日志

要将条目复制到其中的日志的名称（目标日志）。

源日志

要复制其条目的日志的名称（源日志）。

模式

确定采用什么方式将复制下来的条目放入目标日志中：

0 (hmiOverwrite) = 覆盖：覆盖已有的条目。

2 (hmiAppend) = 添加：在目标日志的末尾插入这些条目。当达到设置的日志大小时，对目标日志的处理方式类似于循环日志。

删除源日志

确定复制后是否删除源日志。

0 (hmiNo) = 否：不删除。

1 (hmiYes) = 是：删除。

1.1.21 DecreaseTag

描述

从变量值中减去指定的值。

X = X - a

说明

系统函数使用同一变量作为输入和输出值。当该系统函数用于转换数值时，必须使用辅助变量。使用系统函数“设置变量”将辅助变量分配给变量值。

如果在报警事件中组态了该系统函数但变量未在当前画面中使用，则无法确保在 PLC 中使用实际的变量值。通过设置“连续循环”采集模式可以改善这种情况。

在函数列表中使用

减少变量（变量，值）

在用户自定义函数中使用

DecreaseTag Tag, Value

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

要减去指定值的变量。

值

其值作为减数。

1.1.22 EditAlarm

说明

为所选报警或为最后选择的报警（进行了多重选择时）触发“编辑”事件。如果要编辑的报警尚未确认，则在调用该系统函数时自动确认这些报警。

此系统函数只能用于功能键。

在函数列表中使用

编辑报警

在用户自定义函数中使用

EditAlarm

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

--

1.1.23 Encode

描述

修改传送给自动化系统 (AS) 的变量的“String”数据类型。WinCC 的变量数据类型“String”转换为 AS 的数据类型“Array of byte”。其结果会写入变量。

在函数列表中使用

编码（字节数组，字符串，编码）

在用户自定义函数中使用

Encode Byte_array, String, Encoding

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

字节数组

包含转换值的变量。

说明

Byte array 必须是字符串长度 + 2 的两倍。当字符串长度为 120 个字符时，Byte array 必须包含 242 个数组元素。
如果大小不够，字符将被截断或者无法转换。

字符串

被转换的数据类型“String”的变量。

编码

0 (hmiEncodeUTF16LE) - 字符串采用 UTF16LE 编码 (Unicode 16 Little Endian)。

1.1.24 EncodeEx

描述

修改传送给自动化系统 (AS) 的变量的“String”数据类型。WinCC 的变量数据类型“String”转换为 AS 的数据类型“Array of byte”。其结果会写入变量。

与“编码”系统函数相比，该函数允许定义“换行”参数。使用“换行”参数可以删除分行符或使用预定义的字符替换分行符。

在函数列表中使用

EncodeEx（字节数组，字符串，编码，换行）

在用户自定义函数中使用

EncodeEx Byte_array, String, Encoding, Line_break

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

字节数组

包含转换值的变量。

说明

字节数组长度必须是字符串长度的两倍 +2。如果字符串长度为 120 个字符，则字节数组必须包含 242 个数组元素。
如果大小不够，字符将被截断或者无法转换。

字符串

被转换的数据类型“String”的变量。

编码

0 (hmiEncodeUTF16LE) - 字符串采用 UTF16LE 编码 (Unicode 16 Little Endian)。

换行

所有分行符都将被删除或被替换为预定义的字符。设置为默认值时，请勿替换分行符。

0（使用“\r\n”(0x000D, 0x000A) 替换）- 分行符被替换为“\r\n”。

1（使用“\n”(0x000A) 替换）- 分行符被替换为“\n”。

2（不替换）- 不替换分行符。

3（删除分行符）- 已删除分行符。

1.1.25 ExportDataRecords

描述

将一个配方的一条或全部数据记录导出到一个 CSV 文件或 TXT 文件中。

为每个配方创建一个文件。

在函数列表中使用

导出数据记录（配方编号/名称，数据记录编号/名称，文件名称，覆盖，输出状态消息，处理状态）

在用户自定义函数中使用

ExportDataRecords Recipe_number/name, Data_record number/name, File_name, Overwrite, Output_status_message, Processing_status

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

配方编号/名称

要导出其数据记录的配方的编号或名称。如果要导出所有可用配方中的配方数据记录，则指定“0”。

数据记录编号/名称

要导出的配方数据记录的编号或名称。如果要导出所有配方数据记录，则指定“0”。

文件名称

要向其导出配方数据记录的文件的名称。输入名称，包括存储位置和文件扩展名（*.csv 或 *.txt），例如“C:\TEMP\Orange.csv”。如果文件夹不存在，则会在导出期间创建文件夹。

如果没有完整输入文件名称，会根据已组态的配方响应：

- 如果组态了多个配方并且
  仅指定一个文件名而没有指定存储路径，该文件将保存在系统目录中，例如“C:\Documents and Settings\[User]”。
  如果仅指定一个路径，未指定文件名，则文件名将根据相应的配方名自动创建。例如，这要求已在指定文件夹中创建文件夹“D:\Data\”。如果未创建文件夹 "D:\Data"，文件夹名将用作文件名前缀，例如 Data_配方名.csv。
- 如果仅组态了一个配方并且仅指定了一个路径且没有指定文件名，
  如果文件夹不存在，则创建带有文件夹名称的文件。然而，这没有文件扩展。
  如果存在文件夹导出将带有错误消息而中止。

如果将存储卡用作存储位置，则按如下方式指定存储位置：“\StorageCard\<名称>”。

对于基本面板，输入以下形式的文件名：“\USB_X60.1\<名称>”

覆盖

确定是否覆盖具有相同名称的现有导出文件：

0 (hmiOverwriteForbidden) = 否：不覆盖导出文件。将不执行导出过程。

1 (hmiOverwriteAlways) = 是：覆盖导出文件，且不会出现确认提示。

2 (hmiOverwriteWithPrompting) = 需要确认：只有在确认后，才会覆盖导出文件。

输出状态消息

确定在导出后是否输出状态消息：

0 (hmiOff) = 关：不输出状态消息。

1 (hmiOn) = 开：输出状态消息。

处理状态

返回系统函数的处理状态。例如，可以使用返回值延迟执行其它系统函数，直到本系统函数已经成功完成：

2 = 系统函数正在执行。

4 = 系统函数已经成功完成。

12 = 因为出现了错误，系统函数未执行。

只能为参数使用 HMI 变量。

配方数据记录的导出格式

如果选择".csv"作为导出文件的文件扩展名，则仅支持 ANSI 字符集中的有效字符。这也适用于在十进制数字和列表元素中的分隔符。使用的分隔符在导出计算机的操作系统的国家/地区设置中定义。

也可以为导出文件设置"Unicode text"（".txt"）格式。此文件格式支持 WinCC 和 WinCC Runtime 字符集。而且，使用的分隔符在导出计算机的操作系统的国家/地区设置中指定。此文件格式始终使用列表元素中的制表符。

相应的文件导入功能也支持".csv"和".txt"(Unicode) 文件格式。

应用示例

通过一个按键，导出所有数据记录。

有关组态的注意事项

将“导出数据记录”系统函数组态到所期望的键的“按下”事件中。传送下列参数：

- 配方编号/名称 = 1
- 数据记录编号/名称 = 0

- 文件名 = c:\temp\orange.csv（对于基本面板，则为"\USB_X60.1\orange.csv"）
- 覆盖 = 1
- 输出状态消息 = 1

也可以指定变量代替这些常数。根据组态，操作员可以在 I/O 域中输入所需值，也可以从 PLC 读取。这样，操作员就可以确定要导出哪些配方数据记录。

HMI 设备上的操作步骤

一旦激活了该键，系统函数便被触发。对常数或变量进行判断。配方 1 中的所有数据记录导出到 orange.csv 文件。如果该文件已经存在，将覆盖该文件。

在导出数据记录之后，输出一条系统事件。

1.1.26 ExportDataRecordsWithChecksum

描述

将配方的一条或所有数据记录导出为 CSV 文件，并对文件中的每行都生成校验和。

为每个配方创建一个文件。

在函数列表中使用

导出带有校验和的数据记录（配方编号/名称，数据记录编号/名称，文件名称，覆盖，输出状态消息，处理状态）

在用户自定义函数中使用

ExportDataRecordsWithChecksum Recipe_number_or_name, Data_record number_or_name, File_name, Overwrite, Output_status_message, Processing_status

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

配方编号/名称

要导出其数据记录的配方的编号或名称。如果要导出所有可用配方中的配方数据记录，则指定“0”。

数据记录编号/名称

要导出的配方数据记录的编号或名称。如果要导出所有配方数据记录，则指定“0”。

文件名称

配方数据记录导出到的 CSV 文件的名称。输入路径和文件扩展名，例如“C:\TEMP\Orange.CSV”。

如果要保存到存储卡中，则按如下方式指定存储位置：“\StorageCard\<文件名>”。

如果只定义了文件名而没有指定路径，则文件保存在运行系统启动的目录。如果在 Windows 7 操作系统中没有启用对该目录的写权限，则文件将保存在用户目录的“VirtualStore”文件夹中。

如果仅指定一个导出路径，则文件名将根据对应的配方名自动创建。例如，这要求已经创建“D:\Temp\”目录。如果目录“ "D:\Temp"”不存在，则目录名将用作文件名的前缀，即 Temp_配方名称.csv。

覆盖

确定是否覆盖具有相同名称的已存在的 CSV 文件：

0 (hmiOverwriteForbidden) = 否：不覆盖 CSV 文件。将不执行导出过程。

1 (hmiOverwriteAlways) = 是：不进行确认提示即覆盖 CSV 文件。

2 (hmiOverwriteWithConfirmation) = 需要确认：CSV 文件在确认后才被覆盖。

输出状态消息

确定在导出后是否输出状态消息：

0 (hmiOff) = 关：不输出状态消息。

1 (hmiOn) = 开：输出状态消息。

处理状态

返回系统函数的处理状态。例如，可以使用返回值延迟执行其它系统函数，直到本系统函数已经成功完成：

2 = 系统函数正在执行。

4 = 系统函数已经成功完成。

12 = 因为出现了错误，系统函数未执行。

只能为参数使用 HMI 变量。

配方数据记录的导出格式

如果选择".csv"作为导出文件的文件扩展名，则仅支持 ANSI 字符集中的有效字符。这也适用于在十进制数字和列表元素中的分隔符。使用的分隔符在导出计算机的操作系统的国家/地区设置中定义。

也可以为导出文件设置"Unicode text"（".txt"）格式。此文件格式支持 WinCC 和 WinCC Runtime 字符集。而且，使用的分隔符在导出计算机的操作系统的国家/地区设置中指定。此文件格式始终使用列表元素中的制表符。

相应的文件导入功能也支持".csv"和".txt"(Unicode) 文件格式。

应用实例

用一个键来导出所有数据记录并指定校验和。

有关组态的注意事项

将"ExportDataRecordsWithChecksum"系统函数组态到所期望的键的"Press"事件中。传送下列参数：

- 配方编号/名称 = 1
- 数据记录编号/名称 = 0
- 文件名称 = c:\temp\orange.csv
- 覆盖 = 1
- 输出状态消息 = 1

也可以指定变量代替这些常数。根据组态，操作员可以在 I/O 域中输入所需值，也可以从 PLC 读取。这样，操作员就可以确定要导出哪些配方数据记录。

HMI 设备上的操作步骤

一旦激活了该键，系统函数便被触发。对常数或变量进行判断。配方 1 的所有数据记录导出到 orange.csv 文件并指定校验和。如果该文件已经存在，将覆盖该文件。

在导出数据记录之后，输出一条系统事件。

1.1.27 ExportImportUserAdministration

说明

将当前激活项目的用户管理的所有用户导出到指定文件，或将用户从指定文件导入到当前激活的项目中。

用户、用户口令和权限都保存在用户管理中。

用户管理组态的导出/导入包含所有设置。现有对象（用户、组、登录设置、授权级别）在导入期间会被覆盖。

导入的用户立即生效。

在函数列表中使用

导出导入用户管理（文件名称，方向）

在用户自定义函数中使用

ExportImportUserAdministration File_name, Direction

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

文件名称

包含密码的文件名称或者要写入密码的文件名称。输入文件位置和文件扩展名 (*.txt)，例如“C:\TEMP\Passwords.txt”。

说明

如果要保存到存储卡中，按如下方式指定存储位置：“\StorageCard\<文件名>”。

传输方向

指定要导出还是导入密码：

0 (hmiExport) = 导出：导出密码。

1 (hmiImport) = 导入：导入密码。

1.1.28 GetBrightness

描述

读取亮度值。

在函数列表中使用

获取亮度值(Brightness)

在用户自定义函数中使用

GetBrightness Brightness

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

亮度

要写入值的变量。

1.1.29 GetDataRecordFromPLC

描述

将所选配方数据记录从 PLC 传送到 HMI 设备的存储介质中。

在函数列表中使用

从 PLC 获取数据记录（配方编号/名称，数据记录编号/名称，覆盖，输出状态消息，处理状态）

在用户自定义函数中使用

GetDataRecordFromPLC Recipe_number_or_name, Data_record_number_or_name, Overwrite, Output_status_message, Processing_status

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

配方编号/名称

要传送其配方数据记录的配方的编号或名称。

数据记录编号/名称

从 PLC 传送到 HMI 设备数据介质的配方数据记录的编号或名称。

覆盖

确定是否覆盖具有相同名称的现有配方数据记录：

0 (hmiOverwriteForbidden) = 否：不覆盖配方数据记录。传送过程将不会执行。

1 (hmiOverwriteAlways) = 是：无提示直接覆盖配方数据记录。

2 (hmiOverwriteWithPrompting) = 需要确认：确认后才覆盖配方数据记录。

输出状态消息

确定传送之后是否输出状态消息：

0 (hmiOff) = 关：不输出状态消息。

1 (hmiOn) = 开：输出状态消息。

处理状态

返回系统函数的处理状态。例如，可以使用返回值延迟执行其它系统函数，直到本系统函数已经成功完成：

2 = 系统函数正在执行。

4 = 系统函数已经成功完成。

12 = 因为出现了错误，系统函数未执行。

只能为参数使用 HMI 变量。

应用实例

要使用按键将数据记录从 PLC 传送到 HMI 设备的数据介质中。

有关组态的注意事项

将“从 PLC 获取数据记录”系统函数组态到所期望的键的“按下”事件中。传送下列参数：

配方编号/名称 = 1

数据记录编号/名称 = 1

覆盖 = 1

输出状态消息 = 1

也可以指定变量代替这些常数。根据组态，操作员可以在 I/O 域中输入所需值，也可以从 PLC 读取。这样，操作员可以确定要从 PLC 传送哪些配方数据记录。

HMI 设备上的操作步骤

一旦激活了该键，系统函数便被触发。评估常数或变量，将“配方 1”中的第一个数据记录从 PLC 传送到 HMI 设备的数据介质中。如果该配方数据记录已经存在，它将被覆盖。

传送后输出一则系统事件。

1.1.30 GetDataRecordName

描述

在指定的变量中写入指定配方和配方数据记录的名称。

说明

如果配方或配方数据记录不存在，则将通配符 (“###”) 写入该变量。

说明

作为变量，仅支持内部变量或外部变量。

在函数列表中使用

获取数据记录名称（配方编号，数据记录号，配方名称，数据记录名，处理状态）

在用户自定义函数中使用

GetDataRecordName Recipe_number, Data_record_number, Recipe_name, Data_record_name, Processing_status

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

配方号

名称将写入指定变量的配方的编号。

数据记录号

名称将写入指定变量的配方数据记录的编号。

配方名称

要写入配方名称的变量。该变量必须为 STRING 类型。

数据记录名

要写入配方数据记录名的变量。该变量必须为 STRING 类型。

处理状态

返回系统函数的处理状态。例如，可以使用返回值延迟执行其它系统函数，直到本系统函数已经成功完成：

2 = 系统函数正在执行。

4 = 系统函数已经成功完成。

12 = 因为出现了错误，系统函数未执行。

只能为参数使用 HMI 变量。

应用实例

要在 HMI 设备上输出所显示的配方以及配方数据记录的名称。

组态下列变量：

- INTEGER 类型的“RecNumber”
- INTEGER 类型的“RecDataNumber”
- STRING 类型的“RecName”
- STRING 类型的“RecDataName”

组态一个包含变量“RecNumber”(配方号)和“RecDataNumber”(数据记录号)的配方视图。

将系统函数“获取数据记录名称”组态到按钮的“按下”事件中，并传送下列参数：

- 配方编号：RecNumber
- 数据记录号：RecDataNumber
- 配方名称：RecName
- 数据记录名：RecDataName

组态两个输出域并将它们连接到变量“RecName”和“RecDataName”。

在配方视图中选择配方和相关的数据记录号。一旦激活了按钮，该系统函数便触发，配方和配方数据记录的名称显示在两个输出域中。

1.1.31 GetDataRecordTagsFromPLC

描述

将加载到 PLC 中的配方数据记录的值传送给相应的配方变量。

例如，在设备上进行交互操作期间，使用该系统函数。

在函数列表中使用

从 PLC 获取数据记录变量（配方编号/名称，处理状态）

在用户自定义函数中使用

GetDataRecordTagsFromPLC Recipe_number_or_name, Processing_status

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

配方编号/名称

要将其值从 PLC 写入变量的配方数据记录的编号或名称。

处理状态

返回系统函数的处理状态。例如，可以使用返回值延迟执行其它系统函数，直到本系统函数已经成功完成：

2 = 系统函数正在执行。

4 = 系统函数已经成功完成。

12 = 因为出现了错误，系统函数未执行。

只能为参数使用 HMI 变量。

1.1.32 GetGroupNumber

描述

读取 HMI 设备当前登录用户所属的组编号，并将其写入给定变量。

在函数列表中使用

获取组编号（变量）

在用户自定义函数中使用

GetGroupNumber Tag

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

要写入组编号的变量。

1.1.33 GetPassword

描述

在给定的变量中写入当前登录到 HMI 设备的用户的密码。

说明

确保给定变量的值未显示在项目中的其它位置。

说明

无法读取 SIMATIC Logon 用户的密码。

说明

“ReadPassword”系统功能不适用于设备版本 14.0.0.0 或更高版本。

如果该系统功能在早于 V14 版本的项目中使用，迁移到 V14 版本后将弃用该功能。因此，与该系统功能链接的事件无效。“ReadPassword”系统功能在组态时会显示为故障。编译时会生成警告。“ReadPassword”系统功能不再包含在编译的运行系统项目中。

在函数列表中使用

ReadPassword（变量）

在用户自定义函数中使用

GetPassword Tag

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

要写入密码的变量。

1.1.34 GetUserName

描述

在给定变量中写入当前登录到 HMI 设备的用户的用户名。

如果给出的变量具有控制连接，则用户名在 PLC 连接上也可用。该系统函数将使诸如执行某些功能与用户有关的版本成为可能。

在函数列表中使用

获取用户名（变量）

在用户自定义函数中使用

GetUserName Tag

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

要写入用户名的变量。

1.1.35 GoToEnd

描述

在 HMI 设备上执行 <End> 键功能：

当 HMI 设备在默认情况下不具有该功能时使用此系统函数。

此系统函数只可用于以下功能键：

在函数列表中使用

转到末尾

在用户自定义函数中使用

GoToEnd

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

--

1.1.36 GoToHome

描述

在 HMI 设备上执行 <Home> 键功能：

当 HMI 设备在默认情况下不具有该功能时使用此系统函数。

此系统函数只可用于以下功能键：

在函数列表中使用

转到首页

在用户自定义函数中使用

GoToHome

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

--

1.1.37 ImportDataRecords

说明

从一个 CSV 文件或 TXT 文件中导入配方的一条或全部数据记录。

指定路径时，将导入指定路径的所有记录。

说明

不检查导入文件是否一致

导入函数不检查 CSV 文件中的值在导出和导入之间是否仍然有效，例如，不检查这些值是否仍在规定的限值内。

在函数列表中使用

导入数据记录（文件名称，数据记录编号/名称，覆盖，输出状态消息，处理状态）

在用户自定义函数中使用

ImportDataRecords File_name, Data_record_number_or_name, Overwrite, Output_status_message, Processing_status

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

文件名称

要从中导入配方数据记录的文件的名称。还要指定存储位置和文件扩展名（*.csv 或 *.txt），例如“C:\TEMP\Orange.csv”。

对于基本面板，按如下方式指定文件名：“\USB_X60.1\<Name>”

对于其它设备：如果将存储卡用作存储位置，则按如下方式指定存储位置：“\StorageCard\<Name>”。

要导入所有配方数据记录，仅指定存储位置的路径，不指定文件名：“C:\TEMP\”。系统函数将从该存储位置导入所有 CSV 文件。

数据记录编号/名称

要导入的配方数据记录的编号或名称。如果要导入所有的配方数据记录，则指定“0”。

覆盖

确定是否覆盖现有的配方数据记录：

0 (hmiOverwriteForbidden) = 否：不覆盖配方数据记录。将不执行导入过程。

1 (hmiOverwriteAlways) = 是：不进行确认提示即覆盖配方数据记录。

2 (hmiOverwriteWithPrompting) = 需要确认：配方数据记录在确认后才被覆盖。

输出状态消息

确定在导入后是否输出状态消息：

0 (hmiOff) = 关：不输出状态消息。

1 (hmiOn) = 开：输出状态消息。

处理状态

返回系统函数的处理状态。例如，使用返回值，以确保在执行其它系统函数前此系统函数已成功完成。

2 = 系统函数正在执行。

4 = 系统函数已经成功完成。

12 = 因为出现了错误，系统函数未执行。

只能为参数使用 HMI 变量。

可组态的对象

对象	事件
变量	值改变 超出上限 超出下限
功能键（全局）	释放 按下
功能键（局部）	释放 按下
画面	已加载 已清除

对象	事件
画面对象	按下 释放 单击 切换（或者拨动开关） 打开 关闭 启用 禁用
调度程序	到期

1.1.38 ImportDataRecordsWithChecksum

说明

从 CSV 文件导入配方的一个或全部带校验和的数据记录，并对校验和进行验证。

在函数列表中使用

导入带有校验和的数据记录（文件名称，数据记录编号/名称，覆盖，输出状态消息，处理状态，verify checksum）

在用户自定义函数中使用

ImportDataRecordsWithChecksum File_name, Data_record_number_or_name, Overwrite, Output_status_message, Processing_status, Verify_checksum

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

文件名称

要导入其配方数据记录的 CSV 文件的名称。输入路径和文件扩展名，例如“C:\TEMP\Orange.CSV”。

如果要保存到存储卡中，则按如下方式指定存储位置：“\StorageCard\<FileName>”。

如果指定目录而不是单个的 CSV 文件，则会导入指定目录中的所有文件。

数据记录编号/名称

要导入的配方数据记录的编号或名称。如果要导入所有的配方数据记录，则指定“0”。

覆盖

确定是否覆盖现有的配方数据记录：

0 (hmiOverwriteForbidden) = 否：不覆盖配方数据记录。将不执行导入过程。

1 (hmiOverwriteAlways) = 是：不进行确认提示即覆盖配方数据记录。

2 (hmiOverwriteWithConfirmation) = 需要确认：配方数据记录在确认后才被覆盖。

输出状态消息

确定在导入后是否输出状态消息：

0 (hmiOff) = 关：不输出状态消息。

1 (hmiOn) = 开：输出状态消息。

处理状态

返回系统函数的处理状态。例如，使用返回值，以确保在执行其它系统函数前此系统函数已成功完成。

2 = 系统函数正在执行。

4 = 系统函数已经成功完成。

12 = 因为出现了错误，系统函数未执行。

只能为参数使用 HMI 变量。

Verify checksum

确定在导入过程中是否应验证校验和：

0 (hmiFalse) = 否：不验证校验和。

1 (hmiTrue) = 是：验证校验和。

1.1.39 IncreaseTag

描述

将给定值添加到变量值上。

X = X + a

说明

系统函数使用同一变量作为输入和输出值。当该系统函数用于转换数值时，必须使用辅助变量。可以使用"SetTag"系统函数为辅助变量指定变量值。

如果在报警事件中组态了该系统函数但变量未在当前画面中使用，则无法确保在 PLC 中使用实际的变量值。通过设置“连续循环”采集模式可以改善这种情况。

在函数列表中使用

增加变量（变量，值）

在用户自定义函数中使用

IncreaseTag Tag, Value

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

为其添加给定值的变量。

值

作为加数的数值。

1.1.40 InverseLinearScaling

描述

使用线性函数 X = (Y - b) / a，将通过给定变量 Y 的值计算得出的数值赋给变量 X。

变量 X 和 Y 不能相同。与此函数相反的系统函数是“线性转换”。

说明

变量 X 和 Y 不能相同。如果要将变量转换为其自身，必须使用一个辅助变量。
可用系统函数“设置变量”将要转换变量的值分配给辅助变量。

在函数列表中使用

转换线性转换 (X, Y, b, a)

在用户自定义函数中使用

InverseLinearScaling X, Y, b, a

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

X

要为其分配通过线性方程式计算得出的值的变量。

Y

包含用于计算的值的变量。

b

其值作为减数。

a

其值作为除数。

示例

下面的程序代码使用 InverseLinearScaling 函数为 varX 变量赋值。

```
{
BYTE varX;
BYTE Yvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

//Inverse linear scaling
 InverseLinearScaling (varX, Yvalue, bvalue, avalue);

printf ("varX = %d\r\n, varX);
...
}
```

保存的返回值可在后续代码中进行处理。

1.1.41 InvertBit

描述

对给定的"Bool"型变量的值取反。

- 如果变量具有值 1（真），它将被设置为 0（假）。
- 如果变量具有值 0（假），它将被设置为 1（真）。

在函数列表中使用

对位取反（变量）

在用户自定义函数中使用

InvertBit Tag

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

要设置其位的变量。

示例

以下程序代码会将布尔型变量 bStatus 的值取反，并将所得结果与原始 bSaved 值一起输出。

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Get current value
bValue=myTag.Value

'Save current value
bSaved=bValue

'Invert Bit
InvertBit myTag
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BOOL bStatus = 0;
BOOL bSaved = bStatus;

//Invert variable
invertBit(bStatus);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
  ...
}
```

1.1.42 InvertBitInTag

描述

对给定变量中的位取反：

- 如果变量中的位为值 1（真），它将被设置为 0（假）。
- 如果变量中的位为值 0（假），它将被设置为 1（真）。

在改变了给定位之后，系统函数将整个变量传送回 PLC。但是并不检查变量中的其它位是否同时改变。在所示变量被传回 PLC 之前，操作员和 PLC 对该变量只有只读权限。

说明

如果 PLC 支持 BOOL 型变量，不要使用该系统函数。而是使用“对位取反”系统函数。

在函数列表中使用

对变量中的位取反（变量，位）

在用户自定义函数中使用

InvertBitInTag Tag, Bit

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

要设置其给定位的变量。

位

要设置的位的编号。

在用户自定义函数中使用此系统函数时，变量中的位从右向左计数。计数从 0 开始。

示例

以下程序代码会将 bStatusWord 变量中指定的 bitposition 位取反，并将所得结果与原始 bSaved 值一起输出。

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Get current value
bValue=myTag.Value

'Save current value
bSaved=bValue

'Invert Bit in position
bitposition=2
InvertBit myTag, bitposition
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BYTE bStatusWord;
BYTE bsaved = bStatusWord;
BYTE bitposition = 2;

//Invert bit in bitposition
InvertBitInTag (bStatusWord, bitposition);
//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatusWord, bsaved);
  ...
}
```

1.1.43 LinearScaling

描述

为变量 Y 赋值，该变量通过线性函数 Y= (a *X) + b 利用给定变量 X 的值计算得出。

与此功能相反的系统函数是“转换线性转换”。

说明

变量 X 和 Y 不能相同。如果要将变量转换为其自身，必须使用一个辅助变量。
可用系统函数“设置变量”将要转换变量的值分配给辅助变量。

在函数列表中使用

线性转换 (Y, a, X, b)

在用户自定义函数中使用

LinearScaling Y, a, X, b

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

Y

要为其分配通过线性方程式计算得出的值的变量。

a

作为乘数的数值。

X

包含用于计算的值的变量。

b

作为加数的数值。

示例

下面的程序代码使用 LinearScaling 函数为 Yvar 变量赋值。

```
{
BYTE Yvar;
BYTE Xvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

// linear scaling
 LinearScaling ( Yvar, avalue, Xvalue, bvalue);

printf ("Yvar = %d\r\n, Yvar);
...
}
```

保存的返回值可在后续代码中进行处理。

1.1.44 LoadDataRecord

描述

将给定的配方数据记录从 HMI 设备的存储介质装载到配方变量中。例如，可使用该系统函数在配方画面中显示配方数据记录。

激活配方同步设置中的“同步配方视图和配方变量”(Synchronize recipe view and recipe tags) 选项。如果此选项已禁用，则系统函数无效。

在函数列表中使用

加载数据记录（配方编号/名称，数据记录编号/名称，处理状态）

在用户自定义函数中使用

LoadDataRecord Recipe_number_or_name, Data_record_number_or_name, Confirmation, Output_status_message, Processing_status

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

配方编号/名称

要装载其配方数据记录的配方的编号或名称。

数据记录编号/名称

加载的配方数据记录的编号或名称。

处理状态

返回系统函数的处理状态。例如，可以使用返回值延迟执行其它系统函数，直到本系统函数已经成功完成：

2 = 系统函数正在执行。

4 = 系统函数已经成功完成。

12 = 因为出现了错误，系统函数未执行。

只能为参数使用 HMI 变量。

1.1.45 LogOff

说明

在 HMI 设备上注销当前用户

在函数列表中使用

注销

在用户自定义函数中使用

Logoff

如果组态的设备支持用户自定义函数，则可以使用。 更多信息，请参考“设备相关性”。

参数

--

1.1.46 Logon

描述

在 HMI 设备上登录当前用户

在函数列表中使用

登录（密码，用户名）

在用户自定义函数中使用

Logon Password, User_name

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

密码

从中读取用户登录密码的变量。如果用户已经登录，则变量中的密码会删除。

用户名

从中读取用户登录用户名的变量。

1.1.47 LookupText

说明

从文本列表中标识一个条目。结果取决于值和所选定的运行系统语言。结果保存到数据类型为“String”的变量中。

在函数列表中使用

查找文本（输出文本，索引，语言，文本列表）

在用户自定义函数中使用

LookupText Output_text, Index, Language, Text_list

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

输出文本

要写入结果的变量。

索引

定义列表条目值的变量。

语言

定义标识列表条目所使用的运行系统语言。

- 运行系统语言
  语言代号按照 VBScript 参考，例如"de-DE"为德语（德国）或"en-US"为英语（美国）。此选择取决于激活何种运行系统语言。
- 变量
  包含该语言的变量。输入代表国家/地区标识号的十进制值作为运行系统语言的选择，例如，1031 代表德语 - 标准，1033 代表英语 - 美国。详细的介绍可从 VBScript 基础"Locale identifier (LCID) diagram"中获得。
- 整型
  对应进行语言切换时运行系统语言顺序的数字。
  此选择取决于已激活的运行系统语言，例如，"0"代表首次启动运行系统时显示的语言。有关详细信息，请参见"运行系统中的语言"主题。

文本列表

定义文本列表。列表条目从文本列表中读取。

1.1.48 NotifyUserAction

描述

此系统函数用于记录未在审计跟踪中自动记录的用户操作。也可以使用此系统功能，要求用户对操作员执行的操作进行确认、输入电子签名和注释。使用系统函数的要求是，在"运行系统设置 > GMP"(Runtime settings > GMP) 中激活符合 GMP 的组态。

如果在函数中使用"通知用户操作"系统函数，用于取消输入的"取消"单击操作可能会导致调试器打开。为补偿该响应，可在函数中使用"On Error Resume Next"语句。使用该指令时，出现运行错误后将执行下一条指令。如果使用"On Error Resume Next"语句，还会抑制系统事件的输出。

在函数列表中使用

通知用户操作（确认类型，强制注释，类别，对象名称，说明）

在用户自定义函数中使用

NotifyUserAction Confirmation_type, Mandatory_commenting, Category, Object_name, Description

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

Confirmation type

确定必须以何种方式确认操作

0 = (None)：无需确认，将在审计跟踪中创建一个条目

1 = (Acknowledgement)：确认，用户必须确认该操作；将在审计跟踪中创建一个条目

2 = (Digital Signature)：电子签名；打开相应的对话框窗口，用户必须在其中输入电子签名 - 将在审计跟踪中创建一个条目

Mandatory commenting

确定用户是否必须输入注释。注释将记录在审计跟踪中。

0 = (True)：True；打开一个对话框窗口，用户必须在其中输入注释

1 = (False)：False；无需注释

Category

修改过的对象的种类或类别名

对象名称

修改过的对象的名称

描述

描述归档用户操作的文本。

1.1.49 OpenAllLogs

描述

重新建立 WinCC 和日志之间的连接。 可继续记录。

说明

运行系统函数“开始记录”以重新开始记录。

在函数列表中使用

打开所有日志

在用户自定义函数中使用

OpenAllLogs

如果组态的设备支持用户自定义函数，则可以使用。 更多信息，请参考“设备相关性”。

参数

--

应用实例

处于运行状态时，希望更改记录过程值的数据介质。

有关组态的注意事项

在“Close Archive”按钮上组态系统函数“停止记录”和“关闭所有日志”。

在“Open Archive”按钮上组态系统函数“打开所有日志”和“开始记录”。

将要停止或开始的日志的相应名称作为参数进行传送。

HMI 设备上的操作步骤

当按下按钮“Close Archive”时，停止指定日志并关闭所有打开的日志。 可以改变数据介质。 用“Open Archive”按钮打开所有日志。 继续在指定日志中进行记录。

1.1.50 OpenCommandPrompt

描述

打开 Windows 系统提示。

例如，该函数用来复制文件或调用其它应用程序。

在函数列表中使用

打开命令提示符

在用户自定义函数中使用

OpenCommandPrompt

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

--

1.1.51 OpenControlPanel

描述

打开可用来编辑所选控制面板设置的对话框。

该系统函数允许您在 HMI 设备上进行以下设置：

- IP 地址的属性和值
- 网络上的用户标识
- WinCC Internet 设置

说明

项目安全

系统函数“OpenControlPanelDialog”用于在 HMI 设备中绕过安全模式。采取相应的预防措施确保项目的安全。

在函数列表中使用

打开控制面板对话框（对话框）

在用户自定义函数中使用

-

参数

对话框

设置要打开的控制面板对话框。

- PROFINET_X1：IP 地址和以太网参数的设置
- PROFINET_X3：IP 地址和以太网参数的设置；仅适用于 Comfort Panel KP 1500、TP 1500、TP1900、TP2200
- WinCC Internet 设置：Web 服务器、电子邮件通知的设置（前提是 HMI 设备支持这些功能）
- 网络 ID：网络标识的设置（前提是 HMI 设备支持这些功能）

1.1.52 OpenInternetExplorer

描述

在 HMI 设备上打开 Internet Explorer。

如果在调用系统函数时 Internet Explorer 已打开，那么 Internet Explorer 会关闭并再次打开。

说明

Internet Explorer 将数据临时保存在 HMI 设备 DRAM 文件系统中，例如，上一次调用的网站。该数据可使用系统函数“备份 RAM 文件系统”保存，这样在重启动 HMI 设备后其依然可用。

在函数列表中使用

打开 Internet Explorer（起始页）

在用户自定义函数中使用

OpenInternetExplorer Start_page

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

起始页

启动 Internet Explorer 时载入的页面，例如 "www.siemens.com"。

1.1.53 OpenScreenKeyboard

说明

隐藏或显示屏幕键盘。

屏幕键盘保持打开直到它被明确关闭。这样，屏幕键盘也可用于其它应用程序。

在函数列表中使用

打开屏幕键盘 (显示模式)

在用户自定义函数中使用

OpenScreenKeyboard Display_mode

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

显示模式

指定使用屏幕键盘打开的窗口是最小化还是最大化：

0 (hmiScreenKeyboardMinimized) = 最小化

1 (hmiScreenKeyboardMaximized) = 最大化

1.1.54 OpenTaskManager

描述

显示任务管理器。

任务管理器允许切换到 HMI 设备上的其它打开的应用程序。

说明

任务管理器的外观取决于所安装的操作系统。

在函数列表中使用

打开任务管理器

在用户自定义函数中使用

OpenTaskManager

如果组态的设备支持用户自定义脚本，则可以使用。更多信息，请参考“设备相关性”。

参数

--

1.1.55 PageDown

描述

在 HMI 设备上执行 <Pagedown> 键功能：

此系统函数只能用于功能键。

在函数列表中使用

向下翻页

在用户自定义函数中使用

PageDown

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

-

1.1.56 PageUp

描述

在 HMI 设备上执行 <PageUp> 键功能：

此系统函数只能用于功能键和带时间触发器的任务。

在函数列表中使用

向上翻页

在用户自定义函数中使用

PageUp

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

-

1.1.57 PrintReport

描述

通过连接到 HMI 设备的打印机来打印给定的报表。报表将按 HMI 设备上所设置的语言进行打印。

说明

如果在使用系统函数打印日志数据时关闭运行系统，则运行系统停止后，将立即停止向报告传送数据。

在函数列表中使用

打印报告（报表）

在用户自定义函数中使用

PrintReport Report

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

报表

要打印的报表的名称。

说明

如果已经使用“函数列表”对话框为“PrintReport”函数设置了一个新的报表，则在编译时将显示以下警告：”报表“Report_1”没有打印页。
要对该警告采取纠正措施，请通过项目视图打开“Report_1”，然后重新编译该项目。

1.1.58 PrintScreen

描述

通过连接到 HMI 设备的打印机来打印当前显示在 HMI 设备上的画面。

同时打印已打开的窗口。

说明

从 Windows 操作系统的当前设置中获取打印机设置。

在函数列表中使用

截屏

在用户自定义函数中使用

PrintScreen

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

--

1.1.59 ResetBit

描述

将“Bool”型变量的值设置为 0（假）。

在函数列表中使用

复位（变量）

在用户自定义函数中使用

ResetBit Tag

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

设置为 0（假）的 BOOL 型变量。

示例

以下程序代码使用 ResetBit 函数将布尔型变量 bStatus 的值置 0，并将所得结果与原始 bSaved 值一起输出。

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Set value
bValue=1
myTag.Value=bValue

'Save current value
bSaved=bValue

'Reset Bit
ResetBit myTag
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BOOL bStatus = 1;
BOOL bSaved = bStatus;

//Reset bit
  ResetBit (bStatus);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
...
}
```

1.1.60 ResetBitInTag

描述

将指定变量中的一个位设置为 0（假）。

在改变了给定位之后，系统函数将整个变量传送回 PLC。但是并不检查变量中的其它位是否同时改变。在所示变量被传回 PLC 之前，操作员和 PLC 对该变量只有只读权限。

说明

如果 PLC 支持 BOOL 型变量，不要使用该系统函数。而使用系统函数“复位”。

在函数列表中使用

复位变量中的位（变量，位）

在用户自定义函数中使用

ResetBitInTag Tag, Bit

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

其中一个位要设置为 0（假）的变量。

位

要设置为 0（假）的位的编号。

在用户自定义函数中使用此系统函数时，无论使用何种 PLC，指定变量中的位都是从右向左计数。计数从 0 开始。

示例

以下程序代码会将 bStatusWord 变量中指定的 bitposition 位置 0，并将所得结果与原始 bSaved 值一起输出。

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Save current value
bValue=myTag.Value
bSaved=bValue

'Reset Bit
bitposition=2
ResetBitInTag myTag, bitposition
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{

BYTE bSaved;
BYTE bitposition = 2;

bSaved = GetTagByte("bStatusWord");

//Reset bit in bitposition
ResetBitInTag ("bStatusWord", bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",GetTagByte("bStatusWord"), bSaved);
  ...
}
```

1.1.61 SafelyRemoveHardware

描述

检查是否具有对外部存储介质的读写访问权。如果没有，则可以断开外部存储介质，而不会丢失数据。

在函数列表中使用

安全卸下硬件（路径，结果）

在用户自定义函数中使用

SafelyRemoveHardware Path, Result

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

结果

要在其中写入结果的变量。

TRUE：可以安全断开存储介质。将输出相应的系统报警。

FALSE：无法安全移除存储介质。将输出相应的系统报警。

路径

存储介质路径，例如 \Storage Card USB\

1.1.62 SaveDataRecord

说明

将配方变量的当前值作为数据记录保存到 HMI 设备的存储介质中。

例如，可使用该系统函数来保存配方画面中的配方数据记录。

仅当已在配方中激活“同步”(Synchronization) 选项时，才会保存数据记录。

- 如果未激活“同步”(Synchronization) 选项，则不会保存数据记录。
  要保存数据记录，请使用配方控件或系统功能中的“保存”(Save) 按钮。

同步	按钮	系统函数
激活	保存数据记录。	保存数据记录。
禁用	仅当显示配方视图的列表视图时，才会保存数据记录。	不保存数据记录。 “12”作为处理状态返回。

- 不会显示与操作相关的报警。

在函数列表中使用

保存数据记录（配方编号/名称，数据记录编号/名称，覆盖，输出状态消息，处理状态）

在用户自定义函数中使用

SaveDataRecord Recipe_number_or_name, Data_record_number_or_name, Overwrite, Output_status_message, Processing_status

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

配方编号/名称

要将配方数据记录保存到其中的配方的编号或名称。

数据记录编号/名称

保存的配方数据记录的编号或名称。如果没有在配方中找到该名称或编号的记录，将会创建一条新数据记录，而与“Overwrite”参数的值无关。

覆盖

指定是否覆盖现有的数据记录：

0 (hmiOverwriteForbidden) = 否：不覆盖配方数据记录，也不保存该数据记录。

1 (hmiOverwriteAlways) = 是：不进行确认提示即覆盖配方数据记录。

2 (hmiOverwriteWithConfirmation) = 需要确认：只有经用户确认后才会覆盖配方数据记录。

输出状态消息

确定在保存后是否输出状态消息：

0 (hmiOff) = 关：不输出状态消息。

1 (hmiOn) = 开：输出状态消息。

处理状态

返回系统函数的处理状态。例如，可以使用返回值延迟执行其它系统函数，直到本系统函数已经成功完成。

2 = 系统函数正在执行。

4 = 系统函数已经成功完成。

12 = 因为出现了错误，系统函数未执行。

只能为参数使用 HMI 变量。

1.1.63 SendEMail

描述

从 HMI 设备发送电子邮件到给定地址。

例如，该系统函数用于在维修时将报警直接传递给维修技术人员。

说明

要用电子邮件发送报警，HMI 系统必须具有一个可由其随意支配的电子邮件客户端。

在函数列表中使用

发送电子邮件（地址，主题，文本，收件人地址）

在用户自定义函数中使用

SendEMail Address, Subject, Text, Return_address

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

地址

收件人的电子邮箱地址。

Subject

电子邮件的主题信息。

文本

要用电子邮件发送的文本。

Return address

该电子邮件的收件人应将回复发送到的电子信箱地址。

1.1.64 SetAcousticSignal

描述

在 HMI 设备上组态触摸屏操作的声反馈。

说明

在断开状态下建立的组态将在重启 HMI 设备时重新建立。

在函数列表中使用

设置声音信号（音量）

在用户自定义函数中使用

SetAcousticSignal Volume

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

音量

确定声音信号是否发出以及发出多大的声响：

-1 (hmiToggle) = 切换：对发出的声音信号进行如下切换：静音 > 轻声 > 高声。

0 (hmiMuted) = 静音：没有声音信号

1 (hmiQuiet) = 安静：较轻的声音信号

2 (hmiLoud) = 响亮：较响的声音信号

1.1.65 SetAlarmReportMode

描述

确定是否将报警自动报告到打印机上。

在函数列表中使用

设置报警报告模式（模式）

在用户自定义函数中使用

SetAlarmReportMode Mode

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

模式

确定报警是否自动报告到打印机上：

0 (hmiDisablePrinting) = 报表关闭：报警不自动打印。

1 (hmiEnablePrinting) = 报表打开：报警自动打印。

-1 (hmiToggle) = 切换：在两种模式之间切换。

1.1.66 SetBit

描述

将“Bool”型变量的值设置为 1（真）。

在函数列表中使用

置位（变量）

在用户自定义函数中使用

SetBit Tag

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

要将其值设置为 1（真）的 BOOL 型变量。

示例

以下程序代码使用 SetBit 函数将布尔型变量 bStatus 的值置 1，并将所得结果与原始 bSaved 值一起输出。

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Set value
bValue=0
myTag.Value=bValue

'Save current value
bSaved=bValue

'Set Bit
SetBit myTag
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BOOL bStatus = 0;
BOOL bSaved = bStatus;

//Set bit
  SetBit (bStatus);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
...
}
```

1.1.67 SetBitInTag

描述

将给定变量中的一个位设置为 1（真）。

在改变了给定位之后，系统函数将整个变量传送回 PLC。但是并不检查变量中的其它位是否同时改变。在所示变量被传回 PLC 之前，操作员和 PLC 对该变量只有只读权限。

说明

如果 PLC 支持 BOOL 型变量，不要使用该系统函数。而使用系统函数“置位”。

在函数列表中使用

置位变量中的位（变量，位）

在用户自定义函数中使用

SetBitInTag Tag, Bit

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

要将其中的一个位设置为 1（真）的变量。

位

要设置为 1（真）的位的编号。

在用户自定义函数中使用此系统函数时，无论使用何种 PLC，指定变量中的位都是从右向左计数。计数从 0 开始。

说明

要实现可靠的功能，必须保证与实际过程值一起使用的变量的更新。因此，应在 I/O 域中组态变量或将系统函数分配给画面对象（如按钮）。

如果为系统函数组态了短事件（如激活报警），则只能通过设置连续读取的变量访问实际过程值。

示例

以下程序代码会将 bStatusWord 变量中指定的 bitposition 位置 1，并将所得结果与原始 bSaved 值一起输出。

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Save current value
bValue=myTag.Value
bSaved=bValue

'Set Bit in tag
bitposition=1
SetBitInTag "bStatusWord", bitposition
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "& bSaved & "New Value: " & bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{

BYTE bSaved;
BYTE bitposition = 1;

bSaved = GetTagByte("bStatusWord");

//Reset bit in bitposition
SetBitInTag ("bStatusWord", bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",GetTagByte("bStatusWord"), bSaved);

}
```

1.1.68 SetBrightness

描述

确定显示亮度。

说明

在控制面板/启动中心中设置的组态在重启 HMI 设备时将重新建立。

对于第二代基本面板、移动面板和精智面板：

系统函数“SetBrightness”的值可设为 0% 到 100%。设置值将传送到 HMI 设备。可在“启动中心 > 设置 > 显示”(Start Center > Settings > Display) 中查看和编辑 HMI 设备的亮度设置。HMI 设备支持 10% 到 100% 范围内的亮度设置。

如果为系统函数“SetBrightness”分配值 0%，默认情况下运行系统将关闭 HMI 设备的显示屏。操作员触摸显示屏，显示屏将切换至上一个亮度设置。

如果为系统函数“SetBrightness”分配的值介于 1% 和 10% 之间，同时操作员在“启动中心”(Start Center) 打开了显示屏设置，亮度将复位为 10%。

在函数列表中使用

设置亮度（值）

在用户自定义函数中使用

SetBrightness Value

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

值

新亮度值。

1.1.69 SetConnectionMode

描述

连接或断开给定的连接。

说明

只有在 HMI 设备上设置了“在线”操作模式后，才能与 PLC 建立连接。使用系统函数“设置设备模式”。

如果连接是在“离线”模式下加载的，则每次模式切换为“离线”时，都将再次关闭连接。要在切换为“在线”模式后重新建立连接，需将连接再次设为“在线”模式。

在函数列表中使用

设置连接模式（模式，连接）

在用户自定义函数中使用

SetConnectionMode Mode, Connection

说明

良好生产实践

在用户定义的函数中使用“SetConnectionMode”系统函数不符合 GMP。

在用户定义的函数中使用“SetConnectionMode”系统函数时，请在 HMI 设备的运行时设置中禁用良好生产实践。

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

模式

指定是建立还是断开至 PLC 的连接。对于使用 Windows CE 的设备和 Runtime Advance，数据类型 BOOL 的变量不支持该参数。

0 (hmiOnline) = 在线：建立连接。

1 (hmiOffline) = 离线：断开连接。

连接

与 HMI 设备相连的 PLC。在连接编辑器中指定 PLC 的名称。

在用户自定义函数中多次使用系统函数

如果将“设置连接模式”系统函数用于不同的连接，则可能无法正确执行所有系统函数。为防止出现此类情况，请执行以下步骤：

1. 使用起始值“0”创建变量。
2. 在 HMI 变量的“数值更改”事件上组态“设置连接模式”系统函数。例如，如果要断开三个连接，则必须组态该系统函数三次。
3. 在用户自定义函数中，将“对位取反”系统函数应用于 HMI 变量。

应用实例

此系统函数的两个典型应用实例如下：

- 测试
  只要没有 PLC 与 HMI 设备相连，在 HMI 设备上进行测试期间将不会有错误消息输出。如果 HMI 设备与 PLC 相连，则它与 PLC 的连接可通过按键来建立。
- 调试
  要为一个系统组态多个 PLC。首先，将除一个之外的所有 PLC 组态为“离线”。调试第一个 PLC 之后，可通过按键来建立与其它所有 PLC 的连接。用这种方法，可以依次启动其它 PLC。

1.1.70 SetDataRecordTagsToPLC

描述

将配方变量的值传送到 PLC。该配方变量包含显示在 HMI 设备上的数据记录的值。

在函数列表中使用

将数据记录变量设置到 PLC（配方编号/名称，处理状态）

在用户自定义函数中使用

SetDataRecordTagsToPLC Recipe_ number_or_name, Processing_status

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

配方编号/名称

要将其配方数据记录传送到 PLC 的配方的编号或名称。

处理状态

返回系统函数的处理状态。例如，可以使用返回值延迟执行其它系统函数，直到本系统函数已经成功完成：

2 = 系统函数正在执行。

4 = 系统函数已经成功完成。

12 = 因为出现了错误，系统函数未执行。

只能为参数使用 HMI 变量。

1.1.71 SetDataRecordToPLC

描述

将给定的配方数据记录从 HMI 设备的数据介质直接传送到与 HMI 设备相连的 PLC。

说明

配方数据记录的值不需要显示在 HMI 设备上。

在函数列表中使用

将数据记录设置到 PLC（配方编号/名称，数据记录编号/名称，输出状态消息，处理状态）

在用户自定义函数中使用

SetDataRecordToPLC Recipe_number_or_name, Data_record_number_or_name, Output_status_message, Processing_status

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

配方编号/名称

要将其配方数据记录传送到 PLC 的配方的编号或名称。

数据记录编号/名称

传送到 PLC 的配方数据记录的编号或名称。

输出状态消息

确定传送之后是否输出状态消息：

0 (hmiOff) = 关：不输出状态消息。

1 (hmiOn) = 开：输出状态消息。

处理状态

返回系统函数的处理状态。例如，可以使用返回值延迟执行其它系统函数，直到本系统函数已经成功完成：

2 = 系统函数正在执行。

4 = 系统函数已经成功完成。

12 = 因为出现了错误，系统函数未执行。

只能为参数使用 HMI 变量。

1.1.72 SetDaylightSavingTime

描述

系统函数“设置夏令时时间”可将 HMI 设备从夏时制改为标准时间，或从标准时间改为夏时制。

执行该系统函数后，时间设置将立即生效。

说明

系统函数“设置夏令时时间”不支持不使用夏令时的时区。

说明

Windows 7

系统函数“设置夏令时时间”在使用 Windows 7 的基于 PC 的 HMI 设备中不支持。

在函数列表中使用

设置夏令时时间(夏令时时间)

在用户自定义函数中使用

SetDaylightSavingTime Daylight_saving_time

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

夏令时

确定是否在 HMI 设备上设置了夏令时：

0 = 未激活夏令时

1 = 已激活夏令时。

1.1.73 SetDeviceMode

描述

切换 HMI 设备上的操作模式。可以有下列操作类型：“在线”、“离线”和“加载”。

在函数列表中使用

设置设备模式（操作模式）

在用户自定义函数中使用

SetDeviceMode Operating_mode

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

操作模式

指定 HMI 设备的运行模式。对于使用 Windows CE 的设备，数据类型 BOOL 的变量不支持该参数。

0 (hmiOnline) = 在线：建立至 PLC 的连接。始终在该过程中设置已组态的连接状态。不考虑在运行系统中最近使用的状态。

1 (hmiOffline) = 离线：断开至 PLC 的连接。

2 (hmiTransfer) = 加载：项目可以从组态计算机传送到 HMI 设备。

说明

如果使用 PC 作为 HMI 设备，在“加载”后切换工作模式时，运行系统软件将退出。

1.1.74 SetDisplayMode

说明

改变运行系统软件所在运行的屏幕设置。

运行系统软件在缺省情况下以全屏模式运行。Windows 任务切换将被禁用。

在函数列表中使用

设置显示模式（布局）

在用户自定义函数中使用

SetDisplayMode Display_mode

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

显示模式

确定运行系统软件所在运行的屏幕设置。

1 (hmiScreenFull)：全屏：画面的标题栏不可见

2 (hmiScreenMaximized)：最大化 (Maximized)

3 (hmiScreenRestore)：恢复：使用上次采用的画面设置。只有窗口显示为最小化或最大化时，才能使用该布局。

4 (hmiScreenMinimized)：最小化

5 (hmiScreenAutoAdjust)：自动：设置窗口大小以显示其中的所有画面对象。

6 (hmiScreenOnTop):前景；根据 Windows 设置，窗口在前景中显示或与窗口相关的任务栏上的程序图标闪烁。设置可以在 Windows 组态中改变，并适用于所有 Windows 应用程序。

1.1.75 SetLanguage

描述

切换 HMI 设备上的语言。所有组态的文本和系统事件以新设置的语言显示在 HMI 设备上。

在函数列表中使用

设置语言（语言）

在用户自定义函数中使用

SetLanguage Language

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

语言

确定在 HMI 设备上设置了哪种语言。有下列规范：

- -1 (hmiToggle) = 切换：切换到下一种语言。组态期间在“项目语言”编辑器中确定顺序。
- 在“运行系统设置”编辑器的“语言和字体”中定义的编号。切换到带有给定编号的语言。

- 在“运行系统设置”编辑器的“语言和字体”中定义的语言。
- 按照 VBScript5 参考的语言缩写：这样可切换到与指定语言代码相对应的语言，例如“de-DE”表示德语（德国）或“en-US”表示英语（美国）。
  在 VBScript 基本信息的主题“区域图标识号 (LCID) 图”下具有语言缩写总览。

1.1.76 SetPLCDateTime

说明

更改所连 PLC 的日期和时间

只能为以下 PLC 组态系统函数“SetPLCDateTime”：

- SIMATIC S7-1200
- SIMATIC S7-1500

在函数列表中使用

设置 PLC 日期时间（连接，时间）

在用户自定义函数中使用

-

参数

连接

PLC 和 HMI 设备的连接。

时间

将 HMI 设备的日期和时间传送至 PLC。PLC 应用 HMI 设备的日期和时间。

1.1.77 SetRecipeTags

描述

将配方变量的状态从“在线”改为“离线”，或从“离线”改为“在线”。

该系统函数用于例如在启动设备时需要对配方数据记录值进行精确调整的时候。

在函数列表中使用

设置配方变量（配方编号/名称，状态，输出状态消息，处理状态）

在用户自定义函数中使用

SetRecipeTags Recipe_number_or_name, Status, Output_status_message, Processing_status

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

配方编号/名称

要保存其配方数据记录的配方的编号或名称。

状态

确定配方变量的状态：

0 (hmiOnline) = 在线：配方变量的数值改变立即传送到与 HMI 设备相连的 PLC。

1 (hmiOffline) = 离线：只在执行了如“将数据记录变量设置到 PLC”系统函数时，配方变量的数值改变才传送到与 HMI 设备相连的 PLC。

输出状态消息

确定在保存后是否输出状态消息：

0 (hmiOff) = 关：不输出状态消息。

1 (hmiOn) = 开：输出状态消息。

处理状态

返回系统函数的处理状态。例如，可以使用返回值延迟执行其它系统函数，直到本系统函数已经成功完成：

2 = 系统函数正在执行。

4 = 系统函数已经成功完成。

12 = 因为出现了错误，系统函数未执行。

只能为参数使用 HMI 变量。

1.1.78 SetScreenKeyboardMode

描述

允许或禁止 HMI 设备上屏幕键盘的自动显示。

该系统函数也可用于避免显示画面键盘，因为外部键盘已连接到 HMI 设备。

说明

要在 HMI 设备而不是触摸面板设备上启用系统函数“设置屏幕键盘模式”(“SetScreenKeyboardMode”)，请在设备设置的“运行系统设置”对话框中选择“使用画面键盘”复选框。

在函数列表中使用

设置屏幕键盘模式（模式）

在用户自定义函数中使用

SetScreenKeyboardMode Mode

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

模式

确定隐藏还是显示屏幕键盘：

0 (hmiOff) = 关：隐藏屏幕键盘

1 (hmiOn) = 开：显示屏幕键盘

-1 (hmiToggle) = 切换：在两种模式之间切换。

1.1.79 SetTag

描述

将新值赋给给定的变量。

说明

该系统函数可用于根据变量类型分配字符串和数字。

在函数列表中使用

设置变量（变量，值）

在用户自定义函数中使用

SetTag Tag, Value

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

为其分配给定值的变量。

值

为给定变量所赋的值。

说明

“SetTag”系统函数只能在建立连接后执行。

示例

下面的程序代码使用 SetTag 函数将 gs_tag_bit 变量的值设置为 TRUE，并将返回值保存到 ok 变量中。

```
{
BOOL ok;
BOOL bvalue;

//Set the tag to true
ok = SetTag("gs_tag_bit", TRUE);
//error handling
if(ok)
{
  // succeeded
  printf ( "Function has run through.\r\n" );
bvalue = GetTagBit("gs_tag_bit");
  printf ("Value of gs_tag_bit: %d\r\n", bvalue);
}
else
{
  // failed
  printf ( "Error - function failed." );
}
...
}
```

保存的返回值可在后续代码中进行处理。

1.1.80 ShiftAndMask

说明

此系统函数将源变量的输入位组合格式转换成目标变量的输出位组合格式。这包括移动位和屏蔽位。

说明

如果源变量和目标变量位数不同，则在目标变量中使用系统函数会导致超出取值范围。

在函数列表中使用

移位和掩码（源变量，目标变量，待移动的位数，待屏蔽的位数）

在用户自定义函数中使用

ShiftAndMask Source_tag, Target_tag, Bits_to_shift, Bits_to_mask

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

源变量

变量包括输入位组合格式。整型变量，如，“Byte”、“Char”、“Int”、“UInt”、“Long”和“ULong”。

示例：16 位整型源变量，设置当前实际值为 72：0000000001001000。

目标变量

输出位组合格式保存在变量中。整型变量，如，“Byte”、“Char”、“Int”、“UInt”、“Long”和“ULong”。

示例：移动的输入位组合格式通过逻辑 AND 逐位运算与位屏蔽相乘：0000000000001001。结果为十进制值“8”，保存在目标变量中。

请注意以下事项：

- 源变量和目标变量位数相同。
- 要移动的位数应少于源变量和目标变量中的位数。
- 要屏蔽的位不能多于源变量和目标变量的位数。

待移动的位数

输入位组合格式向右侧移动的位数。负值意味着将输入位组合格式向左侧移动。

示例：“待移动的位数”的值为“+3”。当调用该系统函数时，输入位组合格式向右移动 3 位：0000000000001001。
左侧的各位用“0”进行填充。右侧将截掉 3 位。新的十进制值为“9”。

说明

如果源变量为有符号的整型数据类型，且带有符号“-”，则最左侧位为“1”。符号位向右侧移动后，该位将由“0”填充。符号变为“+”。

待屏蔽的位数

屏蔽位是一个整数。其位组合格式用于与移动的输入位组合格式相乘。示例：整数“2478”的位组合格式为“0000100110101110”。

可通过以下三种方式输入屏蔽位：

- 十六进制：首先输入前缀“0h”或者“0H”，后面可跟一个空格以方便阅读。然后将位组合格式分为四个组 (0000)(1001)(1010)(1110) 且将每个块改为 16 位编码：(0)(9)(A)(E)。只允许使用字符 0-9, A-F, a-f：“0h 09AE”。
- 二进制：首先输入前缀“0b”或者“0B”，后面可跟一个空格以方便阅读。然后将二进制位组合格式分为四个块 0000 1001 1010 1110 且用空格隔开以便检查：只允许使用字符“0”或者“1”：“0b 0000 1001 1010 1110”。
- 十进制：直接输入值“2478”，不加任何前缀。

说明

如果组态后更改 HMI 目标设备的设备版本（如，将“13.1.0”更改为“13.0.0”或相反），则必须对该系统函数的参数进行检查和测试。

在设备版本 V13.1.0 及更高版本中，“源变量”和“目标变量”参数支持数据类型“Char”和“Word”。在设备版本 V13.1.0 及以下版本中，这些参数必须指定为其它数据类型：

- “SInt”，而非“Char”
- “Int”，而非“Word”

否则，可能导致意外结果。如，所组态的系统函数出错或意外结果。

1.1.81 ShowAlarmWindow

描述

隐藏或显示 HMI 设备上的报警窗口。

在函数列表中使用

ShowAlarmWindow (Object name, Display mode)

在用户自定义函数中使用

ShowAlarmWindow Object_name, Display_mode

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

对象名称

要隐藏或显示的报警视图的名称。

Display mode

确定隐藏或显示报警窗口：

0 (hmiOff) = 关：隐藏报警视图

1 (hmiOn) = 开：显示报警视图

-1 (hmiToggle) = 切换：在两种模式之间切换

1.1.82 ShowOperatorNotes

应用

显示针对所选对象而组态的工具提示。

如果功能键上组态了该系统函数，则显示当前具有焦点的画面对象的工具提示。如果为画面自身组态了工具提示，则可以通过按 <Enter> 或双击帮助窗口切换到此文本。

如果按钮上组态了该系统函数，则只显示当前画面的工具提示。如果在按钮自身上组态了工具提示，则最初只显示按钮的工具提示。可以按 <Enter> 或双击帮助窗口切换到当前画面的工具提示。

说明

在帮助窗口打开期间，无法使用任何其它画面对象。要使用画面对象，请关闭帮助窗口。

关闭帮助窗口

可按照下列方式关闭帮助窗口：

使用按键：

- 通过再次按<HELP>键
- 通过按<ESC>键

使用触摸屏：

- 按 ❎ 按钮

在函数列表中使用

显示操作员注释（布局）

在用户自定义函数中使用

ShowOperatorNotes Display_mode

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

显示模式

确定隐藏还是显示已组态的工具提示：

0 (hmiOff) = 关：隐藏已组态的工具提示

1 (hmiOn) = 开：显示已组态的工具提示

-1 (hmiToggle) = 切换：在两种模式之间切换

1.1.83 ShowPopUpScreen

描述

例如，按下按钮后将打开弹出画面。

可输入一个常量值或分配一个变量作为坐标。
如果组态的弹出画面不可见或仅部分可见，则坐标设置为 0.0。

无论永久区域的大小如何，坐标的起始位置始终为 0.0。

在函数列表中使用

显示弹出画面 (画面名称, X 坐标, Y 坐标, 显示模式, 动画, 动画速度)

在用户自定义函数中使用

ShowPopupScreen Screen_name, X_coordinate, Y_coordinate, Display_mode, Animation, Animation_speed

参数

画面名称

指定按下按钮时运行系统中显示的弹出画面的名称。

X 坐标

弹出画面在当前画面中 X 轴方向的位置

Y 坐标

弹出画面在当前画面中 Y 轴方向的位置

显示模式

指定弹出画面的模式：

切换

关

开

动画

指定弹出画面的弹出方向：

关

左侧

顶部

右侧

底部

动画速度

指定弹出画面的弹出速度：

慢速

中等

快速

1.1.84 ShowPopupScreenSizable

描述

打开指定尺寸的弹出画面。使用该系统函数可打开不同尺寸的弹出画面，可使用显示的滚动条导航弹出画面中的内容。

可输入一个常量值或分配一个变量作为坐标。如果组态的弹出画面不可见或仅部分可见，则坐标设置为 0.0。无论永久区域的大小如何，坐标的起始位置始终为 0.0。

用户也可以定义弹出画面的宽度和高度，这因组态的弹出画面尺寸而异。这种情况下，滚动条即显示弹出画面中。

说明

HMI 设备按键操作

使用键盘快捷键 <ALT>+<箭头键> 可在弹出画面中滚动。

仅显示一个滚动条

如果在系统函数参数中仅指定了一个小于已组态弹出画面大小的尺寸，那么该弹出画面中会显示两个滚动条。可用两种方法来实现在弹出画面中仅组态一个滚动条：

- 根据具体设备的滚动条尺寸缩减已组态弹出画面的宽度/高度。
- 针对某个尺寸参数指定一个对应于弹出画面实际尺寸的值，然后加上具体设备的滚动条尺寸。根据需要指定其它尺寸值。

  示例：

  想要在尺寸为 500 x 300 的画面中打开一个带垂直滚动条的尺寸组态为 500 x 420 的弹出画面。将宽度参数指定为 534，该值对应于已组态弹出画面的实际尺寸与滚动条尺寸 (34) 之和。在高度参数中输入值 300。弹出画面在运行系统中打开时的尺寸为 500 x 300，并且仅显示垂直滚动条。

设备类型（版本 V14）	宽度	高度
带 4 英寸显示屏的 HMI 设备 带 19 英寸显示屏的精智面板 Runtime Advanced	25	25
带 9 英寸显示屏的 HMI 设备 带 15 英寸显示屏的精智面板 带 22 英寸显示屏的精智面板	27	27
带 7 英寸显示屏的 HMI 设备 带 12 英寸显示屏的 HMI 设备	34	34

在函数列表中使用

显示可调整大小的弹出画面（画面名称，X 坐标，Y 坐标，宽度，高度，显示模式，动画，动画速度）

在用户自定义函数中使用

ShowPopupScreenSizable Screen_name, X_coordinate, Y_coordinate, Width, Height, Display_mode, Animation, Animation_speed

参数

画面名称

指定按下按钮时运行系统中显示的弹出画面的名称。

X 坐标

弹出画面在当前画面中 X 轴方向的位置

Y 坐标

弹出画面在当前画面中 Y 轴方向的位置

宽度

指定弹出画面的宽度。最大宽度不得超出已组态 HMI 设备的屏幕宽度。

高度

指定弹出画面的高度。最大高度不得超出已组态 HMI 设备的屏幕高度。

显示模式

指定弹出画面的模式：

切换

关

开

动画

指定弹出画面的弹出方向：

关

左侧

顶部

右侧

底部

动画速度

指定弹出画面的弹出速度：

慢速

中等

快速

1.1.85 ShowSlideInScreen

描述

例如，通过操作按钮来调用滑入画面。

在函数列表中使用

显示滑入画面（画面名称，模式）

在用户自定义函数中使用

ShowSlideInScreen SlideInScreen_name, Mode

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

画面名称

指定按下按钮时运行系统中显示的滑入画面：

顶部滑入画面

底部滑入画面

左侧滑入画面

右侧滑入画面

模式

指定滑入画面的模式：

切换

关

开

1.1.86 ShowSoftwareVersion

描述

隐藏或显示运行系统软件的版本号。

例如，如果在维修期间需要所使用的运行系统软件的版本，可以使用该系统函数。

在函数列表中使用

显示软件版本 (显示模式)

在用户自定义函数中使用

ShowSoftwareVersion Display_mode

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

显示模式

确定是否显示版本号：

0 (hmiOff) = 关：不显示版本号

1 (hmiOn) = 开：显示版本号

-1 (hmiToggle) = 切换：在两种模式之间切换

1.1.87 ShowSystemAlarm

描述

显示作为系统事件传递到 HMI 设备的参数的值。

在函数列表中使用

显示系统报警（文本/值）

在用户自定义函数中使用

ShowSystemAlarm Text_or_value

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

文本/值

作为系统报警输出的文本或数值。

1.1.88 ShowSystemDiagnosticsWindow

描述

隐藏或显示 HMI 设备上的系统诊断窗口。系统诊断视图只在精智面板和 WinCC Runtime Advanced 的全局画面中可用。

在函数列表中使用

显示系统诊断窗口（画面对象）

在用户自定义函数中使用

ShowSystemDiagnosticsWindow Object_name

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

画面对象

要隐藏或显示的系统诊断窗口的名称。

1.1.89 StartLogging

描述

在指定日志中启动对数据或报警的记录。该函数也适用于审计跟踪。

可通过使用“停止记录”系统函数在运行系统时中断记录。

在函数列表中使用

开始记录（日志类型，日志）

在用户自定义函数中使用

StartLogging Log_type, Log

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

日志类型

确定日志的类型：

0 (hmiTagArchive) = 变量日志

1 (hmiAlarmArchive) = 报警日志

2 (hmiAudittrailArchive) = 审计跟踪。

日志

要启动的日志的名称。

1.1.90 StartNextLog

描述

停止在给定记录中记录数据或报警。

在为指定日志组态的分段循环日志的下一个日志中继续进行记录。

如果没有为指定日志组态分段循环日志，则该系统函数无效。

在函数列表中使用

启动下一次记录（日志类型，日志）

在用户自定义函数中使用

StartNextLog Log_type, Log

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

日志类型

确定日志的类型：

0 (hmiTagArchive) = 变量日志

1 (hmiAlarmArchive) = 报警日志

日志

停止进行记录的日志名称，记录将在下一个日志中继续。

1.1.91 StartProgram

描述

在 HMI 设备上启动指定程序。

运行系统软件仍然在后台运行。继续输出报警，且数据继续被更新。

当退出给定的应用程序时，在执行系统函数期间被激活的画面将显示在 HMI 设备上。

例如，可以在 HMI 设备上使用该系统函数编辑 MS Excel 中的配方数据记录。

说明

如果在 HMI 设备上安装了 Windows CE，则在组态期间，必须检查是否可以用该系统函数启动所期望的应用程序。
该系统函数允许在 Windows CE 的“执行”对话框中启动所有应用程序。
将要启动的应用程序必须安装在 HMI 设备上。

说明

如果未通过移动面板或精智面板的“StartProgram”函数执行应用程序，请检查系统功能的路径描述。在移动面板和精智面板上，一些应用程序位于文件夹“\flash\addons”。

在函数列表中使用

启动程序 (程序名称, 程序参数, 显示模式, 等待程序结束)

在用户自定义函数中使用

StartProgram Program_name, Program_parameters, Display_mode, Wait_for_program_to_end

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

程序名

要启动的程序的名称和路径。此参数区分大小写。

说明

如果路径中包含空格，则该程序仅当使用引号指定路径时才可以正确启动，例如"C:\Program Files\START\start.exe"。

程序参数

程序启动时所传送的参数，例如，在启动程序后打开的文件。

所需的参数说明请参见要启动的程序的文档。

显示模式

确定程序窗口如何显示在 HMI 设备上：

0 (hmiShowNormal) = 正常

1 (hmiShowMinimized) = 最小化

2 (hmiShowMaximized) = 最大化

3 (hmiShowMinimizedAndInactive) = 最小化且不激活

等待程序结束

确定所调用的程序在结束后是否切换回项目：

0 (hmiNo) = 否：不切换到项目。

1 (hmiYes) = 是：切换到项目。

说明

“等待程序结束”参数只适用于高级运行系统和面板。

1.1.92 StopLogging

描述

在指定日志中停止记录过程值或报警。该函数也适用于审计跟踪。

系统函数“开始记录”用于在运行系统时恢复记录。

说明

停止记录时，WinCC 与日志文件或日志数据库之间的连接仍然存在。使用系统函数“关闭所有日志”可断开此连接。

在函数列表中使用

停止记录（日志类型，日志）

在用户自定义函数中使用

StopLogging Log_type, Log

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

日志类型

确定日志的类型：

0 (hmiTagArchive) = 变量日志

1 (hmiAlarmArchive) = 报警日志

2 (hmiAudittrailArchive) = 审计跟踪。

日志

停止的日志的名称。

应用实例

处于运行状态时，希望更改记录过程值的数据介质。

有关组态的注意事项

在“Close Archive”按钮上组态系统函数“停止记录”和“关闭所有日志”。

在“Open Archive”按钮上组态系统函数“打开所有日志”和“开始记录”。

将要停止或开始的日志的相应名称作为参数进行传送。

HMI 设备上的操作步骤

当按下按钮“Close Archive”时，停止指定日志并关闭所有打开的日志。可以改变数据介质。“Open Archive”按钮可打开所有的日志并继续在指定的日志中进行记录。

1.1.93 StopRuntime

描述

退出运行系统软件，从而退出运行在 HMI 设备上的项目。

在函数列表中使用

停止运行系统（模式）

在用户自定义函数中使用

StopRuntime Mode

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

模式

确定在退出运行系统后操作系统是否关闭。

0 (hmiStopRuntime) = 运行系统：操作系统不关闭

1 (hmiStopRuntimeAndOperatingSystem) = 运行系统和操作系统：操作系统关闭(对于 WinCE 不适用)

示例

下面的程序代码将关闭运行系统和操作系统。

```
{

//Stop runtime and shutdown
StopRuntime (hmiStopRuntimeAndOperationSystem);

}
```

保存的返回值可在后续代码中进行处理。

1.1.94 TerminatePROFIsafe

描述

可通过断开 KTP 移动面板与 PLC 之间的 PROFIsafe 连接来进行故障安全操作。

执行"TerminatePROFIsafe"系统函数之后，可将 KTP 移动面板的连接器从 PLC 上移除，而且系统不会发出错误信号。

在函数列表中使用

终止 PROFIsafe

在用户自定义函数中使用

TerminatePROFIsafe

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

--

1.2 系统函数

1.2.1 ActivateScreen

描述

将画面切换到指定的画面。

使用"ActivateScreenByNumber"系统函数可以从根画面切换到永久性区域，反之亦然。

在函数列表中使用

激活画面（画面名称，对象编号）

在用户自定义函数中使用

ActivateScreen Screen_name, Object_number

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

画面名称

要切换到的画面的名称。

对象编号

画面切换后在指定画面中获得焦点的操作员控件元素。操作员控件元素的编号在组态期间使用 TAB 顺序确定。

在指定为"0"时：

- 如果调用该系统函数时焦点位于永久性区域，则永久性区域保留焦点。
- 如果调用该系统函数时焦点位于根画面，则指定画面中的第一个操作员控件元素获得焦点。

说明

如果将“到达边界”事件分配给"ActivateScreen"系统函数，则只有数值"0"对“对象编号”参数有效。活动对象不是由对象号定义的，而是由画面更改之前其 X 位置定义的。

示例

例如，单击按钮时，以下程序代码将使用 ActivateScreen 函数激活“Screen_2”。

```
Sub ActivateScreen_2()

'User-defined code
'' i. e. when pressing a button

ActivateScreen "Screen_2",0
```

1.2.2 ActivateScreenInScreenWindow

描述

将画面切换到在指定画面窗口中的指定画面。

在函数列表中使用

激活画面窗口中的画面（画面名称，画面窗口、新建画面名称）

在用户自定义函数中使用

ActivateScreenInScreenWindow Screen_name, Screen_window, New_screen_name

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“AUTOHOTSPOT”。

参数

画面名称

在画面窗口中显示的画面的名称。

画面窗口

要显示新画面的画面窗口的名称。

新建画面名称

要在画面窗口中显示的新画面的名称。

示例

单击按钮时，将使用以下程序代码中的 ActivateScreenInScreenWindow 函数激活“Screen_2”画面。

```
{
// User defined code
// i.e. when pressing a button
ActivateScreenInScreenWindow (GetParentScreen(screenName),
GetParentScreenWindow(screenName), "Screen_2");
...
}
```

1.2.3 DecreaseTag

描述

从变量值中减去指定的值。

X = X - a

说明

系统函数使用同一变量作为输入和输出值。当该系统函数用于转换数值时，必须使用辅助变量。可以使用“SetTag”系统函数为辅助变量指定变量值。

如果为报警事件组态了系统函数，但变量未在当前画面中使用，则无法确定在 PLC 中使用的实际变量值。通过设置“连续循环”采集模式可以改善这种情况。

在函数列表中使用

减少变量（变量，值）

在用户自定义函数中使用

DecreaseTag Tag, Value

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“AUTOHOTSPOT”。

参数

变量

要减去指定值的变量。

值

其值作为减数。

示例

下面的程序代码会以 value 变量的值为减量对 varX 变量的值进行递减。输入的值将保存在 old_value 变量中，并与新的 varX 值一起输出。

```
{
BYTE varX;
BYTE value;

//user input
...
BYTE old_value = varX;

//Decrease tag
DecreaseTag(varX, value);

//print original value and function result
printf ("User input: %i\r\n, Result of function DecreaseTag: %i\r\n", old_value, varX);
...
}
```

1.2.4 ExportImportUserAdministration

描述

将当前激活项目的用户管理中的全部用户导出到给定文件，或者将用户从给定文件导入到当前激活的项目中。

用户、用户口令和权限都保存在用户管理中。

在导入时，将覆盖所有用户。导入的用户立即生效。

说明

运行系统版本早于 V14 的 HMI 设备不支持从 V14 中导出和导入用户管理。

在函数列表中使用

导出导入用户管理（文件名称，方向）

在用户自定义函数中使用

ExportImportUserAdministration File_name, Direction

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

文件名称

包含密码的文件名称或者要写入密码的文件名称。输入文件位置和文件扩展名 (*.txt)，例如 “C:\TEMP\Passwords.txt”。

说明

如果要保存到存储卡中，按如下方式指定存储位置：“\StorageCard\<文件名>”。

传输方向

指定要导出还是导入密码：

0 (hmiExport) = 导出：导出密码。

1 (hmiImport) = 导入：导入密码。

1.2.5 获取父画面

说明

此函数用于确定特定画面路径中的父画面名称。

在函数列表中使用

获取父画面（画面）

在用户自定义函数中使用

GetParentScreen Screen

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

画面

传送的“画面”参数的结构必须对应图形系统为画面路径构成的结构：

<Screen_name>.<Screen_window_name>:<Screen_name>.<Screen_window_name>:<Screen_name>...

如果使用函数列表中的对象列表指定该参数，则会输入画面名称（而非画面路径）。

说明

“.”字符用于区分画面和画面对象名称。“:”字符用于层级结构化。因此，在名称标识中仅使用“-”或“_”作为定界符。

返回值

父画面的名称。

原理

“Screenwindow_1”画面窗口位于“Screen_1”画面中。在画面窗口中，该画面将变为包含“Screenwindow_2”画面窗口的“Screen_2”，依此类推。

画面路径：Screen_1.Screenwindow_1:Screen_2.Screenwindow_2:Screen_3

GetParentScreen 返回：

- Screen_2（针对在画面“Screen_3”中调用系统函数的情况）。
- Screen_1（针对在画面“Screen_2”中调用系统函数的情况）。

GetParentScreenwindow 返回：

- Screenwindow_2（针对在画面“Screen_3”中调用系统函数的情况）。
- Screenwindow_1（针对在画面“Screen_2”中调用系统函数的情况）。

1.2.6 获取父画面窗口

说明

此函数用于确定所传送的画面路径中的父画面窗口名称。

在函数列表中使用

获取父画面窗口 (画面)

在用户自定义函数中使用

GetParentScreenWindow Screen

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“设备相关性”。

参数

画面

传送的“画面”参数的结构必须对应图形系统为画面路径构成的结构：

<Screen_name>.<Screen_window_name>:<Screen_name>.<Screen_window_name>:<Screen_name>...

如果使用函数列表中的对象列表指定该参数，则会输入画面名称（而非画面路径）。

说明

“.”字符用于区分画面和画面对象名称。“:”字符用于层级结构化。因此，在名称标识中仅使用“-”或“_”作为定界符。

返回值

父画面窗口的名称。

原理

“Screenwindow_1”画面窗口位于“Screen_1”画面中。在画面窗口中，该画面将变为包含“Screenwindow_2”画面窗口的“Screen_2”，依此类推。

画面路径：Screen_1.Screenwindow_1:Screen_2.Screenwindow_2:Screen_3

GetParentScreen 返回：

- Screen_2（针对在画面“Screen_3”中调用系统函数的情况）。
- Screen_1（针对在画面“Screen_2”中调用系统函数的情况）。

GetParentScreenwindow 返回：

- Screenwindow_2（针对在画面"Screen_3"中调用系统函数的情况）。
- Screenwindow_1（针对在画面"Screen_2"中调用系统函数的情况）。

1.2.7 IncreaseTag

描述

将给定值添加到变量值上。

X = X + a

说明

系统函数使用同一变量作为输入和输出值。当该系统函数用于转换数值时，必须使用辅助变量。可以使用"SetTag"系统函数为辅助变量指定变量值。

如果为报警事件组态了系统函数，但变量未在当前画面中使用，则无法确定在 PLC 中使用的实际变量值。通过设置“连续循环”采集模式可以改善这种情况。

在函数列表中使用

增加变量（变量，值）

在用户自定义函数中使用

IncreaseTag Tag, Value

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见"AUTOHOTSPOT"。

参数

变量

为其添加给定值的变量。

值

作为加数的数值。

示例

下面的程序代码会以 value 变量的值为增量对 varX 变量的值进行递增。输入的值将保存在 old_value 变量中，并与新的 varX 值一起输出。

```
{
BYTE varX;
BYTE value;

//user input
...
BYTE old_value = varX;

//Increase tag
IncreaseTag(varX, value);

//print original value and function result
printf ("User input: %i\r\n, Result of function IncreaseTag: %i\r\n", old_value, varX);
...
}
```

1.2.8 InverseLinearScaling

描述

使用线性函数 X = (Y - b) / a，将通过给定变量 Y 的值计算得出的数值赋给变量 X。

变量 X 和 Y 不能相同。与此函数相反的系统函数是“线性转换”。

说明

变量 X 和 Y 不能相同。如果要将变量转换为其自身，必须使用一个辅助变量。
可用系统函数“设置变量”将要转换变量的值分配给辅助变量。

在函数列表中使用

转换线性转换 (X, Y, b, a)

在用户自定义函数中使用

InverseLinearScaling X, Y, b, a

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

X

要为其分配通过线性方程式计算得出的值的变量。

Y

包含用于计算的值的变量。

b

其值作为减数。

a

其值作为除数。

示例

下面的程序代码使用 InverseLinearScaling 函数为 varX 变量赋值。

```
{
BYTE varX;
BYTE Yvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

//Inverse linear scaling
 InverseLinearScaling (varX, Yvalue, bvalue, avalue);

printf ("varX = %d\r\n, varX);
...
}
```

保存的返回值可在后续代码中进行处理。

1.2.9 InvertBit

描述

对给定的“Bool”型变量的值取反。

- 如果变量具有值 1（真），它将被设置为 0（假）。
- 如果变量具有值 0（假），它将被设置为 1（真）。

在函数列表中使用

对位取反（变量）

在用户自定义函数中使用

InvertBit Tag

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

要设置其位的变量。

示例

以下程序代码会将布尔型变量 bStatus 的值取反，并将所得结果与原始 bSaved 值一起输出。

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Get current value
bValue=myTag.Value

'Save current value
bSaved=bValue

'Invert Bit
InvertBit myTag
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BOOL bStatus = 0;
BOOL bSaved = bStatus;

//Invert variable
invertBit(bStatus);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
  ...
}
```

1.2.10 InvertBitInTag

描述

对给定变量中的位取反：

- 如果变量中的位为值 1（真），它将被设置为 0（假）。
- 如果变量中的位为值 0（假），它将被设置为 1（真）。

在改变了给定位之后，系统函数将整个变量传送回 PLC。但是并不检查变量中的其它位是否同时改变。在所示变量被传回 PLC 之前，操作员和 PLC 对该变量只有只读权限。

说明

如果 PLC 支持 BOOL 型变量，不要使用该系统函数。而是使用“对位取反”系统函数。

在函数列表中使用

对变量中的位取反（变量，位）

在用户自定义函数中使用

InvertBitInTag Tag, Bit

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

要设置其给定位的变量。

位

要设置的位的编号。

在用户自定义函数中使用此系统函数时，变量中的位从右向左计数。计数从 0 开始。

示例

以下程序代码会将 bStatusWord 变量中指定的 bitposition 位取反，并将所得结果与原始 bSaved 值一起输出。

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Get current value
bValue=myTag.Value

'Save current value
bSaved=bValue

'Invert Bit in position
bitposition=2
InvertBit myTag, bitposition
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BYTE bStatusWord;
BYTE bsaved = bStatusWord;
BYTE bitposition = 2;

//Invert bit in bitposition
InvertBitInTag (bStatusWord, bitposition);
//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatusWord, bsaved);
  ...
}
```

1.2.11 LinearScaling

描述

为变量 Y 赋值，该变量通过线性函数 Y= (a *X) + b 利用给定变量 X 的值计算得出。

与此功能相反的系统函数是“转换线性转换”。

说明

变量 X 和 Y 不能相同。如果要将变量转换为其自身，必须使用一个辅助变量。
可用系统函数“设置变量”将要转换变量的值分配给辅助变量。

在函数列表中使用

线性转换 (Y, a, X, b)

在用户自定义函数中使用

LinearScaling Y, a, X, b

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

Y

要为其分配通过线性方程式计算得出的值的变量。

a

作为乘数的数值。

X

包含用于计算的值的变量。

b

作为加数的数值。

示例

下面的程序代码使用 LinearScaling 函数为 Yvar 变量赋值。

```
{
BYTE Yvar;
BYTE Xvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

// linear scaling
 LinearScaling ( Yvar, avalue, Xvalue, bvalue);

printf ("Yvar = %d\r\n, Yvar);
...
}
```

保存的返回值可在后续代码中进行处理。

1.2.12 LookupText

说明

从文本列表中标识一个条目。结果取决于值和所选定的运行系统语言。结果保存到数据类型为“String”的变量中。

在函数列表中使用

查找文本（输出文本，索引，语言，文本列表）

在用户自定义函数中使用

LookupText Output_text, Index, Language, Text_list

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

输出文本

要写入结果的变量。

索引

定义列表条目值的变量。

语言

定义标识列表条目所使用的运行系统语言。

- 运行系统语言
  语言代号按照 VBScript 参考，例如“de-DE”为德语（德国）或“en-US”为英语（美国）。此选择取决于激活何种运行系统语言。
- 变量
  包含该语言的变量。输入代表国家/地区标识号的十进制值作为运行系统语言的选择，例如，1031 代表德语 - 标准，1033 代表英语 - 美国。详细的介绍可从 VBScript 基础“Locale identifier (LCID) diagram”中获得。
- 整型
  对应进行语言切换时运行系统语言顺序的数字。
  此选择取决于已激活的运行系统语言，例如，“0”代表首次启动运行系统时显示的语言。
  有关详细信息，请参见“运行系统中的语言”主题。

文本列表

定义文本列表。列表条目从文本列表中读取。

1.2.13 ResetBit

描述

将“Bool”型变量的值设置为 0（假）。

在函数列表中使用

复位（变量）

在用户自定义函数中使用

ResetBit Tag

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

设置为 0（假）的 BOOL 型变量。

示例

以下程序代码使用 ResetBit 函数将布尔型变量 bStatus 的值置 0，并将所得结果与原始 bSaved 值一起输出。

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Set value
bValue=1
myTag.Value=bValue

'Save current value
bSaved=bValue

'Reset Bit
ResetBit myTag
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BOOL bStatus = 1;
BOOL bSaved = bStatus;

//Reset bit
  ResetBit (bStatus);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
...
}
```

1.2.14 ResetBitInTag

描述

将指定变量中的一个位设置为 0（假）。

在改变了给定位之后，系统函数将整个变量传送回 PLC。但是并不检查变量中的其它位是否同时改变。在所示变量被传回 PLC 之前，操作员和 PLC 对该变量只有只读权限。

说明

如果 PLC 支持 BOOL 型变量，不要使用该系统函数。而使用系统函数“复位”。

在函数列表中使用

复位变量中的位（变量，位）

在用户自定义函数中使用

ResetBitInTag Tag, Bit

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

其中一个位要设置为 0（假）的变量。

位

要设置为 0（假）的位的编号。

在用户自定义函数中使用此系统函数时，无论使用何种 PLC，指定变量中的位都是从右向左计数。计数从 0 开始。

示例

以下程序代码会将 bStatusWord 变量中指定的 bitposition 位置 0，并将所得结果与原始 bSaved 值一起输出。

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Save current value
bValue=myTag.Value
bSaved=bValue

'Reset Bit
bitposition=2
ResetBitInTag myTag, bitposition
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{

BYTE bSaved;
BYTE bitposition = 2;

bSaved = GetTagByte("bStatusWord");

//Reset bit in bitposition
ResetBitInTag ("bStatusWord", bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",GetTagByte("bStatusWord"), bSaved);
  ...
}
```

1.2.15 SetBit

描述

将"Bool"型变量的值设置为 1（真）。

在函数列表中使用

置位（变量）

在用户自定义函数中使用

SetBit Tag

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

要将其值设置为 1（真）的 BOOL 型变量。

示例

以下程序代码使用 SetBit 函数将布尔型变量 bStatus 的值置 1，并将所得结果与原始 bSaved 值一起输出。

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Set value
bValue=0
myTag.Value=bValue

'Save current value
bSaved=bValue

'Set Bit
SetBit myTag
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BOOL bStatus = 0;
BOOL bSaved = bStatus;

//Set bit
  SetBit (bStatus);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
...
}
```

1.2.16 SetBitInTag

描述

将给定变量中的一个位设置为 1（真）。

在改变了给定位之后，系统函数将整个变量传送回 PLC。但是并不检查变量中的其它位是否同时改变。在所示变量被传回 PLC 之前，操作员和 PLC 对该变量只有只读权限。

说明

如果 PLC 支持 BOOL 型变量，不要使用该系统函数。而使用系统函数“置位”。

在函数列表中使用

置位变量中的位（变量，位）

在用户自定义函数中使用

SetBitInTag Tag, Bit

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

要将其中的一个位设置为 1（真）的变量。

位

要设置为 1（真）的位的编号。

在用户自定义函数中使用此系统函数时，无论使用何种 PLC，指定变量中的位都是从右向左计数。计数从 0 开始。

说明

要实现可靠的功能，必须保证与实际过程值一起使用的变量的更新。因此，应在 I/O 域中组态变量或将系统函数分配给画面对象（如按钮）。

如果为系统函数组态了短事件（如激活报警），则只能通过设置连续读取的变量访问实际过程值。

示例

以下程序代码会将 bStatusWord 变量中指定的 bitposition 位置 1，并将所得结果与原始 bSaved 值一起输出。

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Save current value
bValue=myTag.Value
bSaved=bValue

'Set Bit in tag
bitposition=1
SetBitInTag "bStatusWord", bitposition
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "& bSaved & "New Value: " & bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{

BYTE bSaved;
BYTE bitposition = 1;

bSaved = GetTagByte("bStatusWord");

//Reset bit in bitposition
SetBitInTag ("bStatusWord", bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",GetTagByte("bStatusWord"), bSaved);

}
```

1.2.17 设置语言

描述

切换 HMI 设备上的语言。所有组态的文本和系统事件以新设置的语言显示在 HMI 设备上。

在函数列表中使用

设置语言（语言）

在用户自定义函数中使用

SetLanguage Language

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

语言

确定在 HMI 设备上设置了哪种语言。有下列规范：

- -1 (hmiToggle) = 切换：切换到下一种语言。组态期间在“项目语言”编辑器中确定顺序。
- 在“运行系统设置”编辑器的“语言和字体”中定义的编号。切换到带有给定编号的语言。
- 在“运行系统设置”编辑器的“语言和字体”中定义的语言。
- 按照 VBScript5 参考的语言缩写：这样可切换到与指定语言代码相对应的语言，例如"de-DE"表示德语（德国）或"en-US"表示英语（美国）。
  在 VBScript 基本信息的主题“区域图标识号 (LCID) 图”下具有语言缩写总览。

1.2.18 SetPropertyByConstant

说明

指定对象属性值为字符串。

在函数列表中使用

按常量设置属性（画面名称，画面对象，属性名称，值）

在用户自定义函数中使用

SetPropertyByConstant Screen_name, Screen_object, Property_name, Value

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“AUTOHOTSPOT”。

如要更改画面的属性，则参数“对象”必须为空。为此，请使用以下语法，例如：

SetPropertyByConstant Screen_name, Property_name, Value

参数

画面名称

包含对象的画面的名称。

画面对象

更改其属性的对象名称。

属性名称

要更改的属性名称。

值

分配给属性的值。

示例

下面的程序代码将使用 SetPropertyByConstant 函数更改对象属性：在“Trends”画面中，“Control_1”对象的“ToolbarButtonClick”属性设置为值 26。

```
'Programming language: VBS
'Name of the picture: Trends
'Name of the f(t) trend view control: Control_1

SetPropertyByConstant "Trends", "Control_1", "ToolbarButtonClick", "26"

'User defined code
...
```

```
{
//Programming language: C
//Name of the picture: Trends
//Name of the f(t) trend view control: Control_1

SetPropertyByConstant ("Trends", "Control_1", "ToolbarButtonClick", "26");

// User defined code
...
}
```

示例：更改画面属性

下面的程序代码将使用 SetPropertyByConstant 函数更改画面属性：在“Trends”画面中，“Backcolor”属性设置为值 255。

```
'Programming language: VBS
'Name of the picture: Trends

SetPropertyByConstant "Trends", "Trends", "Backcolor", "255"

'User defined code
...
```

```
{
//Programming language: C
//Name of the picture: Trends

SetPropertyByConstant ("Trends", "Trends", "Backcolor", "255");

// User defined code
...
}
```

此外，还可以使用密码 ZERO 或空格字符串代替第二个参数（对象）。

1.2.19 SetPropertyByProperty

说明

通过其它的对象属性指定某一对象属性的值。

在函数列表中使用

SetPropertyByProperty（画面名称，对象，属性名称，目标画面名称，目标画面对象，目标属性名称）

在用户自定义函数中使用

SetPropertyByProperty Screen_name, Screen_object, Property_name, Source_screen_name, Source_screen_object, Source_property_name

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“AUTOHOTSPOT”。

如果想使用另一个画面属性来指定某个画面的属性，则参数“对象”和“目标对象”必须为空。为此，请使用以下语法，例如：

SetPropertyByProperty Screen_name, Property_name, Source_screen_name, Source_property_name

参数

画面名称

包含对象的画面的名称。

对象

属性值要传递给目标对象的对象名称。

属性名称

值要传递给目标对象的属性名称。

目标画面名称

包含目标对象的画面的名称。

目标画面对象

更改属性的目标对象的名称。

目标属性名称

要更改的属性名称。

示例

以下程序代码使用 SetPropertyByProperty 函数将原始画面“Trend_1”中对象 "Control_1" 的属性“ToolbarButtonClick”设置成目标画面“Trend_2”中的相应属性。

```
'Programming language: VBS
'Name of source picture: Trend_1
'Name of target picture: Trend_2
'Name of the f(t) trend view control: Control_1

SetPropertyByProperty "Trend_1", "Control_1", "ToolbarButtonClick", "Trend_2",
"Control_2", "ToolbarButtonClick"

'User defined code
...
```

```
{
//Programming language: C
//Name of source picture: Trend_1
//Name of target picture: Trend_2
//Name of the f(t) trend view control: Control_1

SetPropertyByProperty ("Trend_1", "Control_1", "ToolbarButtonClick", "Trend_2",
"Control_2", "ToolbarButtonClick");

// User defined code
...
}
```

1.2.20 SetPropertyByTag

说明

通过变量值指定对象属性的值。

在函数列表中使用

按变量设置属性（画面名称，画面对象，属性名称，变量名称）

在用户自定义函数中使用

SetPropertyByTag Screen_name, Screen_object, Property_name, Tag_name

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“AUTOHOTSPOT”。

如果要通过变量值指定画面的属性，则“对象”参数必须为空。为此，请使用以下语法，例如：

SetPropertyByTag Screen_name, Property_name, Tag_name

参数

画面名称

包含对象的画面的名称。

画面对象

要使用变量值设置其属性的对象的名称。

属性名称

使用变量值设置的属性的名称。

变量名称

包含属性值的变量名称。

示例

下面的程序代码将使用 SetPropertyByTag 函数更改对象属性：单击对象时，可传送对象名称和包含对象的画面。画面窗口中的 CaptionText 包含 HMI_value_1 变量的值。

```
'Programming language: VBS
SetPropertyByTag screenName, objectName, "CaptionText", "HMI_value_1"

'User defined code
...
```

```
{
//Programming language: C
SetPropertyByTag (screenName, objectName, "CaptionText", "HMI_value_1");

// User defined code
...
}
```

示例

下面的程序代码将使用 SetPropertyByTag 函数更改对象属性：在“Trends”画面中，“Control_1”对象的“ToolbarButtonClick”属性设置为值 26。

```
'Programming language: VBS
'Name of the picture: Trends
'Name of the f(t) trend view control: Control_1

SetPropertyByConstant "Trends", "Control_1", "ToolbarButtonClick", "26"

'User defined code
...
```

```
{
//Programming language: C
//Name of the picture: Trends
//Name of the f(t) trend view control: Control_1

SetPropertyByConstant ("Trends", "Control_1", "ToolbarButtonClick", "26");

// User defined code
...
}
```

1.2.21 SetPropertyByTagIndirect

说明

将间接寻址的变量值写入对象属性。作为参数传送的变量包含其值被读取的另一变量的名称。

在函数列表中使用

按间接变量设置属性（画面名称，画面对象，属性名称，变量名称）

在用户自定义函数中使用

SetPropertyByTagIndirect Screen_name, Screen_object, Property_name, Tag_name

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“AUTOHOTSPOT”。

如果要通过变量指定画面的属性，则“对象”参数必须为空。为此，请使用以下语法，例如：

SetPropertyByTagIndirect Screen_name, Property_name, Tag_name

参数

画面名称

包含对象的画面的名称。

画面对象

要更改其属性的对象的名称。

属性名称

要更改的属性的名称。

变量名称

包含其值被读取的其它变量名称的变量的名称。

示例

下面的程序代码将使用 SetPropertyByTagIndirect 函数更改对象属性：

```
'Programming language: VBS
SetPropertyByTagIndirect GetParentScreen(screenName), GetParentScreenWindow(screenName),
"ScreenName", "HMI_value_1"

'User defined code
...
```

```
{
//Programming language: C
SetPropertyByTagIndirect (GetParentScreen(screenName), GetParentScreenWindow(screenName),
"ScreenName", "HMI_value_1");

// User defined code
...
}
```

1.2.22 SetTag

描述

将新值赋给给定的变量。

说明

该系统函数可用于根据变量类型分配字符串和数字。

在函数列表中使用

设置变量（变量，值）

在用户自定义函数中使用

SetTag Tag, Value

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

为其分配给定值的变量。

值

为给定变量所赋的值。

说明

"SetTag"系统函数只能在建立连接后执行。

示例

下面的程序代码使用 SetTag 函数将 gs_tag_bit 变量的值设置为 TRUE，并将返回值保存到 ok 变量中。

```
{
BOOL ok;
BOOL bvalue;

//Set the tag to true
ok = SetTag("gs_tag_bit", TRUE);
//error handling
if(ok)
{
  // succeeded
  printf ( "Function has run through.\r\n" );
bvalue = GetTagBit("gs_tag_bit");
  printf ("Value of gs_tag_bit: %d\r\n", bvalue);
}
else
{
  // failed
  printf ( "Error - function failed." );
}
...
}
```

保存的返回值可在后续代码中进行处理。

1.2.23 SetTagByProperty

描述

通过对象属性的值指定变量值。更改还会记录在报警系统中。

在函数列表中使用

按属性设置间接变量（变量名称，画面名称，画面对象，属性名称，包含或不包含操作员事件）

在用户自定义函数中使用

SetTagByProperty Tag_name, Screen_name, Screen_object, Property_name, With_or_without_operator_event

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见 “AUTOHOTSPOT”。

参数

变量名称

值由对象属性指定的变量名称。

画面名称

包含对象的画面的名称。

画面对象

其属性提供变量值的对象名称。

属性名称

提供变量值的属性名称。

有或没有操作员事件

0 (hmiWithoutOperatorEvent) = 没有操作员事件

1 (hmiWithOperatorEvent) = 有操作员事件

示例

在组合框中单击时，下面的程序代码将返回所选文本的值。

```
{
char* rt_value;

SetTagByProperty (rt_value, screenName, objectName, "SelectedText",
hmiWithoutOperatorEvent);

...
}
```

1.2.24 SetTagByTagIndirect

描述

将间接寻址的变量值写入变量。作为参数传送的变量包含其值被读取的另一变量的名称。通过操作员输入报警记录报警系统的变化。

在函数列表中使用

按间接变量设置变量（此变量的名称，变量名称，带或不带操作员输入报警）

在用户自定义函数中使用

SetTagByTagIndirect Tag_name, Source_tag_name, With_or_without_operator_event

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“AUTOHOTSPOT”。

参数

此变量的名称

要设置其值的变量名称。

变量名称

字符串变量的名称，该变量包含提供变量值的变量名。

有或没有操作员事件

0 (hmiWithoutOperatorEvent) = 无操作员输入报警

1 (hmiWithOperatorEvent) = 有操作员输入报警

示例

以下程序代码会将变量 Tag4 的值写入变量 Tag1。

```
{
SetTag ("IndirectRead", "Tag4");
SetTagByTagIndirect ("Tag1", "IndirectRead", hmiWithoutOperatorEvent);

...
}
```

1.2.25 SetTagIndirect

描述

向间接寻址变量写入值。作为输出参数传送的变量包含其值通过函数进行更改的另一变量的名称。通过操作员输入报警记录报警系统的变化。

在函数列表中使用

设置间接变量（变量名称，值，包含或不包含操作员事件）

在用户自定义函数中使用

SetTagIndirect Tag_name, Value, With_or_without_operator_event

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见 “AUTOHOTSPOT”。

参数

变量名称

字符串变量的名称，该变量包含要更改其值的变量名。

值

要写入的值。

包含或不包含操作员事件

0 (hmiWithoutOperatorEvent) = 不包含操作员事件

1 (hmiWithOperatorEvent) = 包含操作员事件

示例

以下程序代码会将值 "value" 写入变量 Tag3。

```
{
int value;

SetTag ("IndirectWrite", "Tag3");
SetTagIndirect ("IndirectWrite", "value", hmiWithoutOperatorEvent);
...
}
```

1.2.26 SetTagIndirectByProperty

描述

将对象属性的值写入间接寻址的变量。作为输出参数传送的变量包含其值通过函数进行更改的另一变量的名称。通过操作员输入报警记录报警系统的变化。

在函数列表中使用

按属性设置间接变量（变量名称，画面名称，画面对象，属性名称，有或没有操作员事件）

在用户自定义函数中使用

SetTagIndirectByProperty Tag_name, Screen_name, Screen_object, Property_name, With_or_without_operator_event

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见 “AUTOHOTSPOT”。

参数

变量名称

字符串变量的名称，该变量包含要更改其值的变量名。

画面名称

包含对象的画面的名称。

画面对象

其属性可提供数值的对象名称。

属性名称

可提供数值的属性名称。

有或没有操作员事件

0 (hmiWithoutOperatorEvent) = 无输入报警

1 (hmiWithOperatorEvent) = 有操作员输入报警

示例

以下程序代码将对象属性“背景色”(Background color) 的值写入变量 Tag2。

```
{
SetTag ("IndirectWrite", "Tag2");
SetTagIndirectByProperty ("IndirectWrite", screenName, objectName, "BackColor",
hmiWithoutOperatorEvent);
...
}
```

1.2.27 SetTagIndirectByTagIndirect

说明

将间接寻址的变量值写入间接寻址的变量。作为输出参数传送的变量包含其值通过函数进行更改的另一变量的名称。作为参数传送的变量包含其值被读取的另一变量的名称。通过操作员输入报警记录报警系统的变化。

在函数列表中使用

按间接变量设置间接变量（变量名，变量名，带或不带操作员输入报警）

在用户自定义函数中使用

SetTagIndirectByTagIndirect Tag_name, Source_tag_name, With_or_without_operator_event

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见 “AUTOHOTSPOT”。

参数

变量名称

字符串变量的名称，该变量包含要更改其值的变量名。

变量名称

返回变量值的间接变量的名称。

有或没有操作员事件

0 (hmiWithoutOperatorEvent) = 没有操作员事件

1 (hmiWithOperatorEvent) = 有操作员事件

示例

以下程序代码会将变量 "Tag4" 的值写入变量 "Tag2"。

```
{
SetTag ("IndirectWrite", "Tag2");
SetTag ("IndirectRead", "Tag4");
SetTagIndirectByTagIndirect ("IndirectWrite", "IndirectRead");
...
}
```

1.2.28 SetTagWithOperatorEvent

说明

为变量指定值。更改还会记录在报警系统中。

在函数列表中使用

SetTagWithOperatorEvent（变量名称，值）

在用户自定义函数中使用

SetTagWithOperatorEvent Tag_name, Value

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见 “AUTOHOTSPOT”。

参数

变量名称

要设置其值的变量名称。

值

已写入变量的值。

示例

单击相应按钮时，下面的程序代码将 "value" 变量的值传送到 "result" 变量。

```
'Programming language: VBS
SetTagWithOperatorEvent result, value
...
```

```
//Programming language: C
{
SetTagWithOperatorEvent ("result", "value");
...
}
```

1.2.29 ShowBlockInTIAPortalFromAlarm

说明

该函数用于从报警视图切换到工程组态系统中所选项目受影响的程序代码。从报警视图切换到 TIA Portal 中的项目后，能够进一步分析程序代码（例如，搜索互锁中缺失的触点）。

针对画面对象（例如，按钮或报警视图中的工具栏按钮）组态该系统函数。单击相应的按钮时，系统将存储必要的上下文，并且在切换到工程组态系统后，会自动显示程序代码中的相应位置。使用“写保护”(Write protection) 参数指定切换到工程组态系统所需的权限。

如果已针对某个按钮组态了该系统函数，则 TIA Portal 会以“写保护”(write protection) 参数所指定的模式启动。如要为系统函数实现两种不同的安全访问级别，需为按钮组态两种不同的权限：

- 对于第一个按钮，通过将“写保护”(write protection) 参数设为 FALSE 组态无限制访问权限。
- 对于第二个按钮，通过将“写保护”(write protection) 参数设为 TRUE 组态为不允许更改数据。

在函数列表中使用

在 Tia Portal 中显示来自报警的块（写保护，离线模式，TIA Portal 项目路径，报警视图的画面名称，报警视图的名称，错误变量）

在用户自定义函数中使用

ShowBlockInTIAPortalFromAlarm ReadOnly, Offline_mode, TiaPortalProject_path, AlarmScreen_name, AlarmView_name, Error_tag

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“Runtime Professional 系统功能”。

参数

“写保护”(Write protection)

指定项目是否以只读模式打开。

TRUE（默认设置）：TIA Portal 以读取模式打开，用户无法进行任何更改。

FALSE：TIA Portal 以读写模式打开。

“离线模式”(Offline mode)

指定项目是否以离线模式打开。

TRUE：TIA Portal 不会在块打开后进入在线模式。

FALSE（默认设置）：在块打开后启动在线模式。

“TIA Portal 项目路径”(TIA Portal project path)

要从报警视图切换到的 TIA Portal 中的项目路径和文件名称。

“报警视图的画面名称”(Screen name of the alarm view)

包含报警视图的画面的名称。

报警视图名称

要从 TIA Portal 切换到的报警视图的对象名称。

“错误变量”(Error tag)

如果出错，则“错误变量”(error tag) 会提供相关错误信息。

1.2.30 ShowLogonDialog

描述

在 HMI 设备上打开用户可用于登录到 HMI 设备的对话框。

在函数列表中使用

显示登录对话框

在用户自定义函数中使用

ShowLogonDialog

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参见“AUTOHOTSPOT”。

参数

--

1.2.31 ShowPLCCodeViewFromAlarm

说明

从包含所选 ProDiag 报警或所选 GRAPH 报警的报警视图切换到包含相应程序段的 PLC 代码视图。包含 PLC 代码视图的画面可从根画面或者画面窗口中打开。发出报警的 PC 的名称中不允许使用逗号。

在函数列表中使用

显示来自报警的 PLC 代码视图（报警画面名称，报警视图名称，基本画面名称，画面窗口名称，PLC 代码视图的画面名称，PLC 代码视图的名称，错误变量）

在用户自定义函数中使用

ShowPLCCodeViewFromAlarm Screen_name, Alarmview_name, Screen_name, Screen_window_name, Screen_name, PLCCodeView_name, Error_tag

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“Runtime Professional 系统功能”。

参数

报警画面名称

包含报警视图的画面的名称。

报警视图名称

在其中传输所选报警的报警视图的对象名称。

基本画面名称

包含可显示新画面的画面窗口的画面名称。

画面窗口名称

包含可显示 PLC 代码视图的画面的画面窗口名称。

PLC 代码视图的画面名称

包含 PLC 代码视图的画面的名称。

PLC 代码视图的名称

PLC 代码视图的对象名称。

错误变量

如果出错，则“错误变量”(error tag) 会提供相关错误信息。

错误消息

可使用“错误变量”(error tag) 反馈以下错误：

KOPAPI_E_TPSF_ALGCTRL_NOTSELECTED	必须仅选择一个报警
KOPAPI_E_TPSF_ONLYONLINELISTALLOWED	仅从报警视图的报警列表中选择报警
KOPAPI_E_TPSF_ONALARMWRONGPRODUCERID	未选择 GRAPH/ProDiag 报警
KOPAPI_E_TPSF_ONALARMNOSD0JUMPFLAG	无法跳过此 GRAPH/ProDiag 报警。 对于以下监控报警，可从报警视图中的报警跳转至 PLC 代码视图： • 对于全局监控，仅限互锁监控（互锁） • 对于局部监控，则针对输入参数的所有基本监控

1.2.32 StopRuntime

描述

退出运行系统软件，从而退出运行在 HMI 设备上的项目。

在函数列表中使用

停止运行系统（模式）

在用户自定义函数中使用

StopRuntime Mode

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

模式

确定在退出运行系统后操作系统是否关闭。

0 (hmiStopRuntime) = 运行系统：操作系统不关闭

1 (hmiStopRuntimeAndOperatingSystem) = 运行系统和操作系统：操作系统关闭(对于 WinCE 不适用)

示例

下面的程序代码将关闭运行系统和操作系统。

```
{

//Stop runtime and shutdown
StopRuntime (hmiStopRuntimeAndOperationSystem);

}
```

保存的返回值可在后续代码中进行处理。

1.3 用于 Windows 的 VBScript

1.3.1 用于 Windows 的 VBScript

VBScript

若使用过 Visual Basic 或应用于应用程序的 Visual Basic，则应该熟悉 VBScript。 若不知道 Visual Basic 或正在了解该语言，则要立即开始学习 Visual Basic 编程语言。 Microsoft Press 的逐步操作指南为编程提供详细的指导。

通过 Microsoft 主页可以获取有关 VBScript 语言元素的基本知识：

http://msdn.microsoft.com/en-us/library/t0aew7h6.aspx

本地 ID (LCID)

在 Microsoft 主页上可以获取所有语言代码的概述：

http://msdn.microsoft.com/en-us/goglobal/bb964664

1.4 适用于 Windows CE 的 VBScript

1.4.1 适用于 Windows CE 的 VBScript

适用于 Windows CE 的 VBScript

用户也可以在装有 Windows CE 的设备上使用除了用于访问文件的控件元素之外的全部 VBScript 函数。

在装有 Windows CE 的设备上，可以使用"File"和"FileSystem"控件元素及 CreateObject 函数访问文件和文件系统。

通过 Microsoft 主页可以获取有关 VBScript 语言元素的基本知识：

http://msdn.microsoft.com/en-us/library/t0aew7h6.aspx (http://msdn.microsoft.com/en-us/library/t0aew7h6.aspx)

本地 ID (LCID)

在 Microsoft 主页上可以获取所有语言代码的概述：

http://msdn.microsoft.com/en-us/goglobal/bb964664 (http://msdn.microsoft.com/en-us/goglobal/bb964664)

1.4.2 CreateObject

函数

该函数用于生成对自动化对象的引用。

语法

`CreateObject` (对象)

参数

对象

包含待生成对象的 ProgID 的字符串。

返回值

输出一个对自动化对象的引用。

说明

CreateObject 函数用于在运行系统中生成不可见的 ActiveX 控件。CreateObject 函数不可用于生成图形对象，例如 TreeView 控件或 ListView 控件。使用 CreateObjekt 函数生成的对象不会对事件作出响应。要生成可对事件作出响应的对象，请使用 CreateObjectWithEvents 函数。下表列出了无事件的 ActiveX 控件对应的 ProgID：

Control	ProgID
Microsoft CE File control 6.0	.file
Microsoft CE FileSystem control 6.0	.filesystem
Microsoft CE ImageList control 6.0	CEimageList.imagelistctrl

示例

```
Dim f, fwModeAppend
Set f = CreateObject("FileCtl.File")
fwModeAppend=8
f.Open "\Storage Card\testfile.txt", fwModeAppend
f.Close
```

1.4.3 控件元素

用于文件访问的控件元素

在装有 Windows CE 的设备上，可以使用“File”和“FileSystem”控件元素来访问文件和文件系统。控件元素只能与适用于 Visual Basic 6.0 的 Windows CE 工具包搭配使用。

库

FILECTLdtl

DLL

Mscefile.dll

文件控件元素

文件控件元素支持以下属性：

- Attr
- EOF
- Loc
- LOF
- Seek

文件控件元素支持以下属性：

- Close (File)
- Get
- Input
- Input
- InputFields
- LineInputString
- LinePrint
- Open
- Put
- WriteFields

FileSystem 控件元素

FileSystem 控件元素支持以下方法：

- Dir
- FileCopy
- FileDateTime
- FileLen
- GetAttr
- Kill
- MkDir
- MoveFile

- RmDir
- SetAttr

1.4.4 属性

1.4.4.1 Attr

函数

该属性会输出一个数字，标识打开文件时使用的文件模式。

语法

File.Attr

参数

File

对"File"控件的引用。

返回值

下表中列出的值指示访问模式。如果返回值为 0，则关闭文件。

常量	值
None	0
fsModeInput	1
fsModeOutput	2
fsModeRandom	4
fsModeAppend	8
fsModeBinary	32

说明

"Attr"为只读属性。使用"File"控件的"Open"方法来指定文件模式。

1.4.4.2 EOF

函数

对于打开用于随机或顺序输入的文件，当到达文件末尾时，该属性会输出 True。

语法

File.EOF

参数

File

对"File"控件的引用。

说明

到达文件末尾后，使用 EOF 属性来避免因读取操作导致的错误。

在到达文件末尾之前，EOF 属性始终输出 False。对于使用 fsModeRandom 或 fsModeBinary 打开的文件，在最后执行的 Get 语句返回不完整的数据记录之前，始终输出 False。

对于在 fsModeBinary 模式下打开的文件，系统会尝试使用输入函数读取文件，直到 EOF 输出 True 并生成一个错误。如果要通过输入读取二进制数据，请使用 LOF 和 LOC 属性（而非 EOF），或者结合使用 Get 和 EOF。对于使用 fsModeOutput 打开的文件，EOF 始终输出 True。

1.4.4.3 Loc

函数

该属性输出一个指定当前读取/写入位置的数字。

语法

File.Loc

参数

File

对"File"控件的引用。

说明

对于使用 fsModeRandom 打开的文件，Loc 将以数字形式输出最后被读取或写入的条目。对于使用所有其它方法打开的文件，Loc 将输出最后读取或写入的字节的位置。

1.4.4.4 LOF

函数

该属性以字节数的形式输出文件大小。

语法

File.LOF

参数

file

对"File"控件的引用。

说明

可将 LOF 属性与 Loc 属性搭配使用，以确保读操作不会超出文件范围。

1.4.4.5 Seek

函数

该属性设置将在文件中读取或写入的下一个位置并将其输出。

语法

File.Seek [= Position]

参数

file

对"File"控件的引用。

位置

描述文件中位置的数字地址。

说明

Seek 属性写入的是下一个文件位置，而 Loc 属性描述当前文件位置。Seek 输出的数字通常比 Loc 输出的数字大 1。唯一的例外情况是新打开文件时，此时 Seek 和 Loc 均输出 1。

如果 Seek 生成一个负值或零，则会输出一条错误消息。

1.4.5 方法

1.4.5.1 Close

函数

该方法会关闭打开的"File"控件。

语法

File.Close

参数

File

对"File"控件的引用。

输出值

无。

说明

使用 Open 方法打开文件。

1.4.5.2 Dir

函数

该方法输出一个采用特定模式或包含特定属性的文件名或目录。

语法

FileSystem.Dir (Pathname,[Attributs])

参数

File

对"FileSystem"控件的引用。

Pathname

可选：描述文件名或路径的字符串表达式。

Attributs

可选：总和描述文件属性的数字表达式。如果忽略此参数，则将输出相应路径下的所有文件。

下表列出了属性的参数设置。

常量	值	说明
fsAttrNormal	0	常规
fsAttrReadOnly	1	只读
fsAttrHidden	2	隐藏
fsAttrSystem	4	系统文件
fsAttrVolume	8	卷标。如果定义了此项，则将忽略所有其它属性。
fsAttrDirectory	16	目录
fsAttrArchive	32	日志

输出值

字符串。由路径名称和属性组成的文件名。如未找到路径名称，Dir 将输出一个长度为零的字符串 ("")。

说明

Dir 支持使用多字符通配符 (*) 和单字符通配符 (?) 来描述多次出现的文件。首次使用 Dir 方法时，必须为其指定路径名称。如果指定属性，则其中必须包含路径名称。

Dir 方法输出与路径名称匹配的第一个文件名。如果要输出与路径名称匹配的其它文件名，则需再次执行 Dir（无参数）。如果未能找到更多匹配的文件名，Dir 将返回一个长度为零的字符串 ("")。如果输出一个长度为零的字符串，则通过后续调用来指定路径名称。

1.4.5.3 FileCopy

函数

此方法将一个文件的内容复制到另一个文件中。

语法

FileSystem.FileCopy PathName, NewPathName

参数

FileSystem

对“FileSystem”控件的引用。

PathName

包含路径和文件名的字符串。

NewPathName

包含文件名和新文件路径的字符串。

输出值

无。

说明

如果不存在新文件，FileCopy 将输出一个错误消息。

1.4.5.4 FileDateTime

函数

该方法输出一个包含文件创建或最后编辑日期和时间的变量（日期）。

语法

FileSystem.FileDateTime(Pathname)

参数

Filesystem

对"FileSystem"控件的引用。

Pathname

指定文件名的字符串表达式。路径名称可能包含目录。

输出值

表示文件的最后编辑日期。

说明

如果不存在新文件，FileCopy 输出一条错误消息。

1.4.5.5 FileLen

函数

该方法输出以字节描述文件长度的值。

语法

FileSystem.FileLen(Pathname)

参数

Filesystem

对"FileSystem"控件的引用。

Pathname

描述文件名的字符串。路径名称可能包含目录。

输出值

指示文件长度的字节数。

说明

如果调用 FileLen 时指定的文件已打开，输出值将指示文件打开之前的大小。

1.4.5.6 Get

函数

该方法可将打开文件中的数据读入变量。

语法

File.Get Data, [Recnumber]

参数

File

对"File"控件的引用。

Data

读取数据的变体变量。

Recnumber

可选：开始读取时的数字。对于在二进制模式下打开的文件，Recnumber 定义字节位置。

输出值

无。

说明

通过 Get 方法读取的数据通常使用 Put 方法写入文件中。文件中的第一个数据记录或第一个字节位于位置 1，第二个数据记录或第二个字节位于位置 2，依此类推。如果略过 Recnumber，那么会读取最后一个 Get 或 Put 方法后（或 Seek 方法所引用）的下一个数据记录或下一个字节。

以下规则适用于在随机模式下打开的文件：

- 如果读取的数据长度小于 Open 方法的长度句段中所定义的长度，那么 Get 会以数据记录限制的长度读取以下数据记录。一个数据记录的终点和下一个数据记录的起点之间用文件缓冲区的内容填充。由于无法精确确定填充数据的范围，建议使数据记录的长度与要读取的数据的长度保持一致。
- 如果数据为数字型变体，Get 会读取 2 个字节来确定变体的 VarType，然后读取写入变量中的数据。例如，如果一个类型为 VarType3 的变体通过 Get 读取 6 个字节，其中 2 个字节用于将变体标识为 VarType 3（长整型），4 个字节包含长整型数据。长度句段中定义了数据记录长度。使用该打开方法时，数据记录长度必须比用于保存变量的字节数至少多 2 个字节。
- Get 方法可用于从内存读取变体数组。不过，它不能用于读取含数组的标量变体。使用 Get 无法从内存读取任何对象。
- 如果要读取的变体为 VarType 8（字符串），Get 将读取 2 个字节并将变体识别为 VarType 8，另有 2 个字节定义字符串长度，然后再读取字符串的数据。长度句段定义的 Open 方法的数据记录长度必须比字符串长度至少多 4 个字节。
- 如果要写入的变量为动态数组，Get 将读取描述符，其长度为 2 + 8 * 数组维度 (2 + 8 * NumberOfDimensions)。长度句段所定义的 Open 方法的数据记录长度必须大于或等于用于读取数组数据和数组描述符的总字节数。

对于在二进制模式下打开的文件，Open 方法的长度句段没有任何作用。Get 方法将从内存一次性读取所有变量，文件之间不做任何填充。

1.4.5.7 GetAttr

函数

该方法输出一个描述文件或文件夹属性的值。

语法

FileSystem.GetAttr(Pfadname)

参数

FileSystem

对“FileSystem”控件的引用。

Pathname

描述文件名称或文件夹名称的字符串。路径名称可能包含目录。

输出值

总和描述文件或文件夹属性的数字表达式。下表显示了可能的值。

常量	值	说明
vbNormal	0	常规
vbReadOnly	1	只读
vbHidden	2	隐藏
VbSystem	4	系统
VbDirectory	16	目录
VbArchive	32	文件自上次备份后已发生更改。

说明

要指定设置的属性，需使用 And 运算符对 GetAttr 返回的值和所选属性的值进行逐位比较。

1.4.5.8 Input

函数

该方法输出的字符串中包含在输入或二进制模式下打开的文件中的字符。

语法

File.Input(Number)

参数

File

对"File"控件的引用。

Number

用于定义输出字符数的数字表达式。

输出值

包含从文件中读取的字符的字符串。

说明

通过 Input 方法读取的数据通常使用 LinePrint 或 Put 方法写入文件中。此方法仅用于在输入或二进制模式下打开的文件。

与 LineInputString 方法不同的是，Input 方法会输出所有读取的字符，例如逗号、回车符、换行符、引号和前导空格。

在 EOF 函数返回"True"前，任何一次尝试用 Input 方法读取文件的操作都会导致已打开用于二进制访问的文件出错。为避免出现此类错误，请使用 LOF 和 LOC 函数代替 EOF 函数来通过 Input 方法读取二进制文件，或者结合使用 Get 和 EOF 函数。

1.4.5.9 InputB

函数

该方法返回在输入或二进制模式下打开的文件中的字节。

语法

File.InputB(Number)

参数

File

对"File"控件的引用。

Number

每个描述输出字节数的有效数字表达式。

输出值

包含从文件中读取的字节的数组。

说明

用 InputIB 方法读取的文件通常用 LinePrint 或 Put 方法写入。此方法仅用于在输入或二进制模式下打开的文件。

1.4.5.10 InputFields

函数

该方法可从打开的顺序文件中读取数据，并输出一个一维变体数据字段。

语法

File.InputFields(Number)

参数

File

对“FileSystem”控件的引用。

Number

从文件中读取并由逗号隔开的字段数。

输出值

包含从文件中读取的字段的数组。

说明

通过 InputFields 方法读取的数据通常使用 WriteFields 方法写入。该方法仅用于在二进制或输入模式下打开的文件。InputFields 会按原样读取标准字符串或数字数据。下表描述了 InputFields 如何读取其他输入数据：

数据	已分配变量的内容
逗号定界符或空命令行	空
#NULL#	Null
#TRUE# 或 #FALSE#	True 或 False
#yyyy-mm-dd hh:mm:ss#	以表达式形式显示的日期和/或时间

忽略双引号 ("") 及相关输入数据。

如果在添加数据对象时到达文件末尾，那么添加的数据将被删除并显示一个错误。

为了从文件正确读取数据作为使用 InputFields 的变量，请使用 WriteFields 方法代替 LinePrint 方法来将数据写入文件中。使用 WriteFields 可确保准确分隔各个数据字段。

1.4.5.11 Kill

函数

该方法可删除硬盘中的文件和文件夹。

语法

FileSystem.Kill Pathname

参数

FileSystem

对"FileSystem"控件的引用。

Pathname

用于命名要删除的一个或多个文件的字符串。路径名称可能包含一个文件夹。

输出值

无。

说明

Kill 方法支持使用多字符通配符 (*) 和单字符通配符 (?) 来标识多个文件。

如果尝试使用 Kill 方法删除已打开的文件，则会输出一个错误。

1.4.5.12 LineInputString

函数

该方法可从打开的顺序文件中读取单行内容，并将其与一个字符串变量关联。

语法

File.LineInputString

参数

File

对"File"控件的引用。

输出值

无。

说明

通过 LineInput 字符串读取的数据通常使用 Line Print 写入文件中。
LineInputString 方法可以逐个字符读取文件，直到到达回车序列 (Chr(13)) 或回车/换行 (Chr(13) + Chr(10))。回车和换行序列通常会被略过，而不是附加在字符串后面。

1.4.5.13 LinePrint

函数

该方法向打开的顺序文件中写入单行内容。

语法

File.LinePrint Output

参数

File

对"FileSystem"控件的引用。

Output

写入文件的字符串。

输出值

无

说明

使用 LinePrint 写入文件的数据通常用 LineInput 字符串读取。
字符串结尾追加回车序列 (Chr(13) + Chr(10))。

1.4.5.14 MkDir

函数

该方法会创建一个新文件夹。

语法

FileSystem.MkDir Pathname

参数

FileSystem

对"FileSystem"控件的引用。

Pathname

包含文件夹名称的字符串。

输出值

无。

说明

如果已存在目录，MkDir 将返回一条错误消息。

1.4.5.15 MoveFile

函数

该方法为已存在的文件、目录和所有子目录重新命名。

语法

FileSystem.MoveFile PathName, NewPathName

参数

FileSystem

对"FileSystem"控件的引用。

PathName

包含文件名的字符串。

NewPathName

包含要被复制的文件名的字符串。

输出值

无。

1.4.5.16 Open

函数

该方法可打开一个文件。共有下列几种文件模式：输入 (1)、输出 (2)、随机 (4)、附加 (8) 或二进制 (32)。

语法

File.Open Pathname, Mode, [Access], [Lock], [Reclength]

参数

File

对"FileSystem"控件的引用。

Pathname

包含文件名的字符串。

Mode

指定文件模式：输入 (1)、输出 (2)、随机 (4)、附加 (8) 或二进制 (32)。

Access

可选：不允许对打开的文件执行以下操作：读取、写入或读写 [默认]。(1, 2, 3)

Lock

可选：通过其他进程阻止对已打开文件执行的操作：Shared、LockRead、LockWrite [默认] 和 LockReadWrite。(1, 2, 3, 0)。

Reclength

可选：一个以字节为单位的数字，小于 32,767。对于通过随机访问打开的文件，该值对应于句段长度。对于顺序文件，该值为缓存的字符数。

输出值

无。

说明

reclength 参数在二进制模式下会被忽略。如果在随机模式下打开了某个文件，则必须定义一个大于零的文件大小；否则会输出一个错误。

1.4.5.17 Put

函数

该方法可将变量的数据写入文件。

语法

File.Put Data, [Recnumber]

参数

Data

包含要写入到文件中的数据的变体变量。

Recnumber

可选项。变体（长整型）。开始写操作进程时的数据记录编号（随机模式文件）或字节编号（二进制模式文件）。

输出值

无。

说明

通常，通过 Put 方法写入的数据将使用 Get 从文件中读出。

文件中的第一个数据记录或第一个字节位于位置 1，第二个数据记录或第二个字节位于位置 2，依此类推。如果略过 Recnumber，则会读取最后一个 Get 或 Put 方法后（或 Seek 方法所引用）的下一个数据记录或下一个字节。

以下规则适用于在随机模式下打开的文件：

- 如果要写入的数据长度小于 Open 方法的长度句段中所定义的长度，那么 Put 会以数据记录限制的长度读取以下数据记录。一个数据记录的终点和下一个数据记录的起点之间用文件缓冲区的内容填充。由于无法精确确定填充数据的长度，建议使数据记录的长度与要写入的数据的长度保持一致。如果要写入的数据长度大于 Open 方法的长度句段中所定义的长度，则会输出一个错误。
- 如果写入的变量为数字类型变体，则 Put 会首先写入 2 个字节来将变体声明为 VarType，然后再写入变量。例如，如果写入一个 VarType 3 变体，则 Put 会写入 6 个字节，其中 2 个字节用于将变体标识为 VarType 3（长整型），4 个字节包含长整型数据。长度句段所定义的 Open 方法的数据记录长度至少须比保存变量所需的字节数多 2 个字节。
- Put 方法可用于向内存写入变体数组。不过，它不能用于写入含数组的标量变体。不能使用 Put 将对象写入硬盘。

- 如果要写入的版本为 VarType 8（字符串），则 Put 将写入 2 个字节以将变体标识为 VarType 8，另有 2 个字节用于定义字符串长度，然后再写入字符串的数据。长度句段定义的 Open 方法的数据记录长度必须比字符串长度至少多 4 个字节。
- 如果要写入的变量为动态数组，Put 将写入描述符，其长度为 2 + 8 * 数组维度 (2 + 8 * NumberOfDimensions)。长度句段所定义的 Open 方法的数据记录长度必须大于或等于写入数组数据和数组描述符所需的总字节数。

对于在二进制模式下打开的文件，Open 方法的长度句段没有任何作用。Put 方法将向内存一次性写入所有变量，文件之间不做任何填充。

1.4.5.18 RmDir

函数

该方法会删除空目录。

语法

FileSystem.RmDir Pathname

参数

FileSystem

对“FileSystem”控件的引用。

PathName

包含文件名的字符串。

输出值

无。

说明

目录必须为空才能被删除。必须指定完整的路径。

1.4.5.19 SetAttr

函数

该方法设置文件的属性数据。

语法

FileSystem.SetAttr Pathname, Attributes

参数

FileSystem

对“FileSystem 控件”的引用

Pathname

包含文件名的字符串。

Attributes

包含文件属性总和的数字表达式。下表显示了可能的值。

常量	值	说明
vbNormal	0	标准（默认）
vbReadOnly	1	只读
vbHidden	2	隐藏
VbSystem	4	系统文件
VbArchive	32	文件自上次备份后已发生更改

输出值

无。

说明

如果您试图为打开的文件设置属性，将提示运行系统错误。

1.4.5.20 WriteFields

函数

该方法会向顺序文件中写入数据。

语法

File.WriteFields [Data]

参数

File

对"File"控件的引用。

Data

可选：要写入文件中的变体，或者数字串或字符串表达式的变体数组。

输出值

无。

说明

使用 WriteFields 写入的数据通常通过 InputFiles 从文件中读取。

如果忽略数据，将向文件中写入空白行。

- 句点将作为数字数据的十进制分隔符写入文件。
- 为 Boolean 数据输出 #TRUE# 或 #FALSE#。True 和 False 关键字均未编译，这与位置无关。
- 时间数据以通用日期格式写入文件。如果日期或时间缺失或为 null，则仅将现有信息写入文件。
- 如果数据为空，则不会向文件中写入任何内容。
- 如果数据为 null，则会向文件中写入 #NULL#。

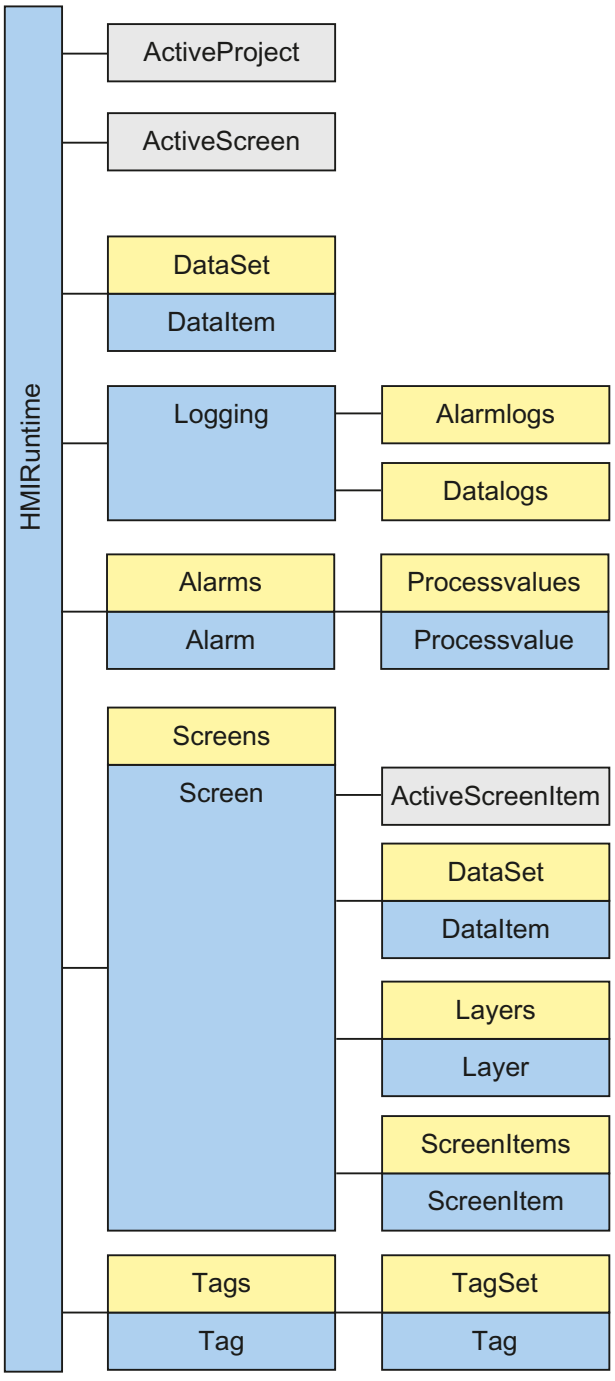
WriteField 方法为写入文件的字符串添加逗号和引号标记。列表中并非必须添加定界符。WriteFields 在向文件中写入最后一个数据字符后，将以回车/换行 (Chr(13) + Chr(10))- 形式插入一个换行符。

1.5 VBS 对象模型

1.5.1 VBS 对象模型

WinCC 的 VBS 对象模型

下图显示了 WinCC 中的 VBS 对象模型：



使用 WinCC 图形运行系统的对象模型访问运行系统的对象和变量。

对象

提供的对象和列表用于访问图形运行系统的所有对象：

- 显示和操作对象
- Screens
- Layers
- 变量

属性

通过各个对象的属性可以相应地更改运行系统的显示和操作对象以及变量。例如：通过单击操作按钮，或通过更改变量值触发颜色更改。

方法

应用到每个对象的方法可用于读取变量值，以便进一步处理或显示运行系统的报警。

更多信息

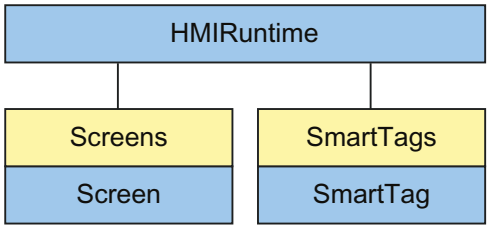
关于 VBS 对象模型的更多信息，请参见 SIMATIC 在线客户支持网站中的 FAQ 条目 ID “53752382”：

https://support.industry.siemens.com/cs/ww/en/view/53752382 (https://support.industry.siemens.com/cs/cn/zh/view/53752382)

1.5.2 对象

1.5.2.1 HMIRuntime

描述



描绘了图形运行系统。

HMIRuntime 对象包含将对象返回到主层的属性和方法。例如：返回画面对象的 ActiveScreen 属性。

应用

按如下操作，使用"HMIRuntime"对象：

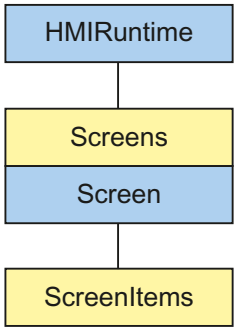
- 读取或设置当前的运行系统语言（"Language"属性）。
- 通过设置新画面名称（属性 "BaseScreenName"）读取当前基本画面的名称或触发基本画面更改
- 访问变量（列表"SmartTags"）
- 结束运行（方法 "Stop"）
- 输出有关顺序跟踪输出的信息（方法"Trace"）
- 访问运行期间显示的画面（列表 "Screens"）

参见

ActiveScreen (页 600)

1.5.2.2 Screen object（列表）

说明



screen 对象的列表。

该列表包含以下两种元素：

- 第一个元素，索引为“0”，表示永久区域。
- 第二个元素，索引为“1”，表示根画面。

或者，也可以按如下方式使用元素名称来寻址这两种元素：

- 永久区域：“Permanent area”
- 根画面：根画面中显示的画面的名称

如果未显示指定的画面，则在访问时会出错。

永久区域“Permanent area”在对象列表中显示，并且完全处于自动模式下。

说明

即使在运行时报警窗口和报警指示器具有焦点，但它们不包含在画面列表中。

应用

使用 screen 属性返回 screen 列表。在下列实例中，背景色由黑色变为绿色：

使用对象名作为索引。

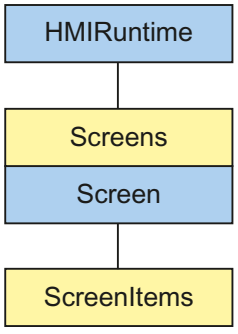
```
'VBS_Example_BackColor
HMIRuntime.Screens("Rootscreen").BackColor = vbGreen
```

说明

若进行画面更改，则对不再可用的画面的公开引用将变得无效。因此，不能再使用这些参考。

1.5.2.3 Screen

描述



表示当前 HMI 设备上正在显示的过程画面或运行系统的永久区域中的过程画面。screen 对象会作为访问 screen 列表的结果返回。

画面对象还包含画面中所有图形对象的列表，可通过“ScreenItems”列表对这些图形对象进行寻址。

应用

还可以使用 screen 对象进行以下操作：

- 读取画面的宽度和高度（属性 "Height" 和 "Width"）。
- 改变背景色（属性"BackColor"）。

使用对象名作为索引。

在下列实例中，背景色由黑色变为绿色：

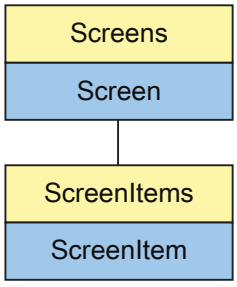
```
'VBS_Example_BackColor

HMIRuntime.Screens("Rootscreen").BackColor = vbGreen
```

参见

ScreenItem (页 206)

1.5.2.4 ScreenItem

描述



在指定画面中显示对象。 ScreenItem 对 象是 ScreenItems 列表的元素。

应用

可以使用 ScreenItem 对象，根据具体的事件访问画面中图形对象的属性。

按如下操作，使用“ScreenItem”对象。例如：

- ""“Visible”属性
  打开或关闭对象的可见性。
- ""“Height”和“Width”属性
  查询对象的宽度和高度。
- ""“Top”和“Left”属性
  更改对象的位置。
- ""“ObjectName”属性
  读取图形对象的名称
- ""“Parent”属性
  设置对父画面的引用

使用 ScreenItems 属性将对象返回到画面。 使用对象名作为索引。

示例

在下面的示例中，“RootScreen”画面中“myCircle”圆中的背景色设置为绿色。

```
'VBS_Example_ScreenItems

Dim objCircle
Set objCircle = HMIRuntime.Screens("RootScreen").ScreenItems("myCircle")
objCircle.BackColor = vbGreen
```

说明

为了节约 HMI 设备中的内存，在传送项目期间将不加载任何对象名。 如果仍要传送对象名称，请在 WinCC 中调用相应 HMI 设备的运行系统设置。 可以在 “常规”(General) 下更改此设置。 需通过对象名访问对象或要调试项目访问对象时，需要对象名。

"ScreenItem" 对象根据其特征具有不同的属性。 每个"ScreenItem" 对象都具有以下属性：

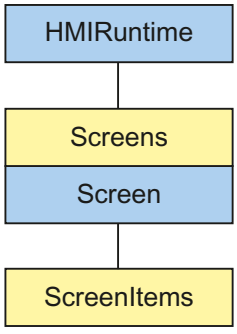
- Enabled
- Height
- Left

- ObjectName
- Parent
- Top
- Type
- Visible
- Width

如果访问特定的对象类型，则有更多属性添加到标准属性中。 在各个对象类型的描述中提供了附加属性。

1.5.2.5 ScreenItems

描述



包含给定过程画面中所有画面对象的 screen item 对象列表。 此列表具有一个父属性。 此属性提供了对画面对象所在过程画面的参考。

应用

通过"ScreenItems"列表，可以

- 编辑或输出列表中的所有对象(即画面中的所有对象)
- 对画面中的对象进行计数（属性 "Count"）。
- 使用列表中的特定对象（方法 "Item"）。

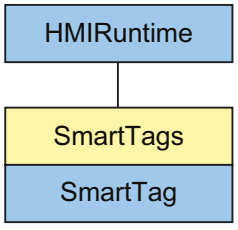
使用 screen items 属性从过程画面返回一个对象。使用对象名作为索引。

在下面的示例中，“RootScreen”画面中“myCircle”圆中的背景色设置为绿色。

```
Dim objCircle
Set objCircle = HMIRuntime.Screens("RootScreen").ScreenItems("myCircle")
objCircle.BackColor = vbGreen
```

1.5.2.6 SmartTags

描述



表示 WinCC Runtime 中所有变量的 SmartTag 对象列表。

说明

SmartTags 列表中包含的函数范围有限。只能使用变量名来访问 SmartTag 对象。不支持通过索引或使用"For each"指令进行访问。

说明

使用 SmartTag 列表以便访问尚未在项目中创建的变量，将不返回任何值。不会对不存在的变量进行赋值：

```
Dim intVar
intVar = SmartTags("FillLevel")
```

如果尚未创建"FillLevel"变量，"intVar"将保持为空。

说明

用户数据类型元素的动态访问的当前系统行为

在运行系统中动态组成用户数据类型元素的变量名称时，在以下情况下**不会**识别组成的变量名称：

- 已创建用户数据类型的数据类型变量。
- 变量**未**使用，例如未在 I/O 域中使用。

示例：

已组态一个包含元素"RPM" (Int)、"开启"(On) (Bool) 和"关闭"(Off) (Bool) 的用户数据类型"Motor"。在项目中，已将变量 "Motor1" 组态为用户数据类型的实例。要仅在运行系统的一个 I/O 域中输出多个元素值。已经为元素名称的输入组态了一个额外的 I/O 域。输入的值保存在内部变量 strElementName 中。通过以下脚本在 I/O 域"IOFieldOutputValue"中输出元素值：

```
Dim strDynElementName, objIOFieldOutputValue
Set objIOFieldOutputValue =
objscreen.ScreenItems("IOFieldOutput")
'Get element name from tag value
strDynElementName = SmartTags("strElementName").Value
'Create tag name
objIOFieldOutputValue.ProcessValue =
SmartTags("Motor1."+strDynElementName).Value
```

帮助：

例如，在附加画面中为各个用户数据类型元素组态单独的 I/O 域。将"过程值"与相应的用户数据类型元素互连。

说明

动态化面板

使用以下语法来访问与面板接口连接的变量（“Properties”）：

```
Smarttags("Properties\int_input1_tag")
```

应用

使用 SmartTags 列表返回 SmartTag 对象。使用变量名引用 SmartTag 对象。

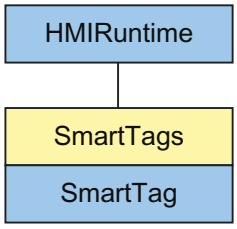
```
'VBS_Example_SmartTags
'将变量值写入本地变量，并通过用于调试报警的操作系统通道返回用户自定义文本。
Dim strAirPressure
strAirPressure = "Current air pressure: " + CStr(SmartTags("AirPressure"))
HMIRuntime.Trace strAirPressure
```

在 Runtime Advanced 和面板中，可以使用名称直接对变量进行寻址。若变量名符合 VBS 的命名约定时，则无需使用 SmartTag 列表。请参见以下示例：

```
Dim strAirPressure
strAirPressure = "Current air pressure: " + CStr(AirPressure)
HMIRuntime.Trace strAirPressure
```

1.5.2.7 SmartTag

说明



表示指定的过程变量的值。SmartTag 对象是 SmartTag 列表的元素。

应用

通过 smart tag 对象可以对指定的过程变量的值进行读写访问。SmartTag 对象不返回对象引用。使用 SmartTag 列表可返回过程变量的值。使用变量名作为索引。

说明

设置“SmartTag 读取缓存中的值”(SmartTags reads values from the cache) 后，将从过程映像（缓存）中读取值而不是直接从控制器读取。

SmartTag 对象也可以直接从控制器中读取值。不过，HMI 设备和控制器之间的通信负载将会大幅增加。

示例

```
'VBS_Example_SmartTags
'将变量值写入本地变量，并通过用于调试报警的操作系统通道返回用户自定义文本。
Dim strAirPressure
strAirPressure = "Current air pressure: " + CStr(SmartTags("AirPressure"))
HMIRuntime.Trace strAirPressure
```

说明

使用 SmartTag 列表以便访问尚未在项目中创建的变量，将不返回任何值。不会对不存在的变量进行赋值：

```
Dim intVar
intVar = SmartTags("FillLevel")
```

如果尚未创建"FillLevel"变量，"intVar"将保持为空。

说明

如果想要通过 VBS 函数"TypeName"返回 SmartTag 对象数据类型的"类型名称"(TypeName)，则使用下列语法：

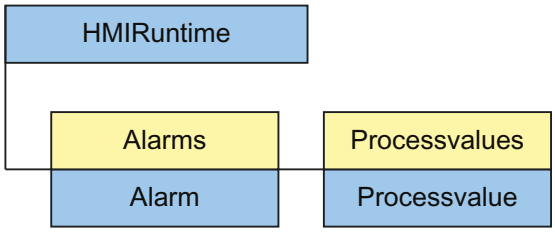
```
TypeName(SmartTags("FillLevel").value)
```

要访问数组元素的值，请使用"SmartTags（'<变量>[索引]'）"。为"索引"设置所需数组元素的数量，例如"SmartTags("AirPressure[2]")"。

1.5.3 对象

1.5.3.1 报警

描述



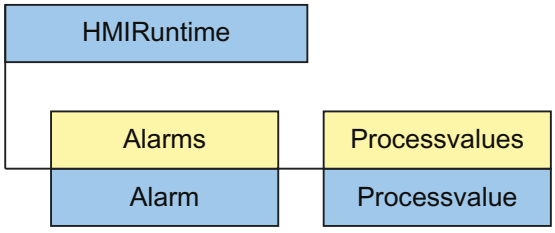
报警对象用于访问 Alarms 对象列表。

说明

报警对象的属性不会在属性值发生更改时自动更新。

1.5.3.2 Alarms（列表）

说明



使用报警对象触发现有消息。

用法

通过"Alarms"列表可以：

- 访问列表中的消息（Item 方法）
- 创建新的报警对象（Create 方法）
- 读取消息的报警 ID（AlarmID 属性）
- 生成报警对象的实例（Instance 属性）
- 读取发出消息的计算机的名称（ComputerName 属性）
- 读取或设置触发消息的用户的名称（UserName 属性）

示例

以下示例中，在“HMI 报警”编辑器中组态的报警编号为“1”的报警被激活。

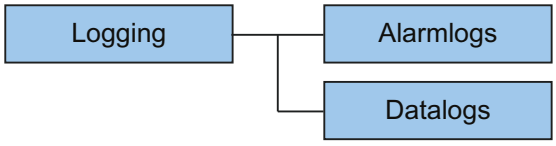
```
'VBS360
Dim MyAlarm
Set MyAlarm = HMIRuntime.Alarms(1)
MyAlarm.UserName = "Hans-Peter"
MyAlarm.Create "MyApplication"
```

参见

AlarmID (页 610)

1.5.3.3 AlarmLogs（列表）

说明



该对象可用于将报警日志的转出日志段重新与运行系统连接，或者删除报警日志中先前转入的日志段。 要转入的日志段复制到 WinCC 项目的“Common logging”文件夹中。 先前转入的日志段从“Common logging”文件夹中删除。

参数用于控制要转入的日志段的位置。要转入或删除的日志部分的时间段可以被指定。

若对日志段操作期间发生错误，则应用的方法返回错误报警。

使用

- “Restore”方法
  将先前转出的报警日志的日志段连接到运行系统。
- “Remove”方法
  将前转入的报警日志的日志段从运行系统的项目中删除。

示例

在下列示例中，对报警日志的日志段进行转入，并将返回值作为跟踪值输出。

```
'VBS187
HMIRuntime.Trace "Ret: " &
HMIRuntime.Logging.AlarmLogs.Restore("D:\Folder","2004-09-14","2004-09-20",-1) & vbNewLine
```

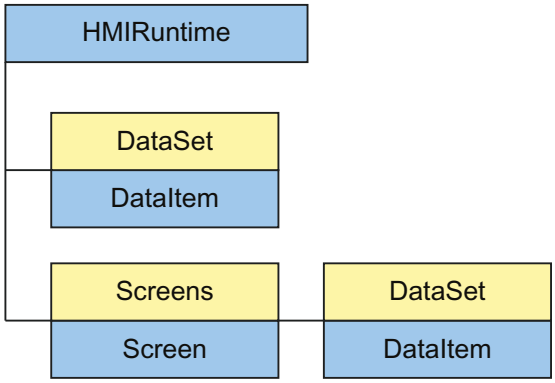
参见

Restore (页 1570)

Remove (页 1565)

1.5.3.4 DataItem

说明



DataItem 对象可用于访问 DataSet 列表的内容。 数值或对象引用作为 DataItem 存储在列表中。

访问使用其下的数值被添加到列表中的名称。 建议不采用使用索引的单次访问，因为添加或删除数值期间索引可以更改。 索引可用于输出列表的完整内容。 输出按字母顺序排序。

说明

对于对象引用，确保对象可以执行多线程操作。

示例

该示例介绍如何使“Motor1”数值作为跟踪值输出。

```
'VBS163
HMIRuntime.Trace "motor1: " & HMIRuntime.DataSet("motor1").Value & vbNewLine
```

下列示例列举了 DataSet 列表的所有 DataItem 对象。 名称和数值作为跟踪值输出。

```
'VBS164
Dim data
For Each data In HMIRuntime.DataSet
HMIRuntime.Trace data.Name & ": " & data.Value & vbNewLine
Next
```

说明

该对象值可能不会直接输出。

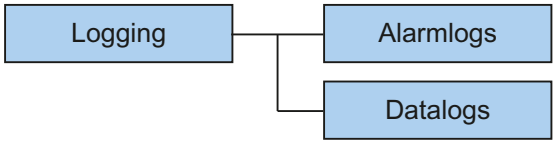
参见

值 (页 1376)

Name (页 985)

1.5.3.5 DataLogs（列表）

说明



该对象可用于将数据日志的转出日志段重新与运行系统连接，或者删除数据日志中先前转入的日志段。 要转入的日志段复制到 WinCC 项目的“Common logging”文件夹中。 先前转入的日志段从“Common logging”文件夹中删除。

参数用于控制要转入的日志段的位置。要转入或删除的日志部分的时间段可以被指定。 还可以设置日志的类型（“Fast data log”、“Slow data log”、“Fast data log 和 Slow data log”）。

若对日志段操作期间发生错误，则应用的方法返回错误报警。

使用

- “Restore”方法
  将先前转出的数据日志的日志段连接到运行系统。
- “Remove”方法
  将先前转入的数据日志的日志段从运行系统的项目中删除。

示例

在下列示例中，对 Fast 数据日志的日志段进行转入，并将返回值作为跟踪值输出。

```
'VBS188
HMIRuntime.Trace "Ret: " &
HMIRuntime.Logging.DataLogs.Restore("D:\Folder","2004-09-14","2004-09-20",-1,1) &
vbNewLine
```

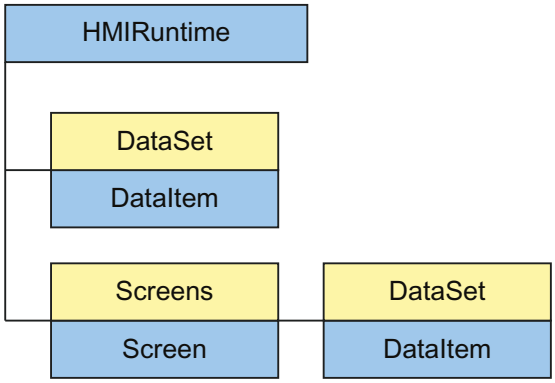
参见

Restore (页 1570)

Remove (页 1565)

1.5.3.6 DataSet（列表）

说明



使用 DataSet 对象，可以通过多次操作交换数据。

DataSet 对象为全局对象，通过画面对象定义。可通过任意 VBS 操作访问数据。

根据画面的层级对画面对象进行寻址。只要画面始终显示，则 DataSet 对象一直存在。全局对象在整个运行系统时间段内始终存在。

访问使用 DataItem 对象。

说明

DataSet 列表不能包含类型为 Screen、Screens、ScreenItem、ScreenItems、Tag 和 TagSet 的对象。

DataSet 对象不支持任何类。

应用

按如下步骤使用"DataSet"列表：

- 枚举
  输出或处理列表中的所有对象
- "Count"属性
  输出所包含的元素的数目
- "Item"方法
  操作列表中的特定对象
- "Add"方法
  向列表中添加对象
- "Remove"方法
  从列表中删除特定对象
- "RemoveAll"方法
  从列表中删除所有对象

按如下步骤操作可以访问列表元素：

```
HMIRuntime.DataSet("Itemname")
```

对于与画面相关的列表，可按如下操作进行访问：

```
HMIRuntime.Screens("Screenname").DataSet("Itemname")
```

按如下步骤访问画面的 DataSet 对象：

```
DataSet（"Itemname"）
```

若访问时所指定的名称在列表中不存在，则返回 VT_Empty 且触发异常事件。

示例

该示例显示了如何使用“Is Nothing”检查“DataSet”对象是否存在：

```
If HMIRuntime.DataSet("MyDataset") Is Nothing Then
    HMIRuntime.Trace "Requested DataSet-object does not exist"
Else
    HMIRuntime.Trace "Requested DataSet-object exsists"
End If
```

该示例显示如何通过相应的操作在列表中输入数值，以及从列表读取和删除数值。

```
'VBS162
HMIRuntime.DataSet.Add "motor1", 23
HMIRuntime.Trace "motor1: " & HMIRuntime.DataSet("motor1").Value & vbNewLine
HMIRuntime.DataSet.Remove("motor1")
```

参见

RemoveAll（页 1569）

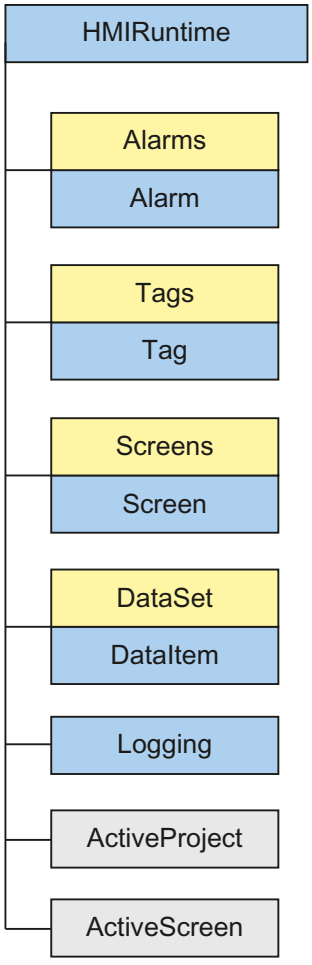
Remove (页 1565)

Item (页 1544)

Add (页 1475)

1.5.3.7 HMIRuntime

说明



HMIRuntime 对象代表图形运行系统环境。

使用

按如下操作，使用"HMIRuntime"对象：

- "Language"属性
  读取或设置当前的运行系统语言
- "BaseScreenName"属性
  读取或设置当前根画面的名称

- “ActiveProject”属性
  读取激活的运行系统项目的路径
- “Tags”属性
  访问变量
- “DataSet”属性
  访问列表变量
- “Stop”方法
  停止运行系统
- “Trace”方法
  显示诊断窗口中的消息

示例

下列命令可关闭 WinCC Runtime：

```
'VBS3
HMIRuntime.Stop
```

参见

Trace (页 1593)

Stop (页 1592)

Language (页 908)

变量 (页 1183)

Logging（页 947)

DataSet（页 774)

CurrentContext (页 768)

MenuToolBarConfig (页 955)

ActiveScreen (页 600)

BaseScreenName（页 659)

SmartTags (页 1130)

1.5.3.8 Item

说明

Item 对象用于引用当前对象。

使用

例如：通过 Item 对象可以对在当前画面中选择的对象属性进行寻址。

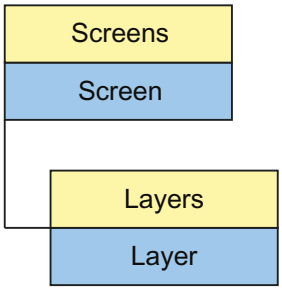
示例

在下面的示例中，可以将在画面中所选对象的背景色设置为红色：

```
'VBS195
Item.BackColor = RGB(255,0,0)
```

1.5.3.9 Layer

说明



Layer 对象将访问的结果返回给图层列表。

父对象

画面图层所在的画面

使用

根据某些事件，可以使用“图层”对象访问整个图层的属性，以根据操作员授权隐藏或显示具有操作元素的图层。

按如下操作，使用“Layer”对象：

- “Visible”属性
  激活或取消激活层的可见性
- “Name”属性
  读取层的名称

说明

“Layer”属性指定对象所在的层。 图层“0”输出为层“0”。
访问时，VBS 中的层从 1 开始计数。 因此，根据“Layers(2)”对层 1 进行寻址。

示例

在下例中，层 1 设置为“invisible”：

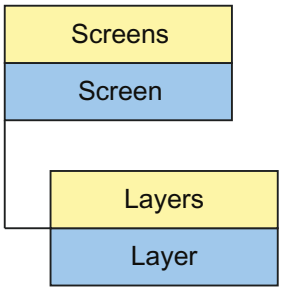
```
'VBS4
Layers(2).Visible = vbFalse
```

参见

Name (页 985)

1.5.3.10 Layers（列表）

说明



使用层列表可访问图形运行系统的所有 32 个层。

父对象

画面图层所在的画面

使用

按如下操作，使用"Layers"列表：

- "_NewEnum"属性
  处理列表中的所有层
- "Count"属性
  计数列表中包含的所有层
- "Item"方法
  处理列表中的层

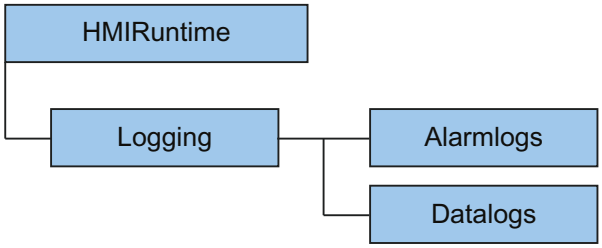
所有属性表示默认的属性和列表方法，在 WinCC 文档中未详细介绍。

参见

Item (页 1544)

1.5.3.11 Logging

说明



该对象可用于将转出日志段重新与运行系统连接，或者删除先前转入的日志段。 要转入的日志段复制到 WinCC 项目的“Common logging”文件夹中。 先前转入的日志段从“Common logging”文件夹中删除。

参数用于控制要转入的日志段的位置。要转入或删除的日志部分的时间段可以被指定。

若对日志段操作期间发生错误，则应用的方法返回错误报警。

使用

- “Restore”方法
  将先前转出的报警日志的日志段和数据日志连接到运行系统。
- “Remove”方法
  将先前转入的报警日志的日志段和数据日志从运行系统的项目中删除。

示例

在下列示例中，对报警日志的日志段和数据日志进行转入，并将返回值作为跟踪值输出。

```
'VBS189
HMIRuntime.Trace "Ret: " &
HMIRuntime.Logging.Restore("D:\Folder","2004-09-14","2004-09-20",-1) & vbNewLine
```

参见

Restore (页 1570)

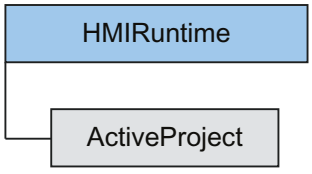
Remove (页 1565)

DataLogs（列表） (页 219)

AlarmLogs（列表） (页 216)

1.5.3.12 Project

说明



对象可用于从当前的运行系统项目查询信息。

项目对象作为 ActiveProject 的结果返回。

使用

可以通过“Project”对象读取下列内容：

- 当前运行系统项目的路径（“Path”属性）。
- 当前运行系统项目的名称，无路径或文件扩展名（“Name”属性）

示例

下列示例将当前运行系统项目的名称和路径作为跟踪值返回：

```
'VBS159
HMIRuntime.Trace "Name: " & HMIRuntime.ActiveProject.Name & vbNewLine
HMIRuntime.Trace "Path: " & HMIRuntime.ActiveProject.Path & vbNewLine
```

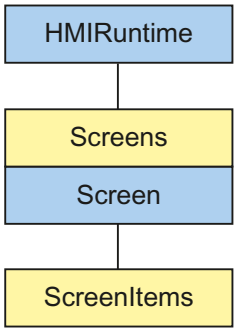
参见

Name (页 985)

Path (页 1022)

1.5.3.13 Screen

说明



表示当前 HMI 设备上正在显示的画面或运行系统的永久区域中的画面。画面对象会作为访问画面列表的结果返回。

画面对象还包含以下列表：

- 使用"ScreenItems"列表可以对寻址画面的所有图形对象进行寻址。
- 使用"Layers"对象可以对寻址画面的所有层进行寻址。

应用

画面对象可以进行下列操作。例如：

- "Width"和"Height"属性
  读取画面的宽度和高度
- "BackColor"属性
  更改背景色

使用对象名作为索引。

示例

在下列示例中，背景色由黑色变为绿色：

```
'VBS_Example_BackColor
HMIRuntime.Screens("Rootscreen").BackColor = vbGreen
```

父对象

嵌入画面对象的画面窗口。

若画面对象是根画面，则不会定义父对象，并将其设置为零。

说明

若进行画面更改，则对不再可用画面的公开引用将无效。因此，不能再使用这些引用。

示例

在下列示例中，运行系统中第一个画面的宽度增加 20 个像素：

```
'VBS7
Dim objScreen
Set objScreen = HMIRuntime.Screens(1)
MsgBox "Screen width before changing: " & objScreen.Width
objScreen.Width = objScreen.Width + 20
MsgBox "Screen width after changing: " & objScreen.Width
```

交叉引用说明

所有通过标准模式寻址的画面由 WinCC 中的 CrossReference 自动编译，并在画面属性中列出。

```
HMIRuntime.BaseScreenName = "Screenname"
```

若在代码中以不同格式调用画面，则可以通过以下 CrossReference 部分实现：

```
' WINCC:SCREENNAME_SECTION_START
Const ScreenNameInAction = "ScreenName"
' WINCC:SCREENNAME_SECTION_END
可以根据需要将该部分内容插入 VBS 脚本中。
```

参见

Refresh (页 1564)

Activate (页 1470)

ObjectSizeDeclutteringEnable (页 996)

ObjectSizeDeclutteringMax (页 997)

ObjectSizeDeclutteringMin (页 998)

LayerDeclutteringEnable (页 919)

Layers (页 920)

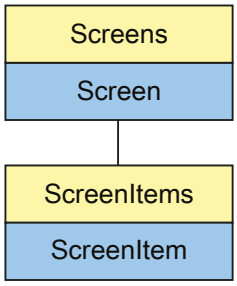
DataSet（页 774)

ExtendedZoomingEnable（页 818)

AccessPath (页 599)

1.5.3.14 ScreenItem

描述



在指定画面中显示对象。 ScreenItem 对 象是 ScreenItems 列表的元素。

应用

可以使用 ScreenItem 对象，根据具体的事件访问画面中图形对象的属性。

按如下操作，使用“ScreenItem”对象。例如：

- ""“Visible”属性
  打开或关闭对象的可见性。
- ""“Height”和“Width”属性
  查询对象的宽度和高度。
- ""“Top”和“Left”属性
  更改对象的位置。
- ""“ObjectName”属性
  读取图形对象的名称
- ""“Parent”属性
  设置对父画面的引用

使用 ScreenItems 属性将对象返回到画面。 使用对象名作为索引。

示例

在下面的示例中，“RootScreen”画面中“myCircle”圆中的背景色设置为绿色。

```
'VBS_Example_ScreenItems

Dim objCircle
Set objCircle = HMIRuntime.Screens("RootScreen").ScreenItems("myCircle")
objCircle.BackColor = vbGreen
```

说明

为了节约 HMI 设备中的内存，在传送项目期间将不加载任何对象名。 如果仍要传送对象名称，请在 WinCC 中调用相应 HMI 设备的运行系统设置。 可以在“常规”(General) 下更改此设置。 需通过对象名访问对象或要调试项目访问对象时，需要对象名。

"ScreenItem" 对象根据其特征具有不同的属性。 每个"ScreenItem" 对象都具有以下属性：

- Enabled
- Height
- Left
- ObjectName
- Parent
- Top
- Type
- Visible
- Width

如果访问特定的对象类型，则有更多属性添加到标准属性中。 在各个对象类型的描述中提供了附加属性。

参见

Activate (页 1470)

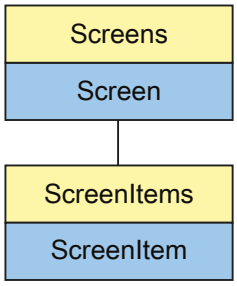
Layer (页 913)

ScreenItems (页 1069)

Top (页 1279)

1.5.3.15 ScreenItems（列表）

说明



包含指定画面的所有画面对象的 ScreenItem 对象列表。 此列表具有“父”属性。“父”属性提供对该画面对象所在画面的引用。

使用

按如下操作，使用"ScreenItems"列表：

- 编辑或输出列表中的所有对象（即画面中的所有对象）
- ""Count"属性
  计数画面的对象。
- ""Item"方法
  操作列表中的特定对象

使用 screen items 属性从画面返回一个对象。 使用对象名作为索引。

示例

在下面的示例中，“RootScreen”画面中“myCircle”圆中的背景色设置为绿色。

```
Dim objCircle
Set objCircle = HMIRuntime.Screens("RootScreen").ScreenItems("myCircle")
objCircle.BackColor = vbGreen
```

ScreenItem 对象的特性

若在 WinCC 中嵌入外部控件（ActiveX 控件或 OLE 对象），则嵌入控件属性的名称可能与 ScreenItem 对象常规属性的名称相同。 对于此情况，ScreenItem 属性具有较高的优先级。

然而，可以通过“对象”属性对嵌入控件的属性进行寻址。“对象”属性仅适用于 ActiveX 控件和 OLE 对象。

示例：

```
'Control1 是属性为“类型”的嵌入式 ActiveX 控件
'VBS196
Dim Control
Set Control=ScreenItems("Control1")
Control.object.type

'Control1 is a WinCC-Control
'VBS197
Dim Control
Set Control=ScreenItems("Control1")
Control.type
```

示例

在下列示例中，在当前画面的消息框中输出对象的名称：

```
Sub OnClick(ByVal Item)
'VBS6
Dim lngAnswer
Dim lngIndex
lngIndex = 1
For lngIndex = 1 To ScreenItems.Count
lngAnswer = MsgBox(ScreenItems(lngIndex).Objectname, vbOKCancel)
If vbCancel = lngAnswer Then Exit For
Next
End Sub
```

参见

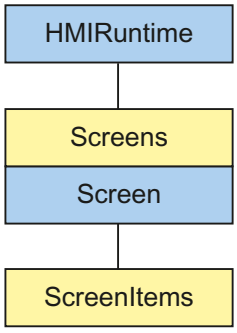
Item (页 1544)

ScreenItems (页 1069)

Top (页 1279)

1.5.3.16 Screen 对象(列表)

描述



在 WinCC Runtime 中可利用画面窗口技术同时打开多个画面，其中只有一个主画面。“Screens”列表允许使用画面名称访问运行系统中所有打开的画面。 画面列表包含所有隐藏画面。

在 VBS 环境中，HMIRuntime.Screens(<访问关键字>) 指令所需的访问关键字必须符合以下语法描述：

```
[<根画面名称>.]<画面窗口名称>[:<画面名称>] ...
.<画面窗口名称>[:<画面名称>]
```

- 访问关键字代表画面层级。
- 在关键字的任何位置都可以省略画面名称。

- “Screen”对象的“AccessPath”属性对应于完整的访问关键字。
- 可以通过 "" 访问关键字来寻址根画面。

示例

通过在列表中指定层级来寻址各个画面。 使用或不使用画面名称都可以寻址画面。 在以下示例中，使用“ScreenWindow”组态“BaseScreenName”根画面。 该画面窗口包含一个“ScreenName”画面。

使用画面名称寻址

```
'VBS8
Set objScreen = HMIRuntime.Screens("BaseScreenName.ScreenWindow:ScreenName")
```

不使用画面名称寻址

```
'VBS9
Set objScreen = HMIRuntime.Screens("ScreenWindow")
```

以各种方式引用根画面

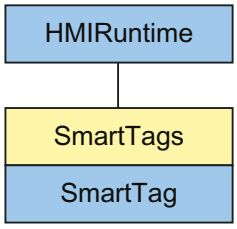
```
'VBS10
Set objScreen = HMIRuntime.Screens(1)

'VBS11
Set objScreen = HMIRuntime.Screens("")

'VBS12
Set objScreen = HMIRuntime.Screens("BaseScreenName")
```

参见

Screens (页 1070)

1.5.3.17 SmartTag

说明



表示指定的过程变量的值。SmartTag 对象是 SmartTag 列表的元素。

应用

通过 smart tag 对象可以对指定的过程变量的值进行读写访问。SmartTag 对象不返回对象引用。使用 SmartTag 列表可返回过程变量的值。使用变量名作为索引。

说明

设置“SmartTag 读取缓存中的值”(SmartTags reads values from the cache) 后，将从过程映像（缓存）中读取值而不是直接从控制器读取。

SmartTag 对象也可以直接从控制器中读取值。不过，HMI 设备和控制器之间的通信负载将会大幅增加。

示例

```
'VBS_Example_SmartTags
'将变量值写入本地变量，并通过用于调试报警的操作系统通道返回用户自定义文本。
Dim strAirPressure
strAirPressure = "Current air pressure: " + CStr(SmartTags("AirPressure"))
HMIRuntime.Trace strAirPressure
```

说明

使用 SmartTag 列表以便访问尚未在项目中创建的变量，将不返回任何值。不会对不存在的变量进行赋值：

```
Dim intVar
intVar = SmartTags("FillLevel")
```

如果尚未创建"FillLevel"变量，"intVar"将保持为空。

说明

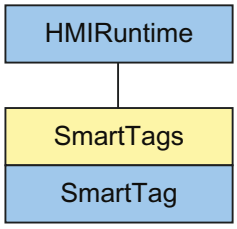
如果想要通过 VBS 函数"TypeName"返回 SmartTag 对象数据类型的"类型名称"(TypeName)，则使用下列语法：

```
TypeName(SmartTags("FillLevel").value)
```

要访问数组元素的值，请使用"SmartTags（'<变量>[索引]'）"。为"索引"设置所需数组元素的数量，例如"SmartTags("AirPressure[2]")"。

1.5.3.18 SmartTags

描述



表示 WinCC Runtime 中所有变量的 SmartTag 对象列表。

说明

SmartTags 列表中包含的函数范围有限。只能使用变量名来访问 SmartTag 对象。不支持通过索引或使用“For each”指令进行访问。

说明

使用 SmartTag 列表以便访问尚未在项目中创建的变量，将不返回任何值。不会对不存在的变量进行赋值：

```
Dim intVar
intVar = SmartTags("FillLevel")
```

如果尚未创建“FillLevel”变量，“intVar”将保持为空。

说明

用户数据类型元素的动态访问的当前系统行为

在运行系统中动态组成用户数据类型元素的变量名称时，在以下情况下**不会**识别组成的变量名称：

- 已创建用户数据类型的数据类型变量。
- 变量**未**使用，例如未在 I/O 域中使用。

示例：

已组态一个包含元素"RPM" (Int)、"开启"(On) (Bool) 和"关闭"(Off) (Bool) 的用户数据类型"Motor"。在项目中，已将变量 "Motor1" 组态为用户数据类型的实例。要仅在运行系统的一个 I/O 域中输出多个元素值。已经为元素名称的输入组态了一个额外的 I/O 域。输入的值保存在内部变量 strElementName 中。通过以下脚本在 I/O 域"IOFieldOutputValue"中输出元素值：

```
Dim strDynElementName, objIOFieldOutputValue
Set objIOFieldOutputValue =
objscreen.ScreenItems("IOFieldOutput")
'Get element name from tag value
strDynElementName = SmartTags("strElementName").Value
'Create tag name
objIOFieldOutputValue.ProcessValue =
SmartTags("Motor1."+strDynElementName).Value
```

帮助：

例如，在附加画面中为各个用户数据类型元素组态单独的 I/O 域。将"过程值"与相应的用户数据类型元素互连。

说明

动态化面板

使用以下语法来访问与面板接口连接的变量（"Properties"）：

```
Smarttags("Properties\int_input1_tag")
```

应用

使用 SmartTags 列表返回 SmartTag 对象。使用变量名引用 SmartTag 对象。

```
'VBS_Example_SmartTags
'将变量值写入本地变量，并通过用于调试报警的操作系统通道返回用户自定义文本。
Dim strAirPressure
strAirPressure = "Current air pressure: " + CStr(SmartTags("AirPressure"))
HMIRuntime.Trace strAirPressure
```

在 Runtime Advanced 和面板中，可以使用名称直接对变量进行寻址。若变量名符合 VBS 的命名约定时，则无需使用 SmartTag 列表。请参见以下示例：

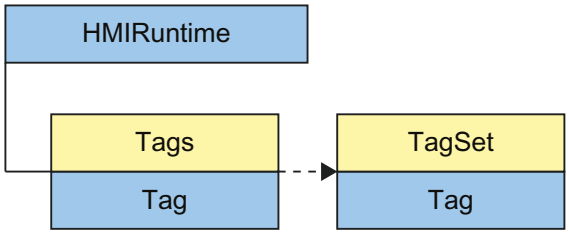
```
Dim strAirPressure
strAirPressure = "Current air pressure: " + CStr(AirPressure)
HMIRuntime.Trace strAirPressure
```

参见

SmartTag (页 240)

1.5.3.19 变量

说明



变量对象通过“Tags”列表返回。变量对象用于对变量的所有属性和方法进行寻址。

创建变量对象时，所有属性通过下列值初始化：

- Value = VT_EMPTY
- Name = Tag name
- QualityCode = BAD NON-SPECIFIC
- TimeStamp = 0

- LastError = 0
- ErrorDescription = " "

说明

有关 Quality Codes 的概要说明，请参见 WinCC 信息系统关键字“Communication”>“Diagnostics”或“Communication”>“Quality Codes”下的内容。

使用

按如下步骤使用“Tag”对象：

- “Name”、“QualityCode”、“TimeStamp”、“LastError”和“ErrorDescription”属性
  读取变量的信息
- “Write”方法、“Value”属性
  设置变量值
- “Read”方法、“Value”属性
  读取变量值

示例

下列示例显示的是读取“Tag1”变量的值：

```
'VBS13
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read()
MsgBox objTag.Value
```

WinCC 变量声明

始终使用“Dim”指令定义 VB 脚本的内部变量，以防止错误写入变量。

新建操作时，“Option explicit”指令在声明部分中自动输入并且无法删除。

在代码中切勿使用“Option explicit”语句，因为该语句会导致运行错误。

示例

以下示例显示了"lngVar"VB 脚本变量的声明：

```
'VBS14
Dim lngVar
lngVar = 5
MsgBox lngVar
```

说明

变量名不能包含任何特殊的字符。

创建变量时，确保其未包含有值 (Value = VT_EMPTY)。声明后通过相应的值对变量进行初始化。

CrossReference 和 ChangeObjectReference

所有通过标准格式进行寻址的变量

```
HMIRuntime.Tags("Tagname")
```

由 WinCC 的 CrossReference 自动编译，然后在画面属性中列出。

CrossReference 和 ChangeObjectReference 将与变量名搭配使用。但是，如果使用了"define"，"const"或字符串变量，则 CrossReference 和 ChangeObjectReference 将无法在工程系统中使用。脚本在运行系统中处于运行状态。

说明

WINCC:TAGNAME_SECTION 代码无效？？？必要时，必须在上一个项目移植期间重写。

说明

CrossReference 可能无法识别所组成的变量名。

参见

Name (页 985)

值 (页 1376)

ErrorDescription (页 804)

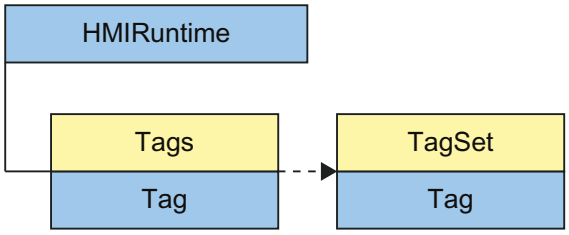
LastError (页 912)

QualityCode (页 1039)

TimeStamp (页 1233)

1.5.3.20 Tags（列表）

描述



“Tags”列表可用于访问 WinCC Runtime 变量。 访问“Tags”列表的结果由“Tag”类型对象返回。 变量对象可用于访问所有的变量属性和方法。

说明

“Tags”是一种具有功能限制范围的列表。 列表中的变量不能通过索引访问，只能使用变量名访问。 标准方法 get_Count 和 get_NewEnum 不能在变量列表中使用。

应用

可以通过以下方法访问列表变量：

```
HMIRuntime.Tags("Tagname")
```

变量列表用于声明读写访问的变量（变量对象）。 为了在执行读写访问期间不发生错误，必须存在适当的 HMI 变量。

可以在 VBScript 中直接对 HMI 变量进行寻址并设置和读取值。 若要查询有关其它变量属性，例如：质量代码或时间戳，或要执行错误处理，则必须通过变量列表对变量进行寻址。 返回的变量对象可以访问所有的变量属性和方法。

使用"CreateTagSet"方法可以生成 TagSet 对象，用于同时访问几个变量。

示例

设置变量时使用变量名称。

```
'VBS16
Dim objTag
Set objTag = HMIRuntime.Tags("Tagname")
如果只使用变量名，"TagPrefix"属性将被分配当前上下文（当前画面窗口）中的值。
```

1.5.3.21 TagSet（列表）

说明

对象"TagSet"可以在一次调用中同时访问多个变量。 同时访问与单次访问多个变量相比，具有更佳的性能和更少的通讯负荷。

使用

按如下操作，使用"TagSet"对象：

- "Add"方法
  向列表中添加变量
- "Item"方法
  访问列表中包含的变量对象和其属性
- "Write"方法
  写入列表的所有变量
- "Read"方法
  读取列表的所有变量

- “Remove”方法
  从列表中删除单个变量
- “RemoveAll”方法
  从列表中删除所有变量

可以通过以下方法访问列表变量：

```
'VBS169
Dim myTags
myTags = HMIRuntime.Tags.CreateTagSet
myTags("Tagname")
```

为了读写访问列表变量（变量对象）时无错误发生，WinCC 中必须存在相应的变量。

若发生读/写访问错误，则使用的方法将通过“LastError”和“ErrorDescription”属性返回错误消息。

示例

以下示例显示的是如何生成 TagSet 对象、如何添加变量以及如何写入值。

```
'VBS168
'Build a Reference to the TagSet Object
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
'Add Tags to the Collection
group.Add "Motor1"
group.Add "Motor2"
'Set the Values of the Tags
group("Motor1").Value = 3
group("Motor2").Value = 9
'Write the Values to the DataManager
group.Write
```

参见

ErrorDescription (页 804)

LastError (页 912)

1.5.4 对象类型

1.5.4.1 VBS 中可用的对象类型

以下各表的用途

以下各表显示了对象类型的 VBS 名称、其在“画面”(Screens) 编辑器中的显示名称以及其在相应运行系统中的可用性。

基本对象

VBS 名称	显示名称	脚本支持	适用对象		
			WinCC RT CE（面板）	WinCC RT Advanced	WinCC RT Professional
Circle	“圆”(Circle)	可用	可用	可用	可用
CircleSegment	“扇形”(Circle segment)	可用	-	-	可用
CircularArc	“圆弧”(Circular arc)	可用	-	-	可用
Connector	“连接器”(Connector)	可用	-	-	可用
Ellipse	“椭圆”(Ellipse)	可用	可用	可用	可用
EllipseSegment	“椭圆扇形”(Ellipse segment)	可用	-	-	可用
EllipticalArc	“椭圆弧”(Ellipse arc)	可用	-	-	可用
GraphicView	“图形视图”(Graphic view)	可用	可用	可用	可用
Line	“线”(Line)	可用	可用	可用	可用
Polygon	“多边形”(Polygon)	可用	可用	可用	可用
Polyline	“折线”(Polyline)	可用	可用	可用	可用
Rectangle	“矩形”(Rectangle)	可用	可用	可用	可用
TextField	“文本域”(Text field)	可用	可用	可用	可用
TubeArcObject	“管道弯头”(Pipe bends)	可用	-	-	可用
TubeDoubleTeeObject	“双 T 形管”(Double T-piece)	可用	-	-	可用

VBS 名称	显示名称	脚本支持	适用对象		
			WinCC RT CE（面板）	WinCC RT Advanced	WinCC RT Professional
TubePolyline	“管道”(Pipe)	可用	-	-	可用
TubeTeeObject	“T 形管”(T piece)	可用	-	-	可用

元素

VBS 名称	显示名称	脚本支持	适用对象		
			WinCC RT CE（面板）	WinCC RT Advanced	WinCC RT Professional
Bar	“棒图”(Bar)	可用	可用	可用	可用
BatteryView	“充电情况”(Charge condition)	可用	可用	-	-
Button	“按钮”(Button)	可用	可用	可用	可用
CheckBox	“复选框”(Check box)	可用	-	-	可用
Clock	“时钟”(Clock)	可用	可用	可用	可用
ComboBox	“组合框”(Combo box)	可用	-	-	可用
DateTimeField	“日期/时间域”(Date/ time field)	可用	可用	可用	-
DiskSpaceView	“磁盘空间视图”(Disk space view)	可用	-	-	可用
Gauge	“量表”(Gauge)	可用	可用	可用	可用
GraphicIOField	“图形 I/O 域”(Graphic I/O field)	可用	可用	可用	可用
HelpIndicator	“帮助指示器”(Help indicator)	不可用 [1]	可用	-	-
IOField	“I/O 域”(I/O field)	可用	可用	可用	可用
MultilineEdit	“可编辑的文本域”(Editable text field)	可用	-	-	可用
ListBox	“列表框”(List box)	可用	-	-	可用
OptionGroup	“选项按钮”(Option buttons)	可用	-	-	可用

VBS 名称	显示名称	脚本支持	适用对象		
			WinCC RT CE（面板）	WinCC RT Advanced	WinCC RT Professional
ProtectedAreaNameView	“有效范围名称(RFID)”(Effective range name (RFID))	可用	是 [2]	-	-
RangeLabelView	“有效范围名称”(Effective range name)	可用	是 [2]	-	-
RangeQualityView	“有效范围信号”(Effective range signal)	可用	是 [2]	-	-
RoundButton	“圆形按钮”(Round button)	可用	-	-	可用
Slider	“滑块”(Slider)	可用	可用	可用	可用
Switch	“开关”(Switch)	可用	可用	可用	-
SymbolicIOField	“符号 I/O 域”(Symbolic I/O field)	可用	可用	可用	可用
SymbolLibrary	“符号库”(Symbol library)	可用	可用	可用	可用
WindowSlider	“滚动条”(Scroll bar)	可用	-	-	可用
WLanQualityView	”WLAN 接收”(WLAN reception)	可用	是 [2]	-	-
ZoneLabelView	“区域名称”(Zone name)	可用	是 [2]	-	-
ZoneQualityView	“区域信号”(Zone signal)	可用	是 [2]	-	-

[1] 仅在全局画面中可组态

[2] 移动面板

控件

VBS 名称	显示名称	脚本支持	适用对象		
			WinCC RT CE（面板）	WinCC RT Advanced	WinCC RT Professional
AlarmControl	“报警视图”(Alarm view)	可用	-	-	可用
AlarmIndicator	“报警指示器”(Alarm indicator)	不可用 [1]	可用	可用	-
AlarmView	“报警视图”(Alarm view)	可用	可用	可用	-
AlarmWindow	“报警窗口”(Alarm window)	不可用 [1]	可用	可用	-
ApplicationWindow	“打印作业/脚本诊断”(Print job/Script diagnostics)	可用	-	-	可用
CameraControl	“摄像机视图”(Camera view)	可用	可用	-	-
ChannelDiagnose	“通道诊断”(Channel diagnostics)	可用	-	-	可用
FunctionTrendControl	“f(x) 趋势视图”(f(x) trend view)	可用	可用	可用	可用
HTMLBrowser	“HTML 浏览器”(HTML browser)	可用	可用	可用	可用
MediaPlayer	“媒体播放器”(Media Player)	可用	可用	-	可用
OnlineTableControl	“表格视图”(Table view)	可用	-	-	可用
OnlineTrendControl	“f(t) 趋势视图”(f(t) trend view)	可用	-	-	可用
PDFview	“PDF 视图”(PDF view)	可用	可用	可用	-
PLCCodeViewer	“PLC 代码查看器”(PLC code viewer)	可用	-	-	可用
RecipeView	“配方视图”(Recipe view)	可用	可用	可用	-

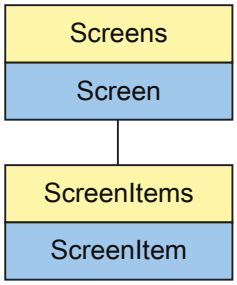
VBS 名称	显示名称	脚本支持	适用对象		
			WinCC RT CE（面板）	WinCC RT Advanced	WinCC RT Professional
S7GraphOverview	“S7-GRAPH 总览”(S7-GRAPH Overview)	可用	-	-	可用
ScreenWindow	画面窗口	可用	-	-	可用
SmartClientView	“Sm@rtClient 视图”(Sm@rtClient view)	可用	可用	可用	-
StatusForce	“监控表”(Watch table)	可用	可用	可用	-
SysDiagControl	“系统诊断视图”(System diagnostics view)	可用	可用	可用	可用
SysDiagWindow	“系统诊断视图”(System diagnostics view)	不可用 [1]	可用	可用	-
TrendRulerControl	“数值表”(Value table)	可用	-	-	可用
TrendView	“趋势视图”(Trend view)	可用	可用	可用	-
UserArchiveControl	“配方视图”(Recipe view)	可用	-	-	可用
UserView	“用户视图”(User view)	可用	可用	可用	可用

[1] 仅在全局画面中可组态

1.5.4.2 Objects A-I

AlarmControl

说明



表示“Alarm view”对象。AlarmControl 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIAlarmControl

示例

在下面的示例中，名称为“Control1”的对象向右移动了 10 个像素：

```
'VBS54
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left + 10
```

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-1　　属性

属性	RT Professional	RT Advanced	Panel RT	说明
AllFilters	-	-	-	-
AllFiltersForHitlist	-	-	-	-
AllServer (页 618)	RW	-	-	指定是否显示所有可用服务器中的报警。
AllTagTypesAllowed	-	-	-	-
ApplyProjectSettings (页 621)	RW	-	-	指定是否从"HMI 报警"(HMI alarms) 编辑器中应用项目设置。
ApplyProjectSettingsForDesignMode	-	-	-	-
AssignedFilters	-	-	-	-
AssignedHitlistFilters	-	-	-	-
AutoCompleteColumns (页 629)	RW	-	-	指定在控件比组态的列宽时是否显示空列。
AutoCompleteRows (页 629)	RW	-	-	指定在控件比组态的行数长时是否显示空行。
AutoScroll (页 631)	RW	-	-	指定发生新报警时报警窗口的响应方式。
AutoSelectionColors (页 632)	RW	-	-	指定是否将系统定义的颜色用作单元格和行的选择颜色。
AutoSelectionRectColor (页 632)	RW	-	-	指定是否使用系统定义的颜色显示选择框架。
BackColor (页 637)	RW	-	-	指定所选对象的背景色。
Blocks	-	-	-	-
BorderColor	-	-	-	-

属性	RT Professional	RT Advanced	Panel RT	说明
BorderWidth (页 697)	RW	-	-	指定所选对象的线宽。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Caption (页 708)	RW	-	-	指定显示在所选对象标题中的文本。
CellCut (页 713)	RW	-	-	指定单元格过窄时是否缩略单元格内容。
CellSpaceBottom (页 714)	RW	-	-	指定表格单元格的下边距。
CellSpaceLeft (页 715)	RW	-	-	指定表格单元格中采用的左缩进。
CellSpaceRight (页 716)	RW	-	-	指定表格单元格中采用的右缩进。
CellSpaceTop (页 717)	RW	-	-	指定表格单元格的上边距。
Closeable (页 722)	RW	-	-	指定是否可在运行系统中关闭对象。
ColumnResize (页 742)	RW	-	-	指定是否可以更改列宽。
Columns	-	-	-	-
ColumnScrollbar (页 743)	RW	-	-	启用列滚动条的显示。
ColumnTitleAlignment (页 754)	RW	-	-	指定列标题对齐的类型。
ColumnTitles (页 755)	RW	-	-	指定是否显示列标题。
ControlDesignMode (页 762)	RW	-	-	指定控件样式。
DefaultFilterEom	-	-	-	-
DefaultHitListFilterEom	-	-	-	-
DefaultMsgFilterSQL (页 775)	RW	-	-	指定用作报警过滤器默认值的 SQL 语句。
DefaultSort (页 776)	RW	-	-	指定排序类型。

属性	RT Professional	RT Advanced	Panel RT	说明
DefaultSort2 (页 777)	RW	-	-	指定排序类型。
DefaultSort2Column (页 777)	RW	-	-	指定默认情况下首先用作报警视图排序依据的列。
DiagnosticsContext (页 778)	RW	-	-	指定诊断环境。
DisplayOptions (页 785)	RW	-	-	指定是否显示视图被抑制的报警。
DoubleClickAction (页 786)	RW	-	-	指定双击消息行时将在运行系统中执行的动作。
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。
ExportDelimiter	-	-	-	-
ExportDirectoryChangeable (页 808)	RW	-	-	指定是否可以在运行系统中更改数据导出目录。
ExportDirectoryname (页 809)	RW	-	-	指定导出的运行系统数据的目标目录。
ExportFileExtension (页 810)	RW	-	-	指定导出文件的文件扩展名。目前仅支持“csv”文件扩展名。
ExportFilename (页 811)	RW	-	-	指定导出的运行系统数据的目标文件名称。
ExportFilenameChangeable (页 812)	RW	-	-	指定是否可以在运行系统中更改导出文件名。
ExportFormat	-	-	-	-
ExportFormatGuid (页 813)	RW	-	-	指定 ID 编号和导出数据源的分配。
ExportFormatName (页 814)	RW	-	-	指定导出文件格式。当前只能导出“csv”格式的文件。
ExportParameters (页 815)	RW	-	-	通过属性对话框指定所选格式的参数。
ExportSelection (页 816)	RW	-	-	指定导出控件的哪些运行系统数据。

属性	RT Professional	RT Advanced	Panel RT	说明
ExportShowDialog (页 817)	RW	-	-	指定是否在运行系统中显示数据导出对话框。
FillPattern	-	-	-	-
FillPatternColor	-	-	-	-
Filter	-	-	-	-
Font (页 846)	RW	-	-	指定字体。
GridLineColor (页 860)	RW	-	-	指定网格线颜色。
GridLineWidth (页 862)	RW	-	-	指定分隔线的宽度（以像素为单位）。
Height (页 863)	RW	-	-	指定所选对象的高度。
Hitlist	-	-	-	-
HitlistColumnAdd (页 871)	RW	-	-	将所选报警文本块从可用报警块列表传送到选定报警块列表。
HitlistColumnCount (页 872)	RW	-	-	指定要显示在运行系统统计列表中的报警文本块的数量。
HitlistColumnIndex (页 872)	RW	-	-	引用为统计列表选择的报警文本块。
HitlistColumnName (页 873)	RW	-	-	显示通过"HitlistColumnIndex"属性引用的统计列表中的报警文本块名称。
HitlistColumnRemove (页 874)	RW	-	-	从所选报警块列表中剪切标记的报警文本块，然后将其粘贴到现有报警块列表中。
HitlistColumnRepos (页 874)	RW	-	-	更改报警块的排序顺序。
HitlistColumnSort (页 875)	RW	-	-	指定统计列表中"HitlistColumnIndex"引用的报警文本块的排序顺序。
HitlistColumnSortIndex (页 876)	RW	-	-	定义统计列表中"HitlistColumnIndex"引用的报警文本块的排序顺序。
HitlistColumnVisible (页 876)	RW	-	-	指定统计列表中用于运行系统控件的选定报警文本块的列表。
HitlistDefaultSort (页 877)	RW	-	-	设置统计列表表格列的默认排序顺序。

属性	RT Professional	RT Advanced	Panel RT	说明
HitlistFilter	-	-	-	-
HitlistMaxSourceItems (页 878)	RW	-	-	设置报警日志中用于创建统计列表的最大数据记录数。
HitlistMaxSourceItemsWarn (页 878)	RW	-	-	指定在达到"HitlistMaxSourceItems"中所定义的最大数据记录数时是否发出警告。
HitListRelTime (页 879)	RW	-	-	设置统计的时间范围。
HitListRelTimeFactor (页 880)	RW	-	-	指定与时间类型"HitlistRelTimeFactorType"对应的时间系数，以确定用于创建统计列表统计数据的时间段。
HitlistRelTimeFactorType (页 880)	RW	-	-	指定时间因数"HitlistRelativeTimeFactor"旁边的时间类型，以确定创建统计列表各项统计的时间段。
HorizontalGridLines (页 883)	RW	-	-	指定是否显示水平分隔线。
IconSpace (页 886)	RW	-	-	指定表格单元格中图标和文本的间距。
IsActive	-	-	-	-
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定对象的 X 坐标的值。
LineBackgroundColor	-	-	-	-
LineColor (页 939)	RW	-	-	指定窗口分隔线的颜色。
LineStyle	-	-	-	-
LineWidth (页 943)	RW	-	-	指定所选对象的线宽。
Location	-	-	-	-
LongTermArchiveConsistency (页 949)	RW	-	-	指定组态“历史报警列表（长期）”(Historical alarm list (long-term)) 时，报警在报警视图中的显示方式。
MessageBlockAlignment (页 956)	RW	-	-	设置所选报警块的表格内容的对齐模式。
MessageBlockAutoPrecisions (页 957)	RW	-	-	启用自动设置小数位数。

属性	RT Professional	RT Advanced	Panel RT	说明
MessageBlockCaption (页 958)	RW	-	-	指定报警视图中所选报警文本块的列标题说明。
MessageBlockCount (页 958)	RW	-	-	指定可用于报警列表和统计列表的现有报警文本块的数量。
MessageBlockDateFormat (页 959)	RW	-	-	指定用于显示报警的日期格式。
MessageBlockExponentialFormat (页 960)	RW	-	-	指定用指数记数法显示所选报警块的值。
MessageBlockFlashOn (页 960)	RW	-	-	出现报警时在运行系统中对所选报警块启用闪烁功能。
MessageBlockHideText (页 961)	RW	-	-	启用所选报警块内容的文本显示。
MessageBlockHideTitleText (页 962)	RW	-	-	指定所选报警块说明的文本显示。
MessageBlockId (页 963)	RW	-	-	指定报警视图中 ID 编号和报警文本块的分配。
MessageBlockIndex (页 963)	RW	-	-	引用现有报警文本块。
MessageBlockLeadingZeros (页 964)	RW	-	-	启用使用前导零格式显示所选报警块。
MessageBlockLength (页 965)	RW	-	-	根据字符数指定所选报警块的长度。
MessageBlockName (页 965)	RW	-	-	指定所选报警文本块的名称。
MessageBlockPrecisions (页 966)	RW	-	-	为所选报警块的值指定小数位数。
MessageBlockSelected (页 966)	RW	-	-	现有报警块是报警列表或统计列表的可用于运行系统控件的数据块。
MessageBlockShowDate (页 967)	RW	-	-	指定是否在“时间”(Time) 报警文本块中显示日期和时间。
MessageBlockShowIcon (页 968)	RW	-	-	启用以图标形式显示所选报警块的内容。

属性	RT Professional	RT Advanced	Panel RT	说明
MessageBlockShowTitleIcon (页 968)	RW	-	-	指定所选报警块说明的文本显示。
MessageBlockTextId (页 969)	RW	-	-	指定通过从文本库获取的文本 ID 对所选报警文本块命名。
MessageBlockTimeFormat (页 970)	RW	-	-	指定用于显示报警的时间或时段的格式。
MessageBlockType (页 971)	RW	-	-	指定可用于报警列表和统计列表的现有报警文本块的数量。
MessageColumnAdd (页 972)	RW	-	-	将所选报警文本块从可用报警块列表添加到选定报警块列表。
MessageColumnCount (页 972)	RW	-	-	指定要显示在运行系统报警列表中的所选报警文本块的数量。
MessageColumnIndex (页 973)	RW	-	-	引用为报警列表选择的报警文本块。
MessageColumnName (页 973)	RW	-	-	指定由"MessageColumnIndex"属性引用的报警列表中的报警文本块名称。
MessageColumnRemove (页 974)	RW	-	-	从所选报警块列表中剪切标记的报警文本块，然后将其粘贴到现有报警块列表中。
MessageColumnRepos (页 975)	RW	-	-	指定报警文本块的顺序。
MessageColumnSort (页 975)	RW	-	-	指定使用"MessageColumnIndex"引用的报警文本块的排序顺序。
MessageColumnSortIndex (页 976)	RW	-	-	定义"MessageColumnIndex"中引用的报警文本块的排序顺序。
MessageColumnVisible (页 977)	RW	-	-	指定是否在报警视图中显示报警文本块。
MessageListType (页 977)	RW	-	-	此设置用于定义在画面打开时显示报警列表中的哪些内容。
Moveable (页 983)	RW	-	-	指定是否可在运行系统中移动窗口。
MsgFilterSQL (页 984)	RW	-	-	针对用户自选的报警指定一个或多个 SQL 语句。
Name	-	-	-	-

属性	RT Professional	RT Advanced	Panel RT	说明
Object	-	-	-	-
OcxGuid	-	-	-	-
OcxState	-	-	-	-
OcxStateForEs2Rt	-	-	-	-
OperatorAlarms	-	-	-	-
OperatorMessageId (页 1001)	RW	-	-	指定报警视图中 ID 编号和触发事件的分配。
OperatorMessageIndex (页 1002)	RW	-	-	引用操作员消息事件。
OperatorMessageName (页 1002)	RW	-	-	指定消息事件中由“OperatorMessageIndex”事件引用的操作员消息的名称。
OperatorMessageNumber (页 1003)	RW	-	-	在*不*使用 WinCC 操作员消息的情况下为所选消息事件的操作员消息指定消息编号。
OperatorMessageSelected (页 1004)	RW	-	-	激活列表中可以触发操作员消息的消息事件。
OperatorMessageSource1 (页 1004)	RW	-	-	指定要添加到在此处组态的操作员消息的“过程值块 1”的操作消息的报警文本块。
OperatorMessageSource10 (页 1011)	RW	-	-	指定要添加到在此处组态的操作员消息的“过程值块 10”的操作消息的报警文本块。
OperatorMessageSource2 (页 1005)	RW	-	-	指定要添加到在此处组态的操作员消息的“过程值块 2”的操作消息的报警文本块。
OperatorMessageSource3 (页 1006)	RW	-	-	指定要添加到在此处组态的操作员消息的“过程值块 3”的操作消息的报警文本块。
OperatorMessageSource4 (页 1007)	RW	-	-	指定要添加到在此处组态的操作员消息的“过程值块 4”的操作消息的报警文本块。
OperatorMessageSource5 (页 1007)	RW	-	-	指定要添加到在此处组态的操作员消息的“过程值块 5”的操作消息的报警文本块。
OperatorMessageSource6 (页 1008)	RW	-	-	指定要添加到在此处组态的操作员消息的“过程值块 6”的操作消息的报警文本块。
OperatorMessageSource7 (页 1009)	RW	-	-	指定要添加到在此处组态的操作员消息的“过程值块 7”的操作消息的报警文本块。

属性	RT Professional	RT Advanced	Panel RT	说明
OperatorMessageSource8 (页 1010)	RW	-	-	指定要添加到在此处组态的操作员消息的“过程值块 8”的操作消息的报警文本块。
OperatorMessageSource9 (页 1010)	RW	-	-	指定要添加到在此处组态的操作员消息的“过程值块 9”的操作消息的报警文本块。
OperatorMessageSourceType1 (页 1012)	RW	-	-	指定要传送的源内容的格式。
OperatorMessageSourceType10 (页 1018)	RW	-	-	指定要传送的源内容的格式。
OperatorMessageSourceType2 (页 1013)	RW	-	-	指定要传送的源内容的格式。
OperatorMessageSourceType3 (页 1013)	RW	-	-	指定要传送的源内容的格式。
OperatorMessageSourceType4 (页 1014)	RW	-	-	指定要传送的源内容的格式。
OperatorMessageSourceType5 (页 1015)	RW	-	-	指定要传送的源内容的格式。
OperatorMessageSourceType6 (页 1015)	RW	-	-	指定要传送的源内容的格式。
OperatorMessageSourceType7 (页 1016)	RW	-	-	指定要传送的源内容的格式。
OperatorMessageSourceType8 (页 1017)	RW	-	-	指定要传送的源内容的格式。
OperatorMessageSourceType9 (页 1018)	RW	-	-	指定要传送的源内容的格式。
PageMode (页 1020)	RW	-	-	支持在历史报警列表（长期）中滚动。
PageModeMessageNumber (页 1021)	RW	-	-	定义对历史报警列表（长期）分页时每页显示的报警数。
PrintJob (页 1036)	RW	-	-	指定在“报表”(Reports) 编辑器中创建的打印作业。
RowScrollbar (页 1052)	RW	-	-	指定是否显示行滚动条。
RowTitleAlignment (页 1053)	RW	-	-	指定行标题对齐的类型。

属性	RT Professional	RT Advanced	Panel RT	说明
RowTitles (页 1054)	RW	-	-	指定将显示行标题。
RTPersistence (页 1054)	RW	-	-	指定如何保留 WinCC 的在线组态。
RTPersistenceAuthorization (页 1055)	RW	-	-	指定运行系统中在线组态所需的权限。
RTPersistenceType (页 1057)	RW	-	-	指定如何保留 WinCC 的在线组态。
SelectedCellColor (页 1075)	RW	-	-	指定所选单元格的背景色。
SelectedCellForeColor (页 1076)	RW	-	-	指定所选单元格的字体颜色。
SelectedRowColor (页 1079)	RW	-	-	指定所选行的背景色。
SelectedRowForeColor (页 1080)	RW	-	-	指定所选行的字体颜色。
SelectedTitleColor (页 1081)	RW	-	-	指定所选表格标题的背景色。
SelectedTitleForeColor (页 1082)	RW	-	-	指定所选表格标题的字体颜色。
SelectionColoring (页 1085)	RW	-	-	指定是否对单元格或行使用选择颜色。
SelectionRect (页 1087)	RW	-	-	指定是否对所选单元格或行使用选择框架。
SelectionRectColor (页 1088)	RW	-	-	如果 SelectionType 等于“1”，则指定报警窗口中选择矩形的颜色。
SelectionRectWidth (页 1089)	RW	-	-	如果 SelectionType 等于“1”，则指定报警窗口中选择矩形的线宽。
SelectionType (页 1090)	RW	-	-	指定可标记的行数。
ServerNames (页 1096)	RW	-	-	指定构成报警视图数据源的分布式系统的服务器。
ShowSortButton (页 1117)	RW	-	-	指定是否在垂直滚动条上显示排序按钮。

属性	RT Professional	RT Advanced	Panel RT	说明
ShowSortIcon (页 1118)	RW	-	-	指定是否显示排序图标。
ShowSortIndex (页 1118)	RW	-	-	指定是否显示排序索引。
ShowTitle (页 1124)	RW	-	-	指定控件的窗口标题的显示形式。
Size	-	-	-	-
Sizeable (页 1129)	RW	-	-	指定可以在运行系统中更改对象的大小。
SortSequence (页 1132)	RW	-	-	指定如何通过鼠标单击更改排序顺序。
StatusbarBackColor (页 1138)	RW	-	-	指定状态栏的背景色。
StatusbarElementAdd (页 1139)	RW	-	-	创建一个新的用户自定义状态栏元素。
StatusbarElementAutoSize (页 1140)	RW	-	-	指定是否自动设置所选状态栏元素的宽度。
StatusbarElementCount (页 1141)	RW	-	-	指定可组态的状态栏元素的数量。
StatusbarElementIconId (页 1142)	RW	-	-	指定状态栏元素的 ID 编号和符号分配。
StatusbarElementID (页 1143)	RW	-	-	指定所选状态栏元素的 ID 编号。
StatusbarElementIndex (页 1144)	RW	-	-	指定状态栏元素的参考。
StatusbarElementName (页 1145)	RW	-	-	指定所选状态栏元素的对象名称。
StatusbarElementRemove (页 1146)	RW	-	-	删除所选用户自定义状态栏元素。
StatusbarElementRename (页 1147)	RW	-	-	更改“StatusbarElementIndex”属性引用的用户定义状态栏元素的名称。
StatusbarElementRepos (页 1148)	RW	-	-	指定显示和操作对象状态栏中的元素索引。
StatusbarElements	-	-	-	-

属性	RT Professional	RT Advanced	Panel RT	说明
StatusbarElementText (页 1149)	RW	-	-	指定所选状态栏元素的文本。
StatusbarElementTooltipText (页 1150)	RW	-	-	指定所选用户自定义状态栏元素的工具提示文本。
StatusbarElementUserDefined (页 1151)	RW	-	-	指定组态工程师是否已添加状态栏元素作为新的用户自定义元素。
StatusbarElementVisible (页 1152)	RW	-	-	指定是否在运行系统中显示状态栏元素的引用元素。
StatusbarElementWidth (页 1153)	RW	-	-	指定所引用状态栏元素的宽度（以像素为单位）。
StatusbarFont (页 1154)	RW	-	-	指定状态栏中文本的字体。
StatusbarFontColor (页 1155)	RW	-	-	指定状态栏中文本的颜色。
StatusbarShowTooltips (页 1156)	RW	-	-	指定是否在运行系统中显示状态栏元素的工具提示。
StatusbarText (页 1157)	RW	-	-	指定状态栏中的默认文本。
StatusbarUseBackColor (页 1158)	RW	-	-	指定是否显示状态栏的背景色。
StatusbarVisible (页 1159)	RW	-	-	指定是否显示控件的状态栏。
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableColor (页 1171)	RW	-	-	指定对象中表格行的背景色。
TableColor2 (页 1172)	RW	-	-	指定行的第二种背景色。
TableForeColor (页 1173)	RW	-	-	指定对象表格单元格中采用的文本颜色。
TableForeColor2 (页 1174)	RW	-	-	指定对象表格单元格中采用的第二种文本颜色。
TimeBase (页 1211)	RW	-	-	指定显示时间值的时区。
TitleColor (页 1236)	RW	-	-	指定表格标题的背景色。

属性	RT Professional	RT Advanced	Panel RT	说明
TitleCut (页 1237)	RW	-	-	指定列宽过窄时是否缩减标题栏中的字段内容。
TitleDarkShadowColor (页 1238)	RW	-	-	为对象列表中的列和行标题指定 3D 底纹的暗边颜色。
TitleForeColor (页 1239)	RW	-	-	指定对象中表格列和行标题的文本颜色。
TitleGridLineColor (页 1240)	RW	-	-	指定表格标题栏中分隔线的颜色。
TitleLightShadowColor (页 1241)	RW	-	-	指定对象表格列和行标题中 3D 底纹亮边的颜色。
TitleSort (页 1242)	RW	-	-	指定如何触发按列标题排序。
TitleStyle (页 1243)	RW	-	-	指定是否对表格标题使用底纹颜色。
ToolbarAlignment (页 1252)	RW	-	-	指定工具栏的位置。
ToolbarBackColor (页 1253)	RW	-	-	指定工具栏的背景色。
ToolbarButtonActive (页 1254)	RW	-	-	指定是否在运行系统中激活按钮的相应功能。
ToolbarButtonAdd (页 1255)	RW	-	-	在对象的工具栏中创建新的用户自定义按钮。
ToolbarButtonAuthorization (页 1256)	RW	-	-	显示所选按钮功能的权限。
ToolbarButtonBeginGroup (页 1257)	RW	-	-	在工具栏上，为所选键功能插入前导分隔符（垂直线）。
ToolbarButtonClick (页 1258)	RW	-	-	单击工具栏按钮。
ToolbarButtonCount (页 1259)	RW	-	-	指定工具栏中的按钮数目。
ToolbarButtonEnabled (页 1260)	RW	-	-	指定是否可以操作用户自定义工具栏按钮。
ToolbarButtonHotKey (页 1261)	RW	-	-	指定所选对象按钮的快捷方式。

属性	RT Professional	RT Advanced	Panel RT	说明
ToolbarButtonID (页 1262)	RW	-	-	通过唯一的 ID 编号指定键功能。
ToolbarButtonIndex (页 1263)	RW	-	-	引用按钮功能。
ToolbarButtonLocked (页 1264)	RW	-	-	启用/禁用用户自定义工具栏按钮锁定、按下状态的显示。
ToolbarButtonName (页 1265)	RW	-	-	指定所选用户自定义按钮的名称。
ToolbarButtonRemove (页 1266)	RW	-	-	删除所选用户自定义按钮。
ToolbarButtonRename (页 1267)	RW	-	-	指定通过"ToolbarButtonIndex"属性引用的用户自定义工具栏按钮的名称。
ToolbarButtonRepos (页 1268)	RW	-	-	在对象工具栏中指定引用按钮的位置。
ToolbarButtons	-	-	-	-
ToolbarButtonTooltipText (页 1270)	RW	-	-	指定工具栏中用户自定义按钮的工具提示文本。
ToolbarButtonUserDefined (页 1271)	RW	-	-	指定项目工程师是否已将此工具栏按钮添加为新的用户自定义按钮。
ToolbarButtonVisible (页 1272)	RW	-	-	指定是否在工具栏中显示按钮。
ToolbarShowTooltips (页 1273)	RW	-	-	指定是否在运行系统中显示按键功能的工具提示。
ToolbarUseBackColor (页 1275)	RW	-	-	指定是否显示工具栏的背景色。
ToolbarUseHotKeys (页 1276)	RW	-	-	指定是否在运行系统中激活按钮功能的热键。
ToolbarVisible (页 1277)	RW	-	-	指定是否显示控件的工具栏。
Top (页 1279)	RW	-	-	指定所选对象的 Y 坐标的值。
UseMessageColor (页 1360)	RW	-	-	指定是否显示消息类别的商定颜色。

属性	RT Professional	RT Advanced	Panel RT	说明
UseSelectedTitleColor (页 1362)	RW	-	-	指定是否将选择颜色用于选定表格单元格的标题。
UseTableColor2 (页 1364)	RW	-	-	指定是否用第二种行颜色来表现表格。
VerticalGridLines (页 1414)	RW	-	-	指定是否显示垂直分隔线。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-2　方法

方法	RT Professional	RT Advanced	Panel RT	说明
激活 (页 1470)	支持	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	支持	-	-	在运行期间，动态激活触发器和属性的指定周期。
AttachDB (页 1477)	支持	-	-	执行控件的“连接备份”键功能。
CopyRows (页 1479)	支持	-	-	执行控件的“复制行”键功能。
DeactivateDynamic (页 1481)	支持	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。
DetachDB (页 1484)	支持	-	-	执行控件的“断开备份”键功能。
导出 (页 1485)	支持	-	-	执行控件的“导出日志”或“导出数据”键功能。
GetHitlistColumn (页 1490)	支持	-	-	以“ICCAxMessageColumn”类型的形式返回报警视统计列表中具有指定名称或索引的列对象。
GetHistlistColumnCollection (页 1488)	支持	-	-	以“ICCAxCollection”类型的形式返回报警视图统计列表中所有列对象的列表。
GetMessageBlock (页 1491)	支持	-	-	以“ICCAxMessageBlock”类型的形式返回报警视图中按名称或索引指定的报警文本块对象。
GetMessageBlockCollection (页 1492)	支持	-	-	以“ICCAxCollection”类型的形式返回报警视图中所有报警文本块对象的列表。

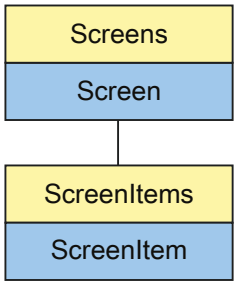
方法	RT Professional	RT Advanced	Panel RT	说明
GetMessageColumn (页 1493)	支持	-	-	以“ICCAxMessageColumn”类型的形式返回报警视图中按名称或索引指定的列对象。
GetMessageColumnCollection (页 1494)	支持	-	-	以“ICCAxCollection”类型的形式返回报警视图中所有列对象的列表。
GetOperatorMessage (页 1496)	支持	-	-	以“ICCAxOperatorMessage”类型的形式返回报警视图中按名称或索引指定的操作员消息对象。
GetOperatorMessageCollection (页 1497)	支持	-	-	以“ICCAxCollection”类型的形式返回报警视图中所有操作员消息对象的列表。
GetRow (页 1498)	支持	-	-	以“ICCAxDataRow”类型的形式返回基于表格的控件中的行编号所定义的行对象。
GetRowCollection (页 1499)	支持	-	-	以“ICCAxDataRowCollection”类型的形式返回基于表格的控件的所有行对象的列表。
GetSelectedRow (页 1507)	支持	-	-	以“ICCAxDataRow”类型的形式返回基于表格的控件的所选行对象。
GetSelectedRows (页 1508)	支持	-	-	以“ICCAxDataRow”类型的形式返回用于多项选择的基于表格的控件的所选行对象。
GetStatusbarElement (页 1514)	支持	-	-	以“ICCAxStatusbarElement”类型的形式返回按名称或索引指定的控件状态栏元素。
GetStatusbarElementCollection (页 1516)	支持	-	-	以“ICCAxCollection”类型的形式返回控件的所有状态栏元素的列表。
GetToolbarButton (页 1523)	支持	-	-	以“ICCAxToolbarButton”类型的形式返回控件工具栏上按名称或索引指定的按钮功能。
GetToolbarButtonCollection (页 1525)	支持	-	-	以“ICCAxCollection”类型的形式返回控件工具栏的所有按钮功能的列表。
HideAlarm (页 1543)	支持	-	-	执行报警视图的“隐藏报警”(Hide alarm) 按钮功能。
LockAlarm (页 1546)	支持	-	-	执行报警视图的“禁用报警”(Disable alarm) 按钮功能。
LoopInAlarm (页 1546)	支持	-	-	执行报警视图的“报警回路”(Loop in alarm) 按钮功能。
MoveToFirstLine (页 1548)	支持	-	-	执行报警视图的“第一个报警”(First alarm) 按钮功能。

方法	RT Professional	RT Advanced	Panel RT	说明
MoveToFirstPage (页 1548)	支持	-	-	执行报警视图的“第一页”(First page) 按钮功能。
MoveToLastLine (页 1550)	支持	-	-	执行报警视图的“最后一个报警”(Last alarm) 按钮功能。
MoveToLastPage (页 1550)	支持	-	-	执行报警视图的“最后一页”(Last page) 按钮功能。
MoveToNextLine (页 1551)	支持	-	-	执行报警视图的“下一个报警”(Next alarm) 按钮功能。
MoveToNextPage (页 1552)	支持	-	-	执行报警视图的“下一页”(Next page) 按钮功能。
MoveToPreviousLine (页 1553)	支持	-	-	执行报警视图的“上一个报警”(Previous alarm) 按钮功能。
MoveToPreviousPage (页 1553)	支持	-	-	执行报警视图的“上一页”(Previous page) 按钮功能。
Print (页 1557)	支持	-	-	执行控件的“打印”(Print) 按钮功能。
QuitHorn (页 1558)	支持	-	-	执行报警视图的“报警器确认”(Alarm annunciator acknowledgment) 按钮功能。
QuitSelected (页 1558)	支持	-	-	执行报警视图的“单个确认”(Single acknowledgment) 按钮功能。
QuitVisible (页 1559)	支持	-	-	执行报警视图的“组确认”(Group acknowledgment) 按钮功能。
SelectAll (页 1573)	支持	-	-	在基于表格的控件中选择所有行。
SelectRow (页 1575)	支持	-	-	在基于表格的控件中选择特定行。
ShowComment (页 1579)	支持	-	-	执行报警视图的“注释对话框”(Comment dialog) 按钮功能。
ShowDisplayOptionsDialog (页 1579)	支持	-	-	执行报警视图的“显示选项对话框”(Display options dialog) 按钮功能。
ShowEmergencyQuitDialog (页 1580)	支持	-	-	执行报警视图的“紧急确认”(Emergency acknowledgment) 按钮功能。
ShowHelp (页 1580)	支持	-	-	执行控件的“帮助”(Help) 按钮功能。
ShowHideList (页 1581)	支持	-	-	执行报警视图的“要隐藏的报警的列表”(List of alarm to hide) 按钮功能。

方法	RT Professional	RT Advanced	Panel RT	说明
ShowHitList (页 1582)	支持	-	-	执行报警视图的“统计列表”(Hit list) 按钮功能。
ShowInfoText (页 1582)	支持	-	-	执行报警视图的“关于对话框”(About dialog) 按钮功能。
ShowLockDialog (页 1583)	支持	-	-	执行报警视图的“锁定对话框”(Lock dialog) 按钮功能。
ShowLockList (页 1583)	支持	-	-	执行报警视图的“锁定列表”(Lock list) 按钮功能。
ShowLongTermArchiveList (页 1584)	支持	-	-	执行报警视图的“历史报警列表（长期）”(Historical alarm list (long-term)) 按钮功能。
ShowMessageList (页 1584)	支持	-	-	执行报警视图的“报警列表”(Alarm list) 按钮功能。
ShowPropertyDialog (页 1585)	支持	-	-	执行控件的“组态对话框”(Configuration dialog) 按钮功能。
ShowSelectionDialog (页 1587)	支持	-	-	执行报警视图的“选择对话框”(Selection dialog) 按钮功能。
ShowShortTermArchiveList (页 1588)	支持	-	-	执行报警视图的“历史报警列表（短期）”(Historical alarm list (short-term)) 按钮功能。
ShowSortDialog (页 1589)	支持	-	-	执行报警视图的“排序对话框”(Sorting dialog) 按钮功能。
ShowTimebaseDialog (页 1590)	支持	-	-	执行报警视图的“时基对话框”(Timebase dialog) 按钮功能。
UnhideAlarm (页 1594)	支持	-	-	执行报警视图的“显示报警”(Unhide alarm) 按钮功能。
UnlockAlarm (页 1594)	支持	-	-	执行报警视图的“解锁报警”(Unlock alarm) 按钮功能。
UnselectAll (页 1595)	支持	-	-	在基于表格的控件的单元格中移除所有选择内容。
UnselectRow (页 1595)	支持	-	-	在基于表格的控件的特定单元格中移除选择内容。

AlarmView

说明



表示“Alarm view”对象。AlarmView 对象是 ScreenItems 列表的元素。

如果通过用户自定义函数更改此对象的设置，则即使再次调用画面后，更改的设置仍然会保留。

说明

“简单 AlarmView”对象不能通过用户定义的函数进行动态化。

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-3　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AlarmAreaHeight	-	-	-	-
AlarmAreaWidth	-	-	-	-
AlarmClasses	-	-	-	-
AlarmLog	-	-	-	-
AlarmSource	-	-	-	-
AlarmTextTag	-	-	-	-
AllTagTypesAllowed	-	-	-	-
Authorization	-	-	-	-
BackColor (页 637)	-	RW	RW	指定所选对象的背景色。
BorderBackColor	-	-	-	-
BorderColor	-	-	-	-
BorderWidth	-	-	-	-
Bounds	-	-	-	-
ButtonBackColor	-	-	-	-
ButtonBackFillStyle	-	-	-	-
ButtonBarElements	-	-	-	-
ButtonBarStyle	-	-	-	-
ButtonBorderBackColor	-	-	-	-
ButtonBorderColor	-	-	-	-
ButtonBorderWidth	-	-	-	-
ButtonCornerRadius	-	-	-	-
ButtonEdgeStyle	-	-	-	-
ButtonFirstGradientColor	-	-	-	-
ButtonFirstGradientOffset	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
ButtonMiddleGradientColor	-	-	-	-
ButtonPositions	-	-	-	-
ButtonSecondGradientColor	-	-	-	-
ButtonSecondGradientOffset	-	-	-	-
CanBeGrouped	-	-	-	-
ColumnOrder	-	-	-	-
Columns	-	-	-	-
ColumnsMoveable	-	-	-	-
ColumnTextAckGroup	-	-	-	-
ColumnTextAlarmState	-	-	-	-
ColumnTextAlarmText	-	-	-	-
ColumnTextClassName	-	-	-	-
ColumnTextDate	-	-	-	-
ColumnTextDevice	-	-	-	-
ColumnTextDiagnosable	-	-	-	-
ColumnTextNumber	-	-	-	-
ColumnTextTime	-	-	-	-
ColumnWidth	-	-	-	-
ConfiguredAlarmClasses	-	-	-	-
CornerRadius	-	-	-	-
CountOfLinesPerAlarms	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
CountOfVisibleAlarms	-	-	-	-
DeviceStyle	-	-	-	-
EdgeStyle	-	-	-	-
Enabled (页 792)	-	RW	RW	指定是否可以在运行系统中操作所选对象。
ES2RT_ButtonPositions	-	-	-	-
ES2RT_ColumnOrder	-	-	-	-
ES2RT_ColumnWidth	-	-	-	-
ES2RT_MessageAreaHeight	-	-	-	-
ES2RT_MessageAreaWidth	-	-	-	-
FilterTag	-	-	-	-
FilterText	-	-	-	-
FitToSize	-	-	-	-
Flashing	-	-	-	-
FocusColor (页 843)	-	RW	RW	当对象获得焦点时，指定焦点边框的颜色。
FocusWidth (页 844)	-	RW	RW	当对象获得焦点时，指定边框宽度。
ForeColor	-	-	-	-
GridLineColor (页 860)	-	RW	RW	指定网格线颜色。
Height	-	R	R	指定高度。
HorizontalScrollingEnabled	-	-	-	-
IsRunningUnderCE	-	-	-	-
Layer (页 913)	-	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	-	RW	RW	指定 X 坐标的值。
LineAlarmView	-	-	-	-
Location	-	-	-	-
MessageAreaHeight	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
MessageAreaLeft	-	-	-	-
MessageAreaTop	-	-	-	-
MessageAreaWidth	-	-	-	-
Name	-	-	-	-
PaddingBottom	-	-	-	-
PaddingLeft	-	-	-	-
PaddingRight	-	-	-	-
PaddingTop	-	-	-	-
PreferredUseOnAck	-	-	-	-
S7Device	-	-	-	-
SecurityForSimpleViewEnabled	-	-	-	-
SelectionBackColor (页 1084)	-	RW	RW	指定所选单元格的背景色。
SelectionForeColor (页 1086)	-	RW	RW	指定所选单元格的前景色。
SeparateLineForAlarmText	-	-	-	-
ShowAcknowledgeButton	-	-	-	-
ShowAlarmsFromDate (页 1098)	-	RW	RW	指定仅显示该变量中保存的那些消息事件。
ShowAlarmsToAcknowledge	-	-	-	-
ShowColumnHeaders	-	-	-	-
ShowHelpButton	-	-	-	-
ShowHorizontalGridlines	-	-	-	-
ShowLoopInAlarmButton	-	-	-	-
ShowMilliseconds	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
ShowPendingAlarms	-	-	-	-
Size	-	-	-	-
SortByTimeDirection (页 1131)	-	RW	RW	指定是否在顶部显示最后接收到的消息（按升序顺序排序）。
SortByTimeEnabled (页 1131)	-	RW	RW	指定是否按照时间改变报警的排序。
StyleItem	-	-	-	-
SupportsS7DiagnosticsInSimpleView	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableBackColor (页 1170)	-	RW	RW	指定表格单元格的背景色。
TableEvenRowBackColor	-	-	-	-
TableFont	-	-	-	-
TableForeColor (页 1173)	-	RW	RW	指定对象表格单元格中采用的文本颜色。
TableHeaderBackColor (页 1176)	-	RW	RW	指定表格标题的背景色。
TableHeaderBackFillStyle	-	-	-	-
TableHeaderBorderBackColor	-	-	-	-
TableHeaderBorderColor	-	-	-	-
TableHeaderBorderWidth	-	-	-	-
TableHeaderCornerRadius	-	-	-	-
TableHeaderEdgeStyle	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
TableHeaderFirstGradientColor	-	-	-	-
TableHeaderFirstGradientOffset	-	-	-	-
TableHeaderFont	-	-	-	-
TableHeaderForeColor (页 1179)	-	RW	RW	指定表格标题的文本颜色。
TableHeaderMiddleGradientColor	-	-	-	-
TableHeaderPaddingBottom	-	-	-	-
TableHeaderPaddingLeft	-	-	-	-
TableHeaderPaddingRight	-	-	-	-
TableHeaderPaddingTop	-	-	-	-
TableHeaderSecondGradientColor	-	-	-	-
TableHeaderSecondGradientOffset	-	-	-	-
ToolbarHeight	-	-	-	-
ToolbarLeft	-	-	-	-
ToolbarTop	-	-	-	-
ToolbarWidth	-	-	-	-
Top (页 1279)	-	RW	RW	指定 Y 坐标的值。
UseButtonFirstGradient	-	-	-	-
UseButtonSecondGradient	-	-	-	-
UseDesignColorSchema	-	-	-	-

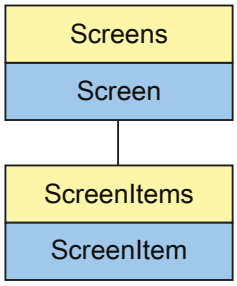
属性	RT Professional	RT Advanced	面板 RT	描述
UseTableHeaderFirstGradient	-	-	-	-
UseTableHeaderSecondGradient	-	-	-	-
VerticalScrollBarEnabled	-	-	-	-
VerticalScrollingEnabled	-	-	-	-
ViewType	-	-	-	-
ViewTypeForSaveStream	-	-	-	-
Visible (页 1418)	-	RW	RW	指定是否显示所选对象。
Width	-	R	R	指定宽度。

表格 1-4　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	-	√	√	启用永久区域或根画面。

ApplicationWindow

说明



表示“ApplicationWindow”对象。ApplicationWindow 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIApplicationWindow

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-5　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
BorderEnabled (页 682)	RW	-	-	指定窗口在运行系统中是否带边框显示。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
GSCRuntimeAllowed	-	-	-	-
Height (页 863)	RW	-	-	指定高度。
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
Location	-	-	-	-
Name	-	-	-	-
ShowCaption (页 1100)	RW	-	-	指定是否显示工具栏。
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Template (页 1184)	RW	-	-	指定用于显示“打印作业/脚本诊断”(Print job/ Script diagnostics) 对象窗口内容的模板。
Top (页 1279)	-	-	-	-
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。
WindowCloseEnabled (页 1436)	RW	-	-	指定是否可在运行系统中关闭窗口。
WindowMaximizeEnabled (页 1436)	RW	-	-	指定运行系统中是否可以最大化对象。
WindowMovingEnabled (页 1437)	RW	-	-	指定运行系统中是否可以移动对象。

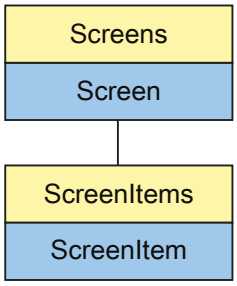
属性	RT Professional	RT Advanced	面板 RT	描述
WindowOnTop (页 1438)	RW	-	-	指定对象是否始终位于运行系统的前景中。
WindowsContents (页 1439)	RW	-	-	指定打印作业或脚本诊断的内容。
WindowSizingEnabled (页 1439)	RW	-	-	指定是否可更改大小。

表格 1-6　方法

方法	RT Professional	RT Advanced	面板 RT	描述
Activate (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

Bar

描述



表示“Bar”对象。Bar 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIBar

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-7　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AlarmLowerLimit (页 611)	RW	-	-	指定触发报警的下限值。
AlarmLowerLimitColor (页 612)	RW	-	-	指定"AlarmLowerLimit"极值的滑块颜色。
AlarmLowerLimitEnabled (页 613)	RW	-	-	指定是否监视"AlarmLowerLimit"限值。
AlarmLowerLimitRelative (页 614)	RW	-	-	指定触发中断的下限值是以百分比还是绝对值进行估算。
AlarmUpperLimit (页 615)	RW	-	-	指定触发中断的上限值。
AlarmUpperLimitColor (页 615)	RW	-	-	指定"AlarmUpperLimit"极值的滑块颜色。
AlarmUpperLimitEnabled (页 616)	RW	-	-	指定是否监视"AlarmUpperLimit"限值。
AlarmUpperLimitRelative (页 617)	RW	-	-	指定触发中断的上限值是以百分比还是绝对值进行估算。
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定操作权限。
AverageLast15Values (页 635)	RW	-	-	指定是否显示最后 15 个值的平均值。
BackColor (页 637)	RW	RW	RW	指定背景色。
BackFillStyle (页 643)	RW	-	-	指定填充图案。
BackFlashingColorOff (页 645)	RW	-	-	指定闪烁状态 "关闭"(Off) 的背景色。
BackFlashingColorOn (页 646)	RW	-	-	指定闪烁状态 "打开"(On) 的背景色。
BackFlashingEnabled (页 648)	RW	-	-	指定是否在运行系统中闪烁背景。

属性	RT Professional	RT Advanced	面板 RT	描述
BackFlashingRate (页 649)	RW	-	-	指定背景的闪烁频率。
BarBackColor (页 653)	RW	RW	RW	确定所选对象的棒图的背景色。
BarBackFillStyle (页 653)	RW	-	-	指定棒图的填充样式。
BarBackFlashingColorOff	-	-	-	-
BarBackFlashingColorOn	-	-	-	-
BarBackFlashingEnabled	-	-	-	-
BarBackFlashingRate	-	-	-	-
BarEdgeStyle	-	-	-	-
BarOrientation (页 658)	RW	-	-	指定棒图对齐方式。
BorderBackColor (页 674)	RW	-	-	指定间断边框线的背景色。
BorderColor (页 678)	RW	-	-	指定线颜色。
BorderFlashingColorOff (页 683)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
BorderFlashingColorOn (页 685)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
BorderFlashingEnabled (页 687)	RW	-	-	指定是否在运行系统中闪烁对象的限值。
BorderFlashingRate (页 689)	RW	-	-	指定边框线的闪烁速率。
BorderStyle (页 695)	RW	-	-	指定边框线的闪烁类型。
BorderWidth (页 697)	RW	-	-	指定所选对象的线宽。
BorderWidth3D	-	-	-	-
Bounds	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
CanBeGrouped	-	-	-	-
ColorChangeHysteresis (页 724)	RW	-	-	将滞后值指定为显示值的百分数。
ColorChangeHysteresisEnabled (页 725)	RW	-	-	指定对象是否滞后显示。
CompatibilityMode	-	-	-	-
CornerRadius	-	-	-	-
CornerStyle (页 764)	RW	-	-	指定特定对象的角形状。
CountDivisions (页 766)	RW	-	-	指定通过主要刻度标记可将棒图分割成的段数。
CountSubDivisions (页 767)	-	RW	RW	指定两个主要标记之间的刻度标记数。
DeviceStyle	-	-	-	-
DrawInsideFrame (页 787)	RW	-	-	指定线宽大于 1 的所选对象的边框线是在边框内描绘，还是与边框对称。
EdgeStyle (页 788)	RW	-	-	指定所选对象的线样式。
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。
FillPatternColor (页 821)	RW	-	-	确定所选对象的填充样式的颜色。
Flashing	-	-	-	-
FlashingColorOff	-	-	-	-
FlashingColorOn	-	-	-	-
FlashingEnabled	-	-	-	-
FlashingOnLimitViolation	-	-	-	-
FlashingRate	-	-	-	-
Font	-	-	-	-
FontBold (页 849)	RW	-	-	指定所选对象的文本是否以粗体显示。
FontName (页 851)	RW	-	-	指定所选对象的字体。
FontSize (页 852)	RW	-	-	指定所选对象的文本字体大小。
ForeColor (页 854)	RW	RW	RW	指定所选对象的文本字体颜色。

属性	RT Professional	RT Advanced	面板 RT	描述
ForeColorTransparency	-	-	-	-
Height (页 863)	RW	R	R	指定所选对象的高度。
InnerHeight	-	-	-	-
InnerWidth	-	-	-	-
IntegerDigits (页 895)	RW	-	-	指定小数点左边的位数（0 到 20）。
LargeTickLabelingStep (页 909)	R	-	-	返回加标签的那些刻度部分。
LargeTicksBold (页 909)	RW	-	-	指定主要刻度标记是否以粗体显示。
LargeTicksSize (页 910)	RW	-	-	指定主要刻度标记的长度。
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定所选对象的 X 坐标的值。
Limit4LowerLimit (页 927)	RW	-	-	指定"Reserve4"的下限值。
Limit4LowerLimitColor (页 927)	RW	-	-	指定"Reserve4"下限值的颜色。
Limit4LowerLimitEnabled (页 928)	RW	-	-	指定是否监视"Reserve4"的下限值。
Limit4LowerLimitRelative (页 929)	RW	-	-	指定"Reserve4"下限值是以百分比还是绝对值进行估算。
Limit4UpperLimit (页 930)	RW	-	-	指定"Reserve4"的上限值。
Limit4UpperLimitColor (页 930)	RW	-	-	指定"Reserve4"上限值的颜色。
Limit4UpperLimitEnabled (页 931)	RW	-	-	指定是否监视"Reserve4"的上限值。
Limit4UpperLimitRelative (页 932)	RW	-	-	指定"Reserve4"上限值是以百分比还是绝对值进行估算。

属性	RT Professional	RT Advanced	面板 RT	描述
Limit5LowerLimit (页 933)	RW	-	-	指定“Reserve5”的下限值。
Limit5LowerLimitColor (页 933)	RW	-	-	设置“Reserve5”下限值的颜色。
Limit5LowerLimitEnabled (页 934)	RW	-	-	指定是否监视“Reserve5”的下限值。
Limit5LowerLimitRelative (页 935)	RW	-	-	指定“Reserve5”下限值是以百分比还是绝对值进行估算。
Limit5UpperLimit (页 936)	RW	-	-	指定“Reserve5”的上限值。
Limit5UpperLimitColor (页 936)	RW	-	-	指定“Reserve5”上限值的颜色。
Limit5UpperLimitEnabled (页 937)	RW	-	-	指定是否监视“Reserve5”的上限值。
Limit5UpperLimitRelative (页 938)	RW	-	-	指定“Reserve5”上限值是以百分比还是绝对值进行估算。
LimitRangeCollection	-	-	-	-
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
Location	-	-	-	-
MaximumValue (页 953)	RW	RW	RW	指定所选对象的刻度范围的最大值。
MinimumValue (页 979)	RW	RW	RW	指定所选对象的刻度范围的最小值。
Name	-	-	-	-
Precision (页 1035)	RW	-	-	确定小数位的个数（0 到 20）。
ProcessValue (页 1037)	RW	RW	RW	指定要显示的数值的默认值。
ScaleColor (页 1060)	RW	RW	RW	指定所选对象的刻度的颜色。
ScaleGradation (页 1061)	RW	RW	RW	指定两个主要刻度标记之间的距离。

属性	RT Professional	RT Advanced	面板 RT	描述
ScaleLabelFieldLength	-	-	-	-
ScaleLabelingDoubleLined	-	-	-	-
ScalePosition (页 1064)	RW	-	-	指定所选对象的刻度位置。
ScaleStart	-	-	-	-
ScalingType (页 1068)	RW	-	-	指定棒图标尺的类型。
SegmentColoring (页 1073)	RW	RW	RW	指定超出限制值时要显示的颜色更改类型。
ShowBadTagState (页 1099)	RW	-	-	定义检测到不良质量代码或变量状态时对象是否变成灰色。
ShowLargeTicksOnly (页 1110)	RW	-	-	指定是否仅显示大刻度线。
ShowLimitLines	-	-	-	-
ShowLimitMarkers (页 1111)	RW	-	-	指定极值是否显示为刻度值。
ShowLimitRanges	-	-	-	-
ShowProcessValue	-	-	-	-
ShowScale (页 1115)	RW	-	-	指定数值是否还显示在刻度范围内。
ShowSignForPositiveLabel	-	-	-	-
ShowTickLabels	-	-	-	-
ShowTrendIndicator (页 1127)	RW	-	-	指定要监视的测量值的趋势（上升或下降）是否通过小箭头指示。
Size	-	-	-	-
StartValue (页 1136)	RW	-	-	定义刻度指示器上零点的绝对值。
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToleranceLowerLimit (页 1246)	RW	-	-	设置容差 1 的下限。

属性	RT Professional	RT Advanced	面板 RT	描述
ToleranceLowerLimitColor (页 1246)	RW	-	-	指定“ToleranceLowerLimit”下限的颜色。
ToleranceLowerLimitEnabled (页 1247)	RW	-	-	指定是否监视“ToleranceLowerLimit”极值。
ToleranceLowerLimitRelative (页 1248)	RW	-	-	指定“ToleranceLowerLimit”下限是以百分比还是绝对值进行估算。
ToleranceUpperLimit (页 1248)	RW	-	-	设置容差 1 的上限。
ToleranceUpperLimitColor (页 1249)	RW	-	-	指定“ToleranceUpperLimit”上限值的颜色。
ToleranceUpperLimitEnabled (页 1250)	RW	-	-	指定是否监视“ToleranceUpperLimit”限值。
ToleranceUpperLimitRelative (页 1251)	RW	-	-	指定“ToleranceUpperLimit”上限是以百分比还是绝对值进行估算。
ToolTipText (页 1278)	RW	RW	RW	指定工具提示文本。
Top (页 1279)	RW	RW	RW	指定所选对象的 Y 坐标的值。
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
TrendIndicatorColor (页 1299)	RW	-	-	指定趋势视图的颜色。
Unit (页 1346)	RW	-	-	指定测量单位。
UseAutoScaling	-	-	-	-
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
UseExponentialFormat (页 1359)	RW	-	-	指定显示的数值是否带有指数（例如：“1.00e+000”）。
Visible (页 1418)	RW	RW	RW	指定是否显示所选对象。
WarningLowerLimit (页 1424)	RW	-	-	指定“WarningLowerLimit”的下限值。

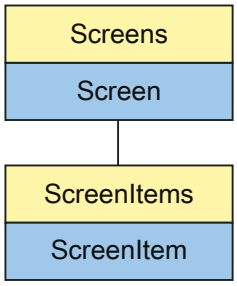
属性	RT Professional	RT Advanced	面板 RT	描述
WarningLowerLimitColor (页 1424)	RW	-	-	指定"WarningLowerLimit"下限的颜色。
WarningLowerLimitEnabled (页 1425)	RW	-	-	指定是否监视"WarningLowerLimit"限值。
WarningLowerLimitRelative (页 1426)	RW	-	-	指定"WarningLowerLimit"下限是以百分比还是绝对值形式进行估算。
WarningUpperLimit (页 1429)	RW	-	-	指定警告上限。
WarningUpperLimitColor (页 1430)	RW	R	R	指定"WarningUpperLimit"上限值的颜色。
WarningUpperLimitEnabled (页 1431)	RW	R	R	指定是否监视上限。
WarningUpperLimitRelative (页 1431)	RW	RW	RW	指定"WarningUpperLimit"上限是以百分比还是绝对值进行估算。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。
ZeroPoint (页 1468)	RW	RW	RW	指定零点位置作为棒图高度的百分比。

表格 1-8　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	√	√	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的"ActivateDynamic"方法的触发器。

BatteryView

描述



表示“Charge condition”对象。BatteryView 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIBatteryView

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-9　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
CanBeGrouped	-	-	-	-
Height	-	R	R	指定高度。
Layer (页 913)	-	-	RW	在包含对象的画面中指定图层。
Left (页 920)	-	-	RW	指定 X 坐标的值。
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (页 1279)	-	-	RW	指定 Y 坐标的值。
Visible (页 1418)	-	-	RW	指定是否显示所选对象。
Width	-	R	R	指定宽度。

表格 1-10　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	-	-	√	启用永久区域或根画面。

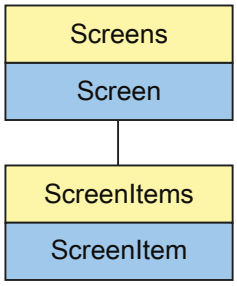
参见

Height (页 863)

Width (页 1432)

Button

说明



表示“Button”对象。Button 对象是 ScreenItems 列表的元素。

下列对象属性的可用性取决于所选的“Button”模式。

属性	“文本”模式	“文本列表”模式	“图形”模式
TextOff	x	--	--
TextOn	x	--	--

VBS 的类型标识符

HMIButton

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-11　属性

属性	RT Professional	RT Advanced	Panel RT	说明
AdaptBorder (页 606)	RW	-	-	指定对象的边框是否根据文本的大小进行动态调整。
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	RW	RW	指定所选对象在运行系统中的操作权限。
BackColor (页 637)	RW	RW	RW	指定所选对象的背景色。
BackFillStyle (页 643)	RW	-	-	指定所选对象的填充图案。
BackFlashingColorOff (页 645)	RW	-	-	指定闪烁状态“关闭”(Off) 的背景色。
BackFlashingColorOn (页 646)	RW	-	-	指定闪烁状态“打开”(On) 的背景色。
BackFlashingEnabled (页 648)	RW	-	-	指定是否在运行系统中闪烁指定对象的背景。
BackFlashingRate (页 649)	RW	-	-	指定所选对象的背景闪烁的频率。
BitNumber	-	-	-	-
BorderBackColor (页 674)	RW	-	-	指定所选对象间断边框线的背景色。
BorderBrightColor3D (页 677)	RW	-	-	指定所选对象 3D 显示中以下边框部分的边框颜色：外边框的顶部和底部；内边框的底部和右部
BorderColor (页 678)	RW	-	-	指定对象的线颜色。
BorderFlashingColorOff (页 683)	RW	-	-	指定闪烁状态“关闭”(Off) 的所选对象的边框线颜色。
BorderFlashingColorOn (页 685)	RW	-	-	指定闪烁状态“打开”(On) 的所选对象的边框线颜色。
BorderFlashingEnabled (页 687)	RW	-	-	指定所选对象的边框线是否在运行系统中闪烁。

属性	RT Professional	RT Advanced	Panel RT	说明
BorderFlashingRate (页 689)	RW	-	-	指定所选对象的边框线的闪烁频率。
BorderShadeColor3D (页 694)	RW	-	-	指定所选对象 3D 显示中以下边框部分的边框颜色：内边框的顶部和底部；外边框的底部和右部
BorderStyle (页 695)	RW	-	-	指定所选对象边框线的类型。
BorderWidth (页 697)	RW	-	-	指定所选对象的线宽。
BorderWidth3D (页 700)	RW	-	-	以 3D 形式表现所选择的对象时，指定内边框的宽度。
CanBeGrouped	-	-	-	-
CornerRadius	-	-	-	-
CornerStyle (页 764)	RW	-	-	指定特定对象的角形状。
DeviceStyle	-	-	-	-
DrawInsideFrame (页 787)	RW	-	-	指定线宽大于 1 的所选对象的边框线是在边框内描绘，还是与边框对称。
EdgeStyle (页 788)	RW	-	-	指定所选对象的线样式。
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。
ES2RT_StoreAsCheckBack	-	-	-	-
FillingDirection (页 825)	RW	-	-	指定填充方向。
FillPatternColor (页 821)	RW	-	-	指定所选对象的填充样式的颜色。
FirstGradientColor	-	-	-	-
FirstGradientOffset	-	-	-	-
FitToLargest	-	-	-	-
Flashing	-	-	-	-
FlashingColorOff (页 832)	RW	-	-	指定闪烁状态“关闭”(Off) 的所选对象的边框线颜色。
FlashingColorOn (页 834)	RW	-	-	指定闪烁状态“打开”(On) 的所选对象的边框线颜色。

属性	RT Professional	RT Advanced	Panel RT	说明
FlashingEnabled (页 836)	RW	-	-	指定所选对象的边框线是否在运行系统中闪烁。
FlashingRate (页 839)	RW	-	-	指定所选对象的边框线的闪烁频率。
FocusColor (页 843)	-	RW	RW	指定聚焦所选对象时其焦点框的颜色。
FocusWidth (页 844)	-	RW	RW	指定聚焦指定对象时其边框宽度。
Font	-	-	-	-
FontBold (页 849)	RW	-	-	指定所选对象的文本是否以粗体显示。
FontItalic (页 849)	RW	-	-	指定所选对象的文本是否以斜体显示。
FontName (页 851)	RW	-	-	指定所选对象的字体。
FontSize (页 852)	RW	-	-	指定所选对象的文本字体大小。
FontUnderline (页 853)	RW	-	-	指定所选对象的文本是否加下划线。
ForeColor (页 854)	RW	RW	RW	指定所选对象的文本字体颜色。
Height (页 863)	RW	R	R	指定所选对象的高度。
HelpText (页 868)	-	RW	RW	返回显示在运行系统中的工具提示，作为指定对象的用户帮助。
HorizontalAlignment (页 881)	RW	RW	RW	指定所选对象内的文本水平对齐。
HorizontalPictureAlignment	-	-	-	-
Hotkey	-	-	-	-
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定所选对象的 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
Location	-	-	-	-
MiddleGradientColor	-	-	-	-
Mode (页 981)	RW	-	-	指定所选对象的字段类型。
Name	-	-	-	-

属性	RT Professional	RT Advanced	Panel RT	说明
PictureAlignment (页 1027)	RW	-	-	指定过程映像中背景画面的显示类型。
PictureAreaBottomMargin	-	-	-	-
PictureAreaLeftMargin	-	-	-	-
PictureAreaRightMargin	-	-	-	-
PictureAreaTopMargin	-	-	-	-
PictureAutoSizing	-	-	-	-
PictureList	-	-	-	-
PictureOff (页 1029)	RW	-	-	指定在“关闭”(Off) 状态下显示的画面。
PictureOn (页 1030)	RW	-	-	指定在“打开”(On) 状态下显示的画面。
Pressed (页 1036)	RW	-	-	指定所选对象是否在“已按下”状态下显示。
ProcessValue (页 1037)	RW	-	-	指定要显示的数值的默认值。
RelativeFillLevel (页 1046)	RW	-	-	指定对象的填充百分比。
SecondGradientColor	-	-	-	-
SecondGradientOffset	-	-	-	-
ShowFillLevel (页 1108)	RW	-	-	指定所选对象是否填充。
Size	-	-	-	-
StyleItem	-	-	-	-
StyleSettings (页 1164)	RW	-	-	指定对象的显示样式。
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-

属性	RT Professional	RT Advanced	Panel RT	说明
TextAreaBottomMargin	-	-	-	-
TextAreaLeftMargin	-	-	-	-
TextAreaRightMargin	-	-	-	-
TextAreaTopMargin	-	-	-	-
TextList	-	-	-	-
TextOff (页 1187)	RW	RW	RW	指定要在所选对象处于“关闭”(Off) 状态时显示的文本。
TextOn (页 1188)	-	RW	RW	指定要在所选对象处于“打开”(On) 状态时显示的文本。
TextOrientation (页 1189)	RW	-	-	指定所选对象的文本方向。
Toggle (页 1244)	RW	-	-	指定在运行系统中对所选对象操作完后是否仍占用该对象。
ToolTipText (页 1278)	RW	RW	RW	指定工具提示文本。
Top (页 1279)	RW	RW	RW	指定所选对象的 Y 坐标的值。
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
TransparentColorPictureOff (页 1287)	RW	-	-	指定在分配的位图对象处于“关闭”(Off) 状态时，设置为“透明”(transparent) 的颜色。
TransparentColorPictureOn (页 1288)	RW	-	-	指定在分配的位图对象处于“打开”(On) 状态时，设置为“透明”(transparent) 的颜色。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有在激活设计中定义的底纹。
UseFirstGradient	-	-	-	-
UseSecondGradient	-	-	-	-
UseTransparentColorPictureOff (页 1367)	RW	-	-	指定在“TransparentColorPictureOff”属性中定义的透明色是否用于“关闭”(Off) 状态。
UseTransparentColorPictureOn (页 1368)	RW	-	-	指定在“TransparentColorPictureOff”属性中定义的透明色是否用于“关闭”(Off) 状态。

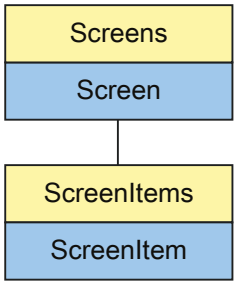
属性	RT Professional	RT Advanced	Panel RT	说明
UseTwoHandOperation	-	-	-	-
VerticalAlignment (页 1413)	RW	RW	RW	指定所选对象的文本垂直对齐。
VerticalPictureAlignment	-	-	-	-
Visible (页 1418)	RW	RW	RW	指定是否显示所选对象。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。
WindowsStyle (页 1440)	RW	-	-	指定是否以普通 Windows 样式显示对象。

表格 1-12　方法

方法	RT Professional	RT Advanced	Panel RT	说明
激活 (页 1470)	支持	支持	支持	启用永久区域或根画面。
ActivateDynamic (页 1473)	支持	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	支持	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

CameraControl

描述



表示“CameraControl”对象。CameraControl 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMICameraControl

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-13 属性

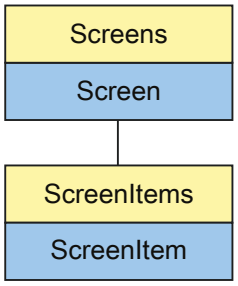
属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-
CameraUrl	-	-	-	-
CanBeGrouped	-	-	-	-
Height	-	R	R	指定高度。
Layer (页 913)	-	-	RW	在包含对象的画面中指定图层。
Left (页 920)	-	-	R	返回 X 坐标的值。
Location	-	-	-	-
MaintainAspectRatio	-	-	-	-
MaintainOriginalSize	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (页 1279)	-	-	R	返回 Y 坐标的值。
UseUdp	-	-	-	-
Visible (页 1418)	-	-	R	返回关于是否显示特定对象的信息。
Width	-	R	R	指定宽度。

表格 1-14 方法

方法	RT Professional	RT Advanced	面板 RT	描述
Activate (页 1470)	-	-	√	启用永久区域或根画面。

ChannelDiagnose

描述



表示“通道诊断显示”对象。ChannelDiagnose 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIChannelDiagnose

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-15　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-

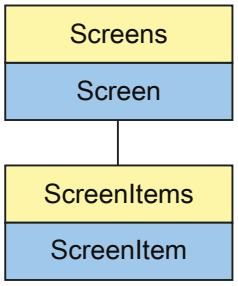
属性	RT Professional	RT Advanced	面板 RT	描述
CanBeGrouped	-	-	-	-
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。
Height (页 863)	RW	-	-	指定高度。
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (页 1279)	RW	-	-	指定 Y 坐标的值。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-16　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

CheckBox

说明



表示“Check box”对象。CheckBox 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMICheckBox

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-17　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AdaptBorder (页 606)	RW	-	-	指定对象的边框是否根据文本的大小进行动态调整。
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定操作权限。
BackColor (页 637)	RW	-	-	指定背景色。
BackFillStyle (页 643)	RW	-	-	指定填充图案。
BackFlashingColorOff (页 645)	RW	-	-	指定闪烁状态“关闭”(Off) 的背景色。
BackFlashingColorOn (页 646)	RW	-	-	指定闪烁状态“打开”(On) 的背景色。
BackFlashingEnabled (页 648)	RW	-	-	指定是否在运行系统中闪烁背景。
BackFlashingRate (页 649)	RW	-	-	指定背景的闪烁频率。
BorderBackColor (页 674)	RW	-	-	指定间断边框线的背景色。
BorderColor (页 678)	RW	-	-	指定线颜色。
BorderFlashingColorOff (页 683)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
BorderFlashingColorOn (页 685)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
BorderFlashingEnabled (页 687)	RW	-	-	指定是否在运行系统中闪烁对象的限值。
BorderFlashingRate (页 689)	RW	-	-	指定边框线的闪烁速率。
BorderStyle (页 695)	RW	-	-	指定边框线的闪烁类型。
BorderWidth (页 697)	RW	-	-	指定线宽。

属性	RT Professional	RT Advanced	面板 RT	描述
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CheckMarkAlignment (页 720)	RW	-	-	指定各域是否右对齐。
CheckMarkCount (页 720)	RW	-	-	指定字段数量。
CornerStyle (页 764)	RW	-	-	指定边角的形状。
DrawInsideFrame (页 787)	RW	-	-	指定要显示的线宽大于 1 边框线是在边框内描绘，还是与边框对称。
EdgeStyle (页 788)	RW	-	-	指定线样式。
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。
FillingDirection (页 825)	RW	-	-	指定填充方向。
FillPatternColor (页 821)	RW	-	-	指定填充样式的颜色。
Flashing	-	-	-	-
FlashingColorOff (页 832)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
FlashingColorOn (页 834)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
FlashingEnabled (页 836)	RW	-	-	指定是否为运行系统中的对象激活闪烁。
FlashingRate (页 839)	RW	-	-	指定边界线的闪烁频率。
Font	-	-	-	-
FontBold (页 849)	RW	-	-	指定文本是否以粗体显示。
FontItalic (页 849)	RW	-	-	指定文本是否以斜体显示。
FontName (页 851)	RW	-	-	指定字体。
FontSize (页 852)	RW	-	-	指定文本字体大小。
FontUnderline (页 853)	RW	-	-	指定文本是否应以下划线标出。
ForeColor (页 854)	RW	-	-	指定文本字体颜色。

属性	RT Professional	RT Advanced	面板 RT	描述
Height (页 863)	RW	-	-	指定高度。
HorizontalAlignment (页 881)	RW	-	-	指定文本水平对齐。
Index (页 887)	RW	-	-	指定所选文本域的索引。
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
Location	-	-	-	-
LogOperation (页 948)	RW	-	-	指定是否在操作对象后将报警输出到报警系统。
Name	-	-	-	-
ProcessValue (页 1037)	RW	-	-	指定要显示的数值的默认值。
RelativeFillLevel (页 1046)	RW	-	-	指定对象的填充百分比。
ShowBadTagState (页 1099)	RW	-	-	定义检测到不良质量代码或变量状态时对象是否变成灰色。
ShowFillLevel (页 1108)	RW	-	-	指定所选对象是否填充。
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Text (页 1185)	RW	-	-	指定文本域的标签。
TextHandles	-	-	-	-
TextOrientation (页 1189)	RW	-	-	指定文本方向。
Texts	-	-	-	-
ToolTipText (页 1278)	RW	-	-	指定工具提示文本。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。

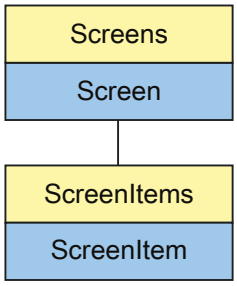
属性	RT Professional	RT Advanced	面板 RT	描述
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
VerticalAlignment (页 1413)	RW	-	-	指定所选对象中的文本垂直对齐方式。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-18　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	√	√	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

Circle

说明



表示“Circle”对象。Circle 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMICircle

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-19　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定所选对象在运行系统中的操作权限。
BackColor (页 637)	RW	RW	RW	指定背景色。
BackFillStyle (页 643)	RW	RW	RW	指定填充图案。
BorderBackColor (页 674)	RW	-	-	指定间断边框线的背景色。
BorderColor (页 678)	RW	RW	RW	指定线颜色。
BorderFlashingColorOff (页 683)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
BorderFlashingColorOn (页 685)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
BorderFlashingEnabled (页 687)	RW	-	-	指定是否在运行系统中闪烁对象的限值。
BorderFlashingRate (页 689)	RW	-	-	指定边界线的闪烁频率。
BorderStyle (页 695)	RW	-	-	指定边框线的闪烁类型。
BorderWidth (页 697)	RW	RW	RW	指定线宽。
CanBeGrouped	-	-	-	-
CornerStyle (页 764)	RW	-	-	指定边角的形状。
DeviceStyle	-	-	-	-
DrawInsideFrame (页 787)	RW	-	-	指定要显示的线宽大于 1 边框线是在边框内描绘，还是与边框对称。
EdgeStyle (页 788)	RW	RW	RW	指定线样式。
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。
FillingDirection (页 825)	RW	-	-	指定填充方向。

属性	RT Professional	RT Advanced	面板 RT	描述
FillPatternColor (页 821)	RW	-	-	指定填充样式的颜色。
Flashing	-	-	-	-
FlashingColorOff (页 832)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
FlashingColorOn (页 834)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
FlashingEnabled (页 836)	RW	-	-	指定是否为运行系统中的对象激活闪烁。
FlashingRate (页 839)	RW	-	-	指定边界线的闪烁频率。
Height (页 863)	RW	RW	RW	指定高度。
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
Location	-	-	-	-
Name	-	-	-	-
Radius (页 1040)	RW	RW	RW	指定半径。
RadiusHeight	-	-	-	-
RadiusWidth	-	-	-	-
RelativeFillLevel (页 1046)	RW	-	-	指定对象的填充百分比。
ShowFillLevel (页 1108)	RW	-	-	指定对象是否填充。
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (页 1278)	RW	RW	RW	指定工具提示文本。
Top (页 1279)	RW	RW	RW	指定 Y 坐标的值。

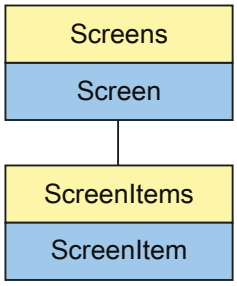
属性	RT Professional	RT Advanced	面板 RT	描述
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
Visible (页 1418)	RW	RW	RW	指定是否显示对象。
Width (页 1432)	RW	RW	RW	指定对象的宽度（以像素为单位）。

表格 1-20　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	√	√	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

CircleSegment

说明



表示“CircleSegment”对象。CircleSegment 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMICircleSegment

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-21　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定操作权限。
BackColor (页 637)	RW	-	-	指定背景色。
BackFillStyle (页 643)	RW	-	-	指定填充图案。
BorderBackColor (页 674)	RW	-	-	指定间断边框线的背景色。
BorderColor (页 678)	RW	-	-	指定线颜色。
BorderFlashingColorOff (页 683)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
BorderFlashingColorOn (页 685)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
BorderFlashingEnabled (页 687)	RW	-	-	指定是否在运行系统中闪烁对象的限值。
BorderFlashingRate (页 689)	RW	-	-	指定边框线的闪烁速率。
BorderStyle (页 695)	RW	-	-	指定边框线的闪烁类型。
BorderWidth (页 697)	RW	-	-	指定线宽。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (页 764)	RW	-	-	指定边角的形状。
DrawInsideFrame (页 787)	RW	-	-	指定要显示的线宽大于 1 边框线是在边框内描绘，还是与边框对称。
EdgeStyle (页 788)	RW	-	-	指定线样式。
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。
EndAngle (页 799)	RW	-	-	指定端点偏离零位置 (0°) 的角度。

属性	RT Professional	RT Advanced	面板 RT	描述
FillingDirection (页 825)	RW	-	-	指定填充方向。
FillPatternColor (页 821)	RW	-	-	指定填充样式的颜色。
Flashing	-	-	-	-
FlashingColorOff (页 832)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
FlashingColorOn (页 834)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
FlashingEnabled (页 836)	RW	-	-	指定是否为运行系统中的对象激活闪烁。
FlashingRate (页 839)	RW	-	-	指定边框线的闪烁速率。
Height (页 863)	RW	-	-	指定高度。
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
Location	-	-	-	-
Name	-	-	-	-
Radius (页 1040)	RW	-	-	指定半径。
RelativeFillLevel (页 1046)	RW	-	-	指定对象的填充百分比。
ShowFillLevel (页 1108)	RW	-	-	指定所选对象是否填充。
Size	-	-	-	-
StartAngle (页 1134)	RW	-	-	指定起始点偏离零位置 (0°) 的角度。
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (页 1278)	RW	-	-	指定工具提示文本。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。

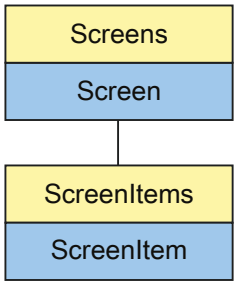
属性	RT Professional	RT Advanced	面板 RT	描述
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-22　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

CircularArc

说明



表示“CircularArc”对象。CircularArc 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMICircularArc

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-23　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定操作权限。
BackColor (页 637)	RW	-	-	指定背景色。
BorderStyle (页 695)	RW	-	-	指定边框线的闪烁类型。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Color (页 723)	RW	-	-	指定线颜色。
CornerStyle (页 764)	RW	-	-	指定边角的形状。
DrawInsideFrame (页 787)	RW	-	-	指定要显示的线宽大于 1 边框线是在边框内描绘，还是与边框对称。
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。
EndAngle (页 799)	RW	-	-	指定端点偏离零位置 (0°) 的角度。
Flashing	-	-	-	-
FlashingColorOff (页 832)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
FlashingColorOn (页 834)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
FlashingEnabled (页 836)	RW	-	-	指定是否为运行系统中的对象激活闪烁。
FlashingRate (页 839)	RW	-	-	指定边框线的闪烁速率。
Height (页 863)	RW	-	-	指定高度。
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
LineWidth (页 943)	RW	-	-	指定线宽。

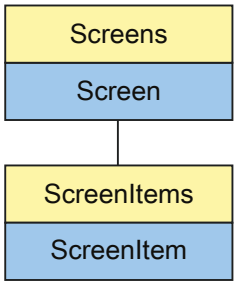
属性	RT Professional	RT Advanced	面板 RT	描述
Location	-	-	-	-
Name	-	-	-	-
Radius (页 1040)	RW	-	-	指定半径。
Size	-	-	-	-
StartAngle (页 1134)	RW	-	-	指定起始点偏离零位置 (0°) 的角度。
Style (页 1163)	RW	-	-	指定线样式。
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (页 1278)	RW	-	-	指定工具提示文本。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-24　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

Clock

说明



表示“Clock”对象。Clock 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

Clock

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-25　属性

属性	RT Professional	RT Advanced	Panel RT	说明
AllTagTypesAllowed	-	-	-	-
Analog (页 619)	RW	-	-	指定时钟是否显示为模拟时钟。

属性	RT Professional	RT Advanced	Panel RT	说明
BackFillStyle (页 643)	RW	-	-	指定所选对象的填充图案。
BorderColor	-	-	-	-
BorderBackColor	-	-	-	-
BorderWidth	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerRadius	-	-	-	-
DeviceStyle	-	-	-	-
DialColor (页 780)	RW	RW	RW	确定所选对象的对话框的颜色。
EdgeStyle	-	-	-	-
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。
Flashing (页 830)	RW	-	-	指定所选对象是否在运行系统中闪烁。
Font (页 846)	RW	-	-	指定字体。
Height (页 863)	RW	R	R	指定所选对象的高度。
HourNeedleHeight (页 885)	RW	RW	RW	指定“Clock”对象时针的长度。
HourNeedleWidth (页 886)	RW	RW	RW	指定“Clock”对象时针的宽度。
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定对象的 X 坐标的值。
Location	-	-	-	-
LockSquaredExtent (页 947)	RW	-	-	指定是否可以通过鼠标调节时钟大小。
MinuteNeedleHeight (页 980)	RW	RW	RW	指定“Clock”对象分针的长度。
MinuteNeedleWidth (页 981)	RW	RW	RW	指定“Clock”对象分针的宽度。
Name	RW	-	-	指定对象名称。
NeedleBorderColor (页 990)	RW	RW	RW	指定“Clock”对象指针的线颜色。

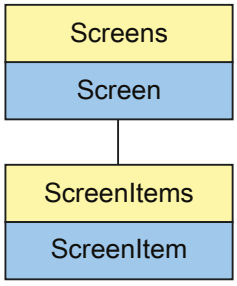
属性	RT Professional	RT Advanced	Panel RT	说明
NeedleColor (页 991)	RW	RW	RW	指定“Clock”对象的指针颜色。
NeedleFillStyle (页 992)	RW	-	-	指定指针显示为填充样式还是透明样式。
NumberStyle	-	-	-	-
Picture (页 1026)	RW	-	-	指定在运行系统的图形对象中显示的画面。
SecondNeedleHeight (页 1072)	RW	RW	RW	指定“Clock”对象秒针的长度。
SecondNeedleWidth (页 1072)	RW	RW	RW	指定“Clock”对象秒针的宽度。
ShowFocusRectangle (页 1109)	RW	-	-	指定按钮在运行系统中激活时，是否指定所选的边框。
ShowTicks (页 1123)	RW	RW	RW	指定是否以所选对象的刻度显示标记长度。
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TickDistance	-	-	-	-
TicksColor (页 1193)	RW	RW	RW	指定“Clock”对象盘面的小时标记的颜色。
TickStyle	-	-	-	-
Top (页 1279)	RW	RW	RW	指定所选对象的 Y 坐标的值。
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有在激活设计中定义的底纹。
Visible (页 1418)	RW	RW	RW	指定是否显示所选对象。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。

表格 1-26　方法

方法	RT Professional	RT Advanced	Panel RT	说明
激活 (页 1470)	支持	支持	支持	启用永久区域或根画面。
ActivateDynamic (页 1473)	支持	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	支持	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

ComboBox

说明



表示“Check box”对象。ComboBox 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIComboBox

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-27　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
AskOperationMotive (页 624)	RW	-	-	指定是否记录操作此对象的原因。
Authorization (页 626)	RW	-	-	指定操作权限。
BackColor (页 637)	RW	-	-	指定背景色。
BackFillStyle (页 643)	RW	-	-	指定填充图案。
BorderBackColor (页 674)	RW	-	-	指定间断边框线的背景色。
BorderColor (页 678)	RW	-	-	指定线颜色。
BorderStyle (页 695)	RW	-	-	指定边框线的闪烁类型。
BorderWidth (页 697)	RW	-	-	指定线宽。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (页 764)	RW	-	-	指定边角的形状。
CountVisibleItems (页 767)	RW	-	-	指定下拉列表中将包含的线数。
EdgeStyle (页 788)	RW	-	-	指定线样式。
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。
FillPatternColor	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
FitToLargest	-	-	-	-
Font	-	-	-	-
FontBold (页 849)	RW	-	-	指定文本是否以粗体显示。
FontItalic (页 849)	RW	-	-	指定文本是否以斜体显示。
FontName (页 851)	RW	-	-	指定字体。
FontSize (页 852)	RW	-	-	指定文本字体大小。
FontUnderline (页 853)	RW	-	-	指定文本是否应以下划线标出。
ForeColor (页 854)	RW	-	-	指定文本字体颜色。
Height (页 863)	RW	-	-	指定高度。
HorizontalAlignment (页 881)	RW	-	-	指定文本水平对齐。
Index (页 887)	RW	-	-	指定所选文本域的索引。
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
Location	-	-	-	-
LogOperation (页 948)	RW	-	-	指定是否在操作对象后将报警输出到报警系统。
Name	-	-	-	-
SelectedIndex (页 1078)	RW	-	-	指定其关联文本在组合框或列表框中高亮显示的索引。
SelectedText (页 1081)	RW	-	-	指定使用“SelectedIndex”引用的条目的文本。
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Text (页 1185)	RW	-	-	指定文本域的标签。
TextHandles	-	-	-	-
Texts	-	-	-	-

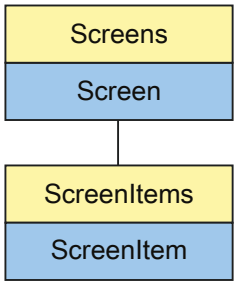
属性	RT Professional	RT Advanced	面板 RT	描述
ToolTipText (页 1278)	RW	-	-	指定工具提示文本。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-28　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

Connector

描述



表示“Connector”对象。Connector 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIConnector

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-29　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定操作权限。
BackColor (页 637)	RW	-	-	指定背景色。
BorderEndStyle (页 683)	RW	-	-	指定所选对象线端的类型。
BorderStyle (页 695)	RW	-	-	指定边框线的闪烁类型。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Color (页 723)	RW	-	-	指定线颜色。
ConnectionType (页 759)	RW	-	-	指定连接器类型。
CornerStyle (页 764)	RW	-	-	指定边角的形状。
DeviceStyle	-	-	-	-
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。
EndStyle (页 800)	RW	-	-	指定如何显示线端。
FirstConnectedObject	-	-	-	-
FirstConnectedObjectIndex (页 827)	RW	-	-	指定连接器上部点的索引值。
FirstConnectedObjectName (页 827)	RW	-	-	指定连接于连接器上部点的对象的名称。
Flashing	-	-	-	-
FlashingColorOff (页 832)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
FlashingColorOn (页 834)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。

属性	RT Professional	RT Advanced	面板 RT	描述
FlashingEnabled (页 836)	RW	-	-	指定是否为运行系统中的对象激活闪烁。
FlashingRate (页 839)	RW	-	-	指定边界线的闪烁频率。
Height (页 863)	RW	-	-	指定高度。
LastConnectedObject	-	-	-	-
LastConnectedObjectIndex (页 910)	RW	-	-	指定最后一个连接对象的连接点的索引值。
LastConnectedObjectName (页 911)	RW	-	-	指定连接于连接器下部点的对象的名称。
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
LineWidth (页 943)	RW	-	-	指定线宽。
Location	-	-	-	-
Name	-	-	-	-
NoHitTest	-	-	-	-
Points	-	-	-	-
PointsCount	-	-	-	-
Size	-	-	-	-
StartStyle (页 1135)	RW	-	-	指定如何显示直线起始端。
Style (页 1163)	RW	-	-	指定线样式。
SwapFirstWithLastConnection (页 1165)	RW	-	-	指定应交换第一个连接和最后一个连接。
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (页 1278)	RW	-	-	指定工具提示文本。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。

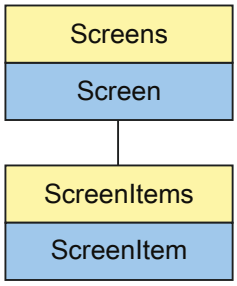
属性	RT Professional	RT Advanced	面板 RT	描述
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-30　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

DateTimeField

说明



表示“Date/time field”对象。DateTimeField 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIDateTimeField

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-31　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	-	RW	RW	指定操作权限。
BackColor (页 637)	-	RW	RW	指定背景色。
BackFillStyle	-	-	-	-
BorderBackColor	-	-	-	-
BorderColor (页 678)	-	RW	RW	指定线颜色。
BorderWidth	-	-	-	-
BottomMargin	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerRadius	-	-	-	-
DeviceStyle	-	-	-	-
DisplayCentury	-	-	-	-
DisplaySystemTime	-	-	-	-
EdgeStyle	-	-	-	-
Enabled (页 792)	-	RW	RW	指定是否可以在运行系统中操作所选对象。
FillPatternColor	-	-	-	-
FitToLargest	-	-	-	-
Flashing	-	-	-	-
Font	-	-	-	-
ForeColor (页 854)	-	RW	RW	指定文本字体颜色。
Format	-	-	-	-
Height	-	R	R	指定高度。
HelpText (页 868)	-	RW	RW	指定作为特定对象的用户帮助显示在运行系统中的工具提示。

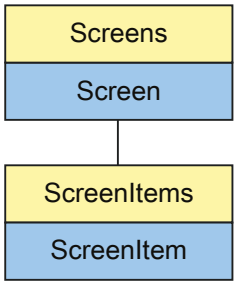
属性	RT Professional	RT Advanced	面板 RT	描述
HorizontalAlignment (页 881)	-	RW	RW	指定文本水平对齐。
Layer (页 913)	-	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	-	RW	RW	指定 X 坐标的值。
LeftMargin	-	-	-	-
LineWrap	-	-	-	-
Location	-	-	-	-
LongDateTimeFormat	-	-	-	-
Mode	-	-	-	-
Name	-	-	-	-
ProcessValue	-	-	-	-
RightMargin	-	-	-	-
ShowDate	-	-	-	-
ShowTime	-	-	-	-
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TextOrientation	-	-	-	-
TimeDisplayMode	-	-	-	-
Top (页 1279)	-	RW	RW	指定 Y 坐标的值。
TopMargin	-	-	-	-
UseDesignColorSchema	-	-	-	-
UseTwoHandOperation	-	-	-	-
VerticalAlignment (页 1413)	-	RW	RW	指定所选对象中的文本垂直对齐方式。
Visible (页 1418)	-	RW	RW	指定是否显示所选对象。
Width	-	R	R	指定宽度。

表格 1-32　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	√	√	启用永久区域或根画面。

DiskSpaceView

说明



表示 “存储空间视图”(Memory space view) 对象。DiskSpaceView 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

IXDiskSpaceView

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

属性	RT Professional	RT Advanced	面板 RT	描述
Alarm	-	-	-	-
AlarmColor	-	-	-	-
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Drive	-	-	-	-
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。
Free	-	-	-	-
FreePercent (页 858)	R	-	-	返回的可用磁盘空间的测量值为百分数。
Height (页 863)	RW	-	-	指定高度。
Interval (页 896)	R	-	-	返回更新所显示测量值的时间间隔。
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
Location	-	-	-	-
Name (页 985)	R	-	-	返回名称。
NormalColor (页 993)	R	-	-	返回内存可用性无关紧要的情况下存储空间视图中所用的存储空间颜色。
Size	-	-	-	-
TabIndex (页 1167)	R	-	-	返回使用 Tab 键在各个对象之间进行切换时对象在顺序中所处的位置。
TabIndexAlpha (页 1167)	R	-	-	返回在 Alpha 光标模式下使用 Tab 键在各个对象之间进行切换时对象在顺序中所处的位置。
Tolerance (页 1244)	R	-	-	返回存储空间视图的限制，超出该限制后将报告偏差。
ToleranceColor (页 1245)	R	-	-	返回超过容差范围时存储空间视图滚动条的显示颜色。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。
Total (页 1283)	R	-	-	返回存储容量。
Used (页 1353)	R	-	-	返回使用的存储空间量。

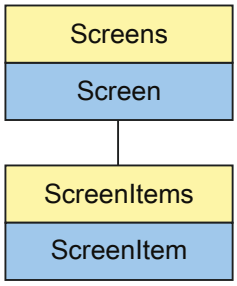
属性	RT Professional	RT Advanced	面板 RT	描述
UsedPercent (页 1358)	R	-	-	返回占用存储空间百分比形式的测量值。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Warning (页 1422)	R	-	-	返回占用的存储空间百分比限值，超出此值将生成警告。
WarningColor (页 1423)	R	-	-	返回达到警告范围时存储空间视图滚动条的显示颜色。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-33　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

Ellipse

说明



表示“Ellipse”对象。Ellipse 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIEllipse

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-34　属性

属性	RT Professional	RT Advanced	Panel RT	说明
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定所选对象在运行系统中的操作权限。
BackColor (页 637)	RW	RW	RW	指定所选对象的背景色。
BackFillStyle (页 643)	RW	RW	RW	指定所选对象的填充图案。
BorderBackColor (页 674)	RW	-	-	指定所选对象间断边框线的背景色。
BorderColor (页 678)	RW	RW	RW	指定所选对象的线颜色。
BorderFlashingColorOff (页 683)	RW	-	-	指定闪烁状态“关闭”(Off) 的所选对象的边框线颜色。
BorderFlashingColorOn (页 685)	RW	-	-	指定闪烁状态“打开”(On) 的所选对象的边框线颜色。
BorderFlashingEnabled (页 687)	RW	-	-	指定所选对象的边框线是否在运行系统中闪烁。
BorderFlashingRate (页 689)	RW	-	-	指定所选对象的边框线的闪烁频率。
BorderStyle (页 695)	RW	-	-	指定所选对象边框线的类型。
BorderWidth (页 697)	RW	RW	RW	指定所选对象的线宽。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (页 764)	RW	-	-	指定特定对象的角形状。
DeviceStyle	-	-	-	-
DrawInsideFrame (页 787)	RW	-	-	指定线宽大于 1 的所选对象的边框线是在边框内描绘，还是与边框对称。
EdgeStyle (页 788)	RW	RW	RW	指定所选对象的线样式。
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。

属性	RT Professional	RT Advanced	Panel RT	说明
FillingDirection (页 825)	RW	-	-	指定填充方向。
FillPatternColor (页 821)	RW	-	-	指定所选对象的填充样式的颜色。
Flashing	-	-	-	-
FlashingColorOff (页 832)	RW	-	-	指定闪烁状态“关闭”(Off) 的所选对象的边框线颜色。
FlashingColorOn (页 834)	RW	-	-	指定闪烁状态“打开”(On) 的所选对象的边框线颜色。
FlashingEnabled (页 836)	RW	-	-	指定所选对象的边框线是否在运行系统中闪烁。
FlashingRate (页 839)	RW	-	-	指定所选对象的边框线的闪烁频率。
Height (页 863)	RW	RW	RW	指定所选对象的高度。
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定所选对象的 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
Location	-	-	-	-
Name	-	-	-	-
RadiusHeight (页 1041)	RW	RW	RW	指定特定对象的副轴。
RadiusWidth (页 1042)	RW	RW	RW	指定特定对象的主轴。
RelativeFillLevel (页 1046)	RW	-	-	指定对象的填充百分比。
ShowFillLevel (页 1108)	RW	-	-	指定所选对象是否填充。
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-

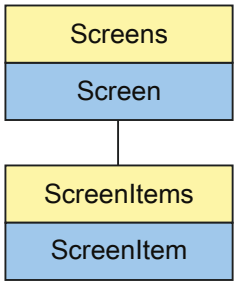
属性	RT Professional	RT Advanced	Panel RT	说明
ToolTipText (页 1278)	RW	RW	RW	指定工具提示文本。
Top (页 1279)	RW	RW	RW	指定所选对象的 Y 坐标的值。
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有在激活设计中定义的底纹。
Visible (页 1418)	RW	RW	RW	指定是否显示所选对象。
Width (页 1432)	RW	RW	RW	指定对象的宽度（以像素为单位）。

表格 1-35　方法

方法	RT Professional	RT Advanced	Panel RT	说明
激活 (页 1470)	支持	支持	支持	启用永久区域或根画面。
ActivateDynamic (页 1473)	支持	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	支持	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

EllipseSegment

说明



表示“EllipseSegment”对象。EllipseSegment 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIEllipseSegment

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-36　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定操作权限。
BackColor (页 637)	RW	-	-	指定背景色。
BackFillStyle (页 643)	RW	-	-	指定填充图案。
BorderBackColor (页 674)	RW	-	-	指定间断边框线的背景色。
BorderColor (页 678)	RW	-	-	指定线颜色。
BorderFlashingColorOff (页 683)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
BorderFlashingColorOn (页 685)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
BorderFlashingEnabled (页 687)	RW	-	-	指定是否在运行系统中闪烁对象的限值。
BorderFlashingRate (页 689)	RW	-	-	指定边界线的闪烁频率。
BorderStyle (页 695)	RW	-	-	指定边框线类型。
BorderWidth (页 697)	RW	-	-	指定线宽。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (页 764)	RW	-	-	指定角形状。
DrawInsideFrame (页 787)	RW	-	-	指定线宽大于 1 的边框线是显示在边框内，还是与边框对称。
EdgeStyle (页 788)	RW	-	-	指定线样式。
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。
EndAngle (页 799)	RW	-	-	指定端点偏离零位置 (0°) 的角度。

属性	RT Professional	RT Advanced	面板 RT	描述
FillingDirection (页 825)	RW	-	-	指定填充方向。
FillPatternColor (页 821)	RW	-	-	指定填充图案的颜色。
Flashing	-	-	-	-
FlashingColorOff (页 832)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
FlashingColorOn (页 834)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
FlashingEnabled (页 836)	RW	-	-	指定是否为运行系统中的对象激活闪烁。
FlashingRate (页 839)	RW	-	-	指定边界线的闪烁频率。
Height (页 863)	RW	-	-	指定高度。
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
Location	-	-	-	-
Name	-	-	-	-
RadiusHeight (页 1041)	RW	-	-	指定副轴。
RadiusWidth (页 1042)	RW	-	-	指定主轴。
RelativeFillLevel (页 1046)	RW	-	-	指定对象的填充百分比。
ShowFillLevel (页 1108)	RW	-	-	指定所选对象是否填充。
Size	-	-	-	-
StartAngle (页 1134)	RW	-	-	指定起始点偏离零位置 (0°) 的角度。
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-

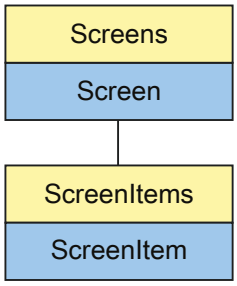
属性	RT Professional	RT Advanced	面板 RT	描述
ToolTipText (页 1278)	RW	-	-	指定工具提示文本。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-37　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	√	√	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

EllipticalArc

说明



表示“EllipticalArc”对象。EllipticalArc 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIEllipticalArc

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-38　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定操作权限。
BackColor (页 637)	RW	-	-	指定背景色。
BorderStyle (页 695)	RW	-	-	指定边框线类型。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Color (页 723)	RW	-	-	指定线颜色。
CornerStyle (页 764)	RW	-	-	指定角形状。
DrawInsideFrame (页 787)	RW	-	-	指定线宽大于 1 的边框线是显示在边框内，还是与边框对称。
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。
EndAngle (页 799)	RW	-	-	指定端点偏离零位置 (0°) 的角度。
Flashing	-	-	-	-
FlashingColorOff (页 832)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
FlashingColorOn (页 834)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
FlashingEnabled (页 836)	RW	-	-	指定是否为运行系统中的对象激活闪烁。
FlashingRate (页 839)	RW	-	-	指定边界线的闪烁频率。
Height (页 863)	RW	-	-	指定高度。
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
LineWidth (页 943)	RW	-	-	指定线宽。

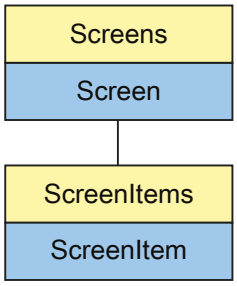
属性	RT Professional	RT Advanced	面板 RT	描述
Location	-	-	-	-
Name	-	-	-	-
RadiusHeight (页 1041)	RW	-	-	指定副轴。
RadiusWidth (页 1042)	RW	-	-	指定主轴。
Size	-	-	-	-
StartAngle (页 1134)	RW	-	-	指定起始点偏离零位置 (0°) 的角度。
Style (页 1163)	RW	-	-	指定线样式。
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (页 1278)	RW	-	-	指定工具提示文本。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-39　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

FunctionTrendControl

说明



表示"f(x)FunctionTrendView"对象。FunctionTrendControl 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIFunctionTrendControl

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-40　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
ApplyProjectSettingsForDesignMode	-	-	-	-
AvailableStatusbarElements	-	-	-	-
AvailableToolbarButtons	-	-	-	-
BackColor (页 637)	RW	-	-	指定背景色。
BorderColor (页 678)	RW	-	-	指定线颜色。
BorderWidth (页 697)	RW	-	-	指定线宽。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Caption (页 708)	RW	-	-	指定标题中将要显示的文本。
Closeable (页 722)	RW	-	-	
ConnectTrendWindows (页 760)	RW	-	-	指定是否连接已组态的趋势视图。
ControlDesignMode (页 762)	RW	-	-	指定控制设计。
Enabled (页 792)	-	RW	RW	指定是否可以在运行系统中操作所选对象。
ExportDelimiter	-	-	-	-
ExportDirectoryChangeable (页 808)	RW	-	-	指定是否可以在运行系统中更改数据导出目录。
ExportDirectoryname (页 809)	RW	-	-	指定导出的运行系统数据的目标目录。
ExportFileExtension (页 810)	RW	-	-	指定导出文件的文件扩展名。
ExportFilename (页 811)	RW	-	-	指定导出的运行系统数据的目标文件名称。

属性	RT Professional	RT Advanced	面板 RT	描述
ExportFilenameChangeable (页 812)	RW	-	-	指定是否可以在运行系统中更改导出文件名。
ExportFormat	-	-	-	-
ExportFormatGuid (页 813)	RW	-	-	指定 ID 编号和导出数据源的分配。
ExportFormatName (页 814)	RW	-	-	指定导出文件格式。
ExportParameters (页 815)	RW	-	-	通过属性对话框指定所选格式的参数。
ExportSelection (页 816)	RW	-	-	指定导出控件的哪些运行系统数据。
ExportShowDialog (页 817)	RW	-	-	指定是否在运行系统中显示数据导出对话框。
Font (页 846)	RW	-	-	指定字体。
GraphDirection (页 859)	RW	-	-	指定当前值在趋势视图的哪个边框上显示。
Height (页 863)	RW	R	R	指定高度。
Layer (页 913)	-	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定 X 坐标的值。
LineColor (页 939)	RW	-	-	指定窗口分隔线的颜色。
LineWidth (页 943)	RW	-	-	指定线宽。
LoadDataImmediately (页 946)	RW	-	-	指定是否下载日志中记录的在调用画面时显示的时间范围的变量值。
Location	-	-	-	-
Moveable (页 983)	RW	-	-	指定关于运行系统中是否可以移动对象的信息。
Name	-	-	-	-
Object	-	-	-	-
OcxGuid	-	-	-	-
OcxState	-	-	-	-
OcxStateForEs2Rt	-	-	-	-
Online (页 999)	RW	-	-	指定更新的起始与停止。

属性	RT Professional	RT Advanced	面板 RT	描述
PrintJob (页 1036)	RW	-	-	指定一个在“报表”编辑器中创建的打印作业。
RTPersistence (页 1054)	RW	-	-	指定在画面更改后是否保留在线组态。
RTPersistenceAuthorization (页 1055)	RW	-	-	指定运行系统中在线组态所需的权限。
RTPersistenceType (页 1057)	RW	-	-	指定如何保留 WinCC 的在线组态。
ShowRuler (页 1114)	RW	-	-	指定是否为对象“OnlineTrendControl”的轴标签显示刻度分度（帮助线）。
ShowRulerInAxis (页 1114)	RW	-	-	指定是否在时间轴中显示标尺。
ShowScrollbars (页 1116)	RW	-	-	指定是否显示滚动条。
ShowTitle (页 1124)	RW	-	-	为对象指定窗口边框和窗口标题的样式。
ShowTrendIcon (页 1126)	RW	-	-	指定是否在数值轴下显示图标。
Size	-	-	-	-
Sizeable (页 1129)	RW	-	-	指定可以在运行系统中更改对象的大小。
StatusbarBackColor (页 1138)	RW	-	-	指定状态栏的背景色。
StatusbarElementAdd (页 1139)	RW	-	-	创建一个新的用户自定义状态栏元素。
StatusbarElementAutoSize (页 1140)	RW	-	-	指定是否自动设置引用“StatusbarElementIndex”的状态栏元素的宽度。
StatusbarElementCount (页 1141)	RW	-	-	指定已组态的状态栏元素的数量。
StatusbarElementIconId (页 1142)	RW	-	-	使用其图标 ID 来引用状态栏元素。
StatusbarElementID (页 1143)	RW	-	-	使用其元素 ID 来引用状态栏元素。

属性	RT Professional	RT Advanced	面板 RT	描述
StatusbarElementIndex (页 1144)	RW	-	-	引用状态栏元素。
StatusbarElementName (页 1145)	RW	-	-	指定使用“StatusbarElementIndex”引用的状态栏元素的名称。
StatusbarElementRemove (页 1146)	RW	-	-	删除引用其名称的用户自定义状态栏元素。
StatusbarElementRename (页 1147)	RW	-	-	指定使用“StatusbarElementIndex”引用的用户自定义的状态栏元素的新名称。
StatusbarElementRepos (页 1148)	RW	-	-	指定对象状态栏中使用“StatusbarElementIndex”引用的元素位置。
StatusbarElements	-	-	-	-
StatusbarElementText (页 1149)	RW	-	-	指定使用“StatusbarElementIndex”引用的用户自定义的状态栏元素的文本。
StatusbarElementTooltipText (页 1150)	RW	-	-	指定使用“StatusbarElementIndex”引用的用户自定义的状态栏元素的工具提示文本。
StatusbarElementUserDefined (页 1151)	RW	-	-	指定组态工程师是否已添加使用“StatusbarElementIndex”引用的状态栏元素作为新的用户自定义元素。
StatusbarElementVisible (页 1152)	RW	-	-	指定是否在对象中显示使用“StatusbarElementIndex”引用的状态栏元素。
StatusbarElementWidth (页 1153)	RW	-	-	指定使用“StatusbarElementIndex”引用的状态栏元素的宽度（以像素为单位）。
StatusbarFont (页 1154)	RW	-	-	指定状态栏中文本的字体。
StatusbarFontColor (页 1155)	RW	-	-	指定状态栏中文本的颜色。
StatusbarShowTooltips (页 1156)	RW	-	-	指定是否在运行系统中显示状态栏元素的工具提示。
StatusbarText (页 1157)	RW	-	-	指定状态栏中的默认文本。
StatusbarUseBackColor (页 1158)	RW	-	-	指定是否显示状态栏的背景色。

属性	RT Professional	RT Advanced	面板 RT	描述
StatusbarVisible (页 1159)	RW	-	-	指定是否显示控件的状态栏。
SupportsUserDefinedToolbarButtons	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TimeBase (页 1211)	RW	-	-	指定显示时间值的时区。
ToolbarAlignment (页 1252)	RW	-	-	指定工具栏的位置。
ToolbarBackColor (页 1253)	RW	-	-	指定工具栏的背景色。
ToolbarButtonActive (页 1254)	RW	-	-	指定是否在运行系统中激活按钮的相应功能。
ToolbarButtonAdd (页 1255)	RW	-	-	在对象的工具栏中创建一个新的用户自定义按钮。
ToolbarButtonAuthorization (页 1256)	RW	-	-	指定所选按键功能的权限。
ToolbarButtonBeginGroup (页 1257)	RW	-	-	在工具栏上，为所选按键功能插入前导分隔符（垂直线）。
ToolbarButtonClick (页 1258)	RW	-	-	单击工具栏按钮。
ToolbarButtonCount (页 1259)	RW	-	-	指定工具栏中的已组态按钮数目。
ToolbarButtonEnabled (页 1260)	RW	-	-	指定是否可以操作用户自定义工具栏按钮。
ToolbarButtonHotKey (页 1261)	RW	-	-	指定所选对象按钮的热键。
ToolbarButtonID (页 1262)	RW	-	-	使用其 ID 引用按钮。
ToolbarButtonIndex (页 1263)	RW	-	-	引用按钮。

属性	RT Professional	RT Advanced	面板 RT	描述
ToolbarButtonLocked (页 1264)	RW	-	-	指定是否显示使用"ToolbarButtonIndex"引用的用户自定义按钮的锁定、按下状态。
ToolbarButtonName (页 1265)	RW	-	-	指定使用"ToolbarButtonIndex"引用的用户自定义按钮的名称。
ToolbarButtonRemove (页 1266)	RW	-	-	移除一个引用了其名称的用户自定义按钮。
ToolbarButtonRename (页 1267)	RW	-	-	指定使用"ToolbarButtonIndex"引用的用户自定义按钮的新名称。
ToolbarButtonRepos (页 1268)	RW	-	-	在对象的工具栏中指定使用"ToolbarButtonIndex"引用的按钮位置。
ToolbarButtons	-	-	-	-
ToolbarButtonTooltipText (页 1270)	RW	-	-	指定工具栏中用户自定义按钮的工具提示文本。
ToolbarButtonUserDefined (页 1271)	RW	-	-	指定项目工程师是否已将此工具栏按钮添加为新的用户自定义按钮。
ToolbarButtonVisible (页 1272)	RW	-	-	指定是否在工具栏中显示使用"ToolbarButtonIndex"引用的按钮。
ToolbarShowTooltips (页 1273)	RW	-	-	指定是否在运行系统中显示按键功能的工具提示。
ToolbarUseBackColor (页 1275)	RW	-	-	指定是否显示工具栏的背景色。
ToolbarUseHotKeys (页 1276)	RW	-	-	指定是否激活工具栏中按钮的热键。
ToolbarVisible (页 1277)	RW	-	-	指定是否显示控件的工具栏。
Top (页 1279)	-	R	R	返回 Y 坐标的值。
TrendActualize (页 1289)	RW	-	-	指定是否更新所选趋势。
TrendAdd (页 1290)	RW	-	-	创建新趋势。
TrendBeginTime (页 1293)	RW	-	-	定义用于将数据传送到所选趋势的时间范围的开始时间。
TrendColor (页 1294)	RW	-	-	指定趋势视图中引用的趋势的颜色。

属性	RT Professional	RT Advanced	面板 RT	描述
TrendCount (页 1295)	RW	-	-	指定所组态趋势的数目。
TrendEndTime (页 1295)	RW	-	-	定义用于所选趋势数据连接的时间范围的结束时间。
TrendExtendedColorSet (页 1296)	RW	-	-	指定是否显示趋势的点和填充颜色。
TrendFill (页 1297)	RW	-	-	指定趋势下方的区域是否显示为已填充。
TrendFillColor (页 1298)	RW	-	-	定义趋势的填充颜色。
TrendIndex (页 1299)	RW	-	-	引用一个已组态的趋势。
TrendLabel (页 1300)	RW	-	-	为使用"TrendIndex"引用的趋势定义标签文本。
TrendLineStyle (页 1301)	RW	-	-	指定用于趋势显示的线类型。
TrendLineType (页 1302)	RW	-	-	指定如何显示趋势。
TrendLineWidth (页 1303)	RW	-	-	指定所选趋势的线宽（以像素为单位）。
TrendLowerLimit (页 1303)	RW	-	-	指定在对象中作为趋势显示的变量下限。
TrendLowerLimitColor (页 1304)	RW	-	-	指定用于标记低于"TrendLowerLimit"值的趋势值的颜色。
TrendLowerLimitColoring (页 1305)	RW	-	-	指定是否使用"TrendLowerLimitColor"标识小于"TrendLowerLimit"值的变量值。
TrendMeasurePoints (页 1305)	RW	-	-	指定用于显示选定趋势的测量点的数量。
TrendName (页 1306)	RW	-	-	指定使用"TrendIndex"引用的趋势的名称。
TrendPointColor (页 1307)	RW	-	-	指定引用趋势上的点的颜色。
TrendPointStyle (页 1308)	RW	-	-	指定如何显示趋势点。
TrendPointWidth (页 1309)	RW	-	-	指定点宽度（以像素为单位）。

属性	RT Professional	RT Advanced	面板 RT	描述
TrendProvider (页 1309)	RW	-	-	指定所选趋势的数据源。
TrendProviderCLSID (页 1310)	RW	-	-	指定趋势中数据源的 CLSID。
TrendRangeType (页 1311)	RW	-	-	指定为趋势提供数据的时间范围。
TrendRemove (页 1312)	RW	-	-	移除一个引用了其名称的趋势。
TrendRename (页 1313)	RW	-	-	指定使用“TrendIndex”引用的趋势的新名称。
TrendRepos (页 1313)	RW	-	-	在对象的趋势窗口中指定使用“TrendIndex”引用的趋势位置。
Trends	-	-	-	-
TrendSelectTagNameX (页 1315)	RW	-	-	指定用于选择 x 轴数据源的变量名称的对话框在运行系统中初次显示。
TrendSelectTagNameY (页 1315)	RW	-	-	指定用于选择 Y 轴数据源的变量名称的对话框在运行系统中初次显示。
TrendTagNameX (页 1317)	RW	-	-	指定 X 轴连接的 HMI 变量或列名称。
TrendTagNameY (页 1317)	RW	-	-	指定 Y 轴连接的 HMI 变量或列名称。
TrendTimeRangeBase (页 1319)	RW	-	-	指定用于计算时间范围的时间单位。
TrendTimeRangeFactor (页 1319)	RW	-	-	指定确定为其显示引用趋势的时间间隔的因数。
TrendTrendWindow (页 1320)	RW	-	-	指定显示所选趋势的趋势窗口。
TrendUncertainColor (页 1321)	RW	-	-	指定不确定状态的值的颜色。
TrendUncertainColoring (页 1321)	RW	-	-	指定不确定状态的值将突出显示。
TrendUpperLimit (页 1322)	RW	-	-	指定在特定对象中作为趋势显示的变量上限。

属性	RT Professional	RT Advanced	面板 RT	描述
TrendUpperLimitColor (页 1323)	RW	-	-	指定用于标记高于"TrendUpperLimit"值的趋势值的颜色。
TrendUpperLimitColoring (页 1324)	RW	-	-	指定是否使用系统定义的颜色显示选择框架。
TrendVisible (页 1326)	RW	-	-	指定是否在对象中显示使用"TrendIndex"引用的趋势。
TrendWindowAdd (页 1327)	RW	-	-	创建新的趋势视图。
TrendWindowCoarseGrid (页 1327)	RW	-	-	指定是否显示主刻度的网格线。
TrendWindowCoarseGridColor (页 1328)	RW	-	-	指定对象中引用图的主网格的颜色。
TrendWindowCount (页 1329)	RW	-	-	在趋势视图中指定所组态的趋势图。
TrendWindowFineGrid (页 1330)	RW	-	-	指定是否对次刻度显示网格线。
TrendWindowFineGridColor (页 1330)	RW	-	-	指定对象中引用图的辅助网格的颜色。
TrendWindowForegroundTrendGrid (页 1331)	RW	-	-	指定是否在所选趋势窗口中仅显示前景趋势的网格线。
TrendWindowGridInTrendColor (页 1332)	RW	-	-	指定是否使用趋势颜色显示主刻度的网格线。
TrendWindowHorizontalGrid (页 1333)	RW	-	-	指定是否显示水平网格线。
TrendWindowIndex (页 1333)	RW	-	-	引用趋势视图。
TrendWindowName (页 1334)	RW	-	-	指定使用"TrendWindowIndex"引用的趋势视图的名称。
TrendWindowRemove (页 1335)	RW	-	-	移除一个引用了其名称的趋势视图。
TrendWindowRename (页 1335)	RW	-	-	指定使用"TrendWindowIndex"引用的趋势视图的新名称。

属性	RT Professional	RT Advanced	面板 RT	描述
TrendWindowRepos (页 1336)	RW	-	-	指定使用“TrendWindowIndex”引用的趋势视图的位置。
TrendWindowRulerColor (页 1337)	RW	-	-	指定标尺的颜色。
TrendWindowRulerLayer (页 1338)	RW	-	-	指定趋势视图中标尺的显示层。
TrendWindowRulerStyle (页 1339)	RW	-	-	指定标尺的外观。
TrendWindowRulerWidth (页 1339)	RW	-	-	指定标尺的宽度（以像素为单位）。
TrendWindows	-	-	-	-
TrendWindowSpacePortion (页 1340)	RW	-	-	在对象的图表区域中，指定引用图的范围比例。
TrendWindowVerticalGrid (页 1343)	RW	-	-	指定是否显示垂直网格线。
TrendWindowVisible (页 1344)	RW	-	-	指定是否显示使用“TrendWindowIndex”引用的趋势视图。
TrendXAxis (页 1345)	RW	-	-	定义要用于所引用趋势的 X 轴。
TrendYAxis (页 1345)	RW	-	-	定义要用于所引用趋势的 y 轴。
UseTrendNameAsLabel (页 1369)	RW	-	-	指定使用 “名称”(Name) 还是 “标签”(Label) 属性作为运行系统中趋势的标识。
Visible (页 1418)	-	R	R	指定是否显示所选对象。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。
XAxes	-	-	-	-
XAxisAdd (页 1442)	RW	-	-	创建新的 X 轴。
XAxisAlignment (页 1442)	RW	-	-	指定使用“XAxisIndex”引用的 X 轴的对齐方式。
XAxisAutoPrecisions (页 1443)	RW	-	-	指定是否自动确定使用“XAxisIndex”引用的 X 轴的小数位数。
XAxisAutoRange (页 1444)	RW	-	-	指定是否自动计算使用“XAxisIndex”引用的 X 轴的取值范围。

属性	RT Professional	RT Advanced	面板 RT	描述
XAxisBeginValue (页 1444)	RW	-	-	指定使用"XAxisIndex"引用的 X 轴的取值范围下限。
XAxisColor (页 1445)	RW	-	-	指定使用"XAxisIndex"引用的 X 轴的颜色。
XAxisCount (页 1446)	RW	-	-	指定已组态的 X 轴的数目。
XAxisEndValue (页 1447)	RW	-	-	指定使用"XAxisIndex"引用的 X 轴的取值范围上限。
XAxisExponentialFormat (页 1447)	RW	-	-	指定是否在指数计数法中显示使用"XAxisIndex"引用的 X 轴的值。
XAxisIndex (页 1448)	RW	-	-	引用 X 轴。
XAxisInTrendColor (页 1449)	RW	-	-	指定使用"XAxisIndex"引用的轴的颜色是否与趋势颜色相对应。
XAxisLabel (页 1449)	RW	-	-	指定使用"XAxisIndex"引用的 X 轴的标签文本。
XAxisName (页 1450)	RW	-	-	指定使用"XAxisIndex"引用的轴的名称。
XAxisPrecisions (页 1451)	RW	-	-	指定使用"XAxisIndex"引用的 X 轴的值显示的小数位数。
XAxisRemove (页 1451)	RW	-	-	移除一个引用了其名称的 X 轴。
XAxisRename (页 1452)	RW	-	-	指定使用"XAxisIndex"引用的轴的新名称。
XAxisRepos (页 1452)	RW	-	-	对于多个 X 轴，指定使用"XAxisIndex"引用的 X 轴的位置。
XAxisScalingType (页 1453)	RW	-	-	指定使用"XAxisIndex"引用的 X 轴比例缩放的类型。
XAxisTrendWindow (页 1454)	RW	-	-	指定使用"XAxisIndex"引用的轴在其中显示的趋势图。
XAxisVisible (页 1455)	RW	-	-	指定是否在对象中显示使用"XAxisIndex"引用的 X 轴。
YAxes	-	-	-	-
YAxisAdd (页 1455)	RW	-	-	创建新的 Y 轴。
YAxisAlignment (页 1456)	RW	-	-	指定使用"YAxisIndex"引用的 Y 轴的对齐方式。

属性	RT Professional	RT Advanced	面板 RT	描述
YAxisAutoPrecisions (页 1457)	RW	-	-	指定是否自动指定使用"YAxisIndex"引用的Y轴的小数位数。
YAxisAutoRange (页 1457)	RW	-	-	指定是否自动计算使用"YAxisIndex"引用的Y轴的取值范围。
YAxisBeginValue (页 1458)	RW	-	-	指定使用"YAxisIndex"引用的Y轴的取值范围下限。
YAxisColor (页 1459)	RW	-	-	指定使用"YAxisIndex"引用的 Y 轴的颜色。
YAxisCount (页 1459)	RW	-	-	指定已组态的 Y 轴的数目。
YAxisEndValue (页 1460)	RW	-	-	指定使用"YAxisIndex"引用 Y 轴的取值范围上限。
YAxisExponentialFormat (页 1461)	RW	-	-	指定是否在指数计数法中显示使用"YAxisIndex"引用的 Y 轴的值。
YAxisIndex (页 1461)	RW	-	-	引用 Y 轴。
YAxisInTrendColor (页 1462)	RW	-	-	指定使用"YAxisIndex"引用的轴的颜色是否与趋势颜色相对应。
YAxisLabel (页 1463)	RW	-	-	指定使用"YAxisIndex"引用的 Y 轴的标签文本。
YAxisName (页 1463)	RW	-	-	指定使用"YAxisIndex"引用的 Y 轴的名称。
YAxisPrecisions (页 1464)	RW	-	-	指定使用"YAxisIndex"引用的Y轴的值显示的小数位数。
YAxisRemove (页 1464)	RW	-	-	移除一个引用了其名称的 Y 轴。
YAxisRename (页 1465)	RW	-	-	指定使用"YAxisIndex"引用的 Y 轴的新名称。
YAxisRepos (页 1465)	RW	R	R	对于多个Y轴，指定使用"YAxisIndex"引用的Y轴的位置。
YAxisScalingType (页 1466)	RW	R	R	指定使用"YAxisIndex"引用的Y轴比例缩放的类型。
YAxisTrendWindow (页 1467)	RW	R	R	指定使用"YAxisIndex"引用的轴在其中显示的趋势图。
YAxisVisible (页 1467)	RW	R	R	指定是否在对象中显示使用"YAxisIndex"引用的 Y 轴。

表格 1-41　方法

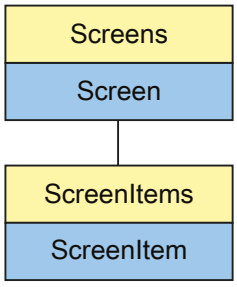
方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	√	√	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
AttachDB (页 1477)	√	-	-	执行控件的“连接备份”键功能。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。
DetachDB (页 1484)	√	-	-	执行控件的“断开备份”键功能。
导出 (页 1485)	√	-	-	执行控件的“导出日志”(Export log) 或“导出数据”(Export data) 按键功能。
GetStatusbarElement (页 1514)	√	-	-	以“ICCAxStatusbarElement”类型的形式返回按名称或索引指定的控件状态栏元素。
GetStatusbarElementCollection (页 1516)	√	-	-	以类型“ICCAxCollection”形式返回控件的所有状态栏元素的列表。
GetToolbarButton (页 1523)	√	-	-	以“ICCAxToolbarButton”类型的形式返回控件工具栏上按名称或索引指定的按钮功能。
GetToolbarButtonCollection (页 1525)	√	-	-	以“ICCAxCollection”类型的形式返回控件工具栏的所有按钮功能的列表。
GetTrend (页 1526)	√	-	-	以“ICCAxTrend”或“ICCAxFunctionTrend”类型的形式返回按名称或索引指定的 f(t) 或 f(x) 趋势视图趋势。
GetTrendCollection (页 1527)	√	-	-	以“ICCAxCollection”类型的形式返回 f(t) 或 f(x) 趋势视图的所有趋势的列表。
GetTrendWindow (页 1529)	√	-	-	以“ICCAxTrendWindow”类型的形式返回按名称或索引指定的 f(t) 或 f(x) 趋势视图的趋势窗口对象。
GetTrendWindowCollection (页 1530)	√	-	-	以“ICCAxCollection”类型的形式返回 f(t) 或 f(x) 趋势视图的所有趋势窗口对象的列表。
GetXAxis (页 1537)	√	-	-	以“ICCAxValueAxis”类型的形式返回具有指定名称或索引的 f(x) 趋势视图 X 轴对象。
GetXAxisCollection (页 1538)	√	-	-	以“ICCAxCollection”类型的形式返回所有 f(x) 趋势视图 X 轴对象的列表。

方法	RT Professional	RT Advanced	面板 RT	描述
GetYAxis (页 1540)	√	-	-	以“ICCAxValueAxis”类型的形式返回具有指定名称或索引的 f(x) 趋势视图 Y 轴对象。
GetYAxisCollection (页 1541)	√	-	-	以“ICCAxCollection”类型的形式返回所有 f(x) 趋势视图 Y 轴对象的列表。
MoveAxis (页 1547)	√	-	-	执行 f(t) 和 f(x) 趋势视图的“移动轴区域”(Move axes area) 按钮功能。
NextTrend (页 1554)	√	-	-	执行 f(t) 和 f(x) 趋势视图的“下一个趋势”(Next trend) 按钮功能。
OneToOneView (页 1555)	√	-	-	执行 f(t) 和 f(x) 趋势视图的“原始视图”(Original view) 按钮功能。
PreviousTrend (页 1556)	√	-	-	执行 f(t) 和 f(x) 趋势视图的“上一个趋势”(Previous trend) 按钮功能。
Print (页 1557)	√	-	-	执行控件的“打印”(Print) 按钮功能。
ShowHelp (页 1580)	√	-	-	执行控件的“帮助”(Help) 按钮功能。
ShowPropertyDialog (页 1585)	√	-	-	执行控件的“组态对话框”(Configuration dialog) 按钮功能。
ShowTagSelection (页 1589)	√	-	-	执行控件的“选择数据连接”(Select data connection) 按钮功能。
ShowTimeSelection (页 1590)	√	-	-	执行控件的“选择时间范围”(Select time range) 按钮功能。
ShowTrendSelection (页 1591)	√	-	-	执行 f(t) 和 f(x) 趋势视图的“选择趋势”(Select trends) 按钮功能。
StartStopUpdate (页 1592)	√	-	-	执行控件的“开始”(Start) 或“停止”(Stop) 按钮功能。
ZoomArea (页 1600)	√	-	-	执行 f(t) 和 f(x) 趋势视图的“缩放区域”(Zoom area) 按钮功能。
ZoomInOut (页 1600)	√	-	-	执行 f(t) 和 f(x) 趋势视图的“缩放”(Zoom +/-) 按钮功能。
ZoomInOutX (页 1602)	√	-	-	执行 f(x) 趋势视图的“X 轴缩放 +/-”(Zoom X axis +/-) 按钮功能。

方法	RT Professional	RT Advanced	面板 RT	描述
ZoomInOutY (页 1603)	√	-	-	执行 f(x) 趋势视图的"Y 轴缩放 +/-"(Zoom Y axis +/-) 按钮功能。
ZoomMove	√	-	-	执行 f(t) 和 f(x) 趋势视图的 "移动趋势区域"(Move trend area) 按钮功能。

Gauge

说明



表示"Gauge"对象。Gauge 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIGauge

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-42　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
AngleMax (页 620)	RW	RW	RW	指定"Gauge"对象刻度范围的角度。
AngleMin (页 620)	RW	RW	RW	指定"Gauge"对象起始刻度的角度。
BackColor (页 637)	RW	RW	RW	指定背景色。
BackFillStyle (页 643)	RW	-	-	指定填充图案。
BackPicture (页 651)	RW	-	-	指定背景图形。
BackStyle (页 652)	RW	-	-	指定背景样式。
BorderBackColor	-	-	-	-
BorderColor	-	-	-	-
BorderInnerStyle3D (页 691)	RW	-	-	指定对象边框的内侧部分的外观。
BorderOuterStyle3D (页 692)	RW	-	-	指定对象边框的外侧部分的外观。
BorderWidth (页 697)	RW	-	-	指定线宽。
BorderWidth3D (页 700)	RW	-	-	指定 3D 显示的内边框宽度。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CaptionColor (页 710)	-	RW	RW	指定将要在标题栏中显示文本的颜色。
CaptionFont (页 711)	RW	-	-	指定说明的字符集。
CaptionText (页 712)	RW	RW	RW	指定标题中将要显示的文本。
CaptionTop (页 713)	RW	-	-	指定仪器标签距对象上端的距离。
CenterColor (页 717)	RW	RW	RW	指定中心点的颜色。
CenterSize (页 718)	RW	-	-	指定刻度中心点的直径。
CompatibilityMode	-	-	-	-
CornerRadius	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
DangerRangeColor (页 770)	RW	RW	RW	指定"Gauge"对象危险刻度范围的颜色。
DangerRangeStart (页 771)	RW	RW	RW	指定"Gauge"对象危险范围的起始刻度值。
DangerRangeVisible (页 771)	RW	RW	RW	指定是否显示"Gauge"对象的危险刻度范围。
DeviceStyle	-	-	-	-
DialColor (页 780)	RW	RW	RW	确定所选对象的对话框的颜色。
DialFillStyle (页 780)	RW	-	-	指定背景类型。
DialPicture (页 781)	RW	-	-	指定表盘的图形。
DialSize (页 782)	RW	-	-	指定与几何属性"Width"或"Height"的较小值相关的刻度装置的直径。
EdgeStyle (页 788)	-	-	-	-
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。
Flashing (页 830)	RW	-	-	指定所选对象是否在运行系统中闪烁。
Gradation (页 858)	RW	RW	RW	指定"Gauge"对象两个主要标记之间的差值。
InnerDialColor	-	-	-	-
InnerDialInnerDistance	-	-	-	-
InnerDialOuterDistane	-	-	-	-
Height (页 863)	RW	R	R	指定高度。
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定 X 坐标的值。
LimitRangeCollection	-	-	-	-
Location	-	-	-	-
LockSquaredExtent (页 947)	RW	-	-	指定是否可以通过鼠标调节时钟大小。
MaximumValue (页 953)	RW	RW	RW	指定所选对象的刻度范围的最大值。

属性	RT Professional	RT Advanced	面板 RT	描述
MinimumValue (页 979)	RW	RW	RW	指定所选对象的刻度范围的最小值。
Name (页 985)	RW	-	-	指定对象名称。
NeedleHeight	-	-	-	-
NormalRangeColor (页 993)	RW	RW	RW	指定“Gauge”对象正常刻度范围的颜色。
NormalRangeVisible (页 994)	RW	RW	RW	指定是否显示“Gauge”对象的正常刻度范围。
PointerColor (页 1033)	RW	RW	RW	指定“Gauge”对象的指针颜色。
ProcessValue (页 1037)	RW	RW	RW	指定要显示的数值的默认值。
ScaleLabelColor (页 1062)	RW	RW	RW	指定“Gauge”对象刻度标记标签的颜色。
ScaleLabelFont (页 1063)	RW	-	-	指定刻度标签的字体。
ScaleTickColor (页 1065)	RW	RW	RW	指定“Gauge”对象刻度分度的颜色。
ScaleTickLabelPosition (页 1065)	RW	-	-	指定刻度标记标签所在的理论圆的直径。
ScaleTickLength (页 1066)	RW	-	-	指定主要刻度标记的长度。
ScaleTickPosition (页 1067)	RW	-	-	指定刻度所在的理论圆的直径。
ShowDecimalPoint (页 1102)	RW	-	-	指定是否对刻度加小数值（小数点后一位）或整数值标签。
ShowInnerDial	-	-	-	-
ShowLimitRanges	-	-	-	-
ShowPeakValuePointer (页 1112)	RW	RW	RW	指定从属指针是否将用于所选的对象。
Size	-	-	-	-
StyleItem	-	-	-	-

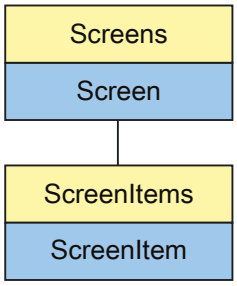
属性	RT Professional	RT Advanced	面板 RT	描述
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (页 1279)	RW	RW	RW	指定 Y 坐标的值。
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UnitColor (页 1347)	RW	RW	RW	指定测量单位的文本颜色。
UnitFont (页 1347)	RW	-	-	指定测量单位的字体。
UnitText (页 1348)	RW	RW	RW	指定测量单位的文本。
UnitTop (页 1349)	RW	-	-	指定测量单位距对象上端的距离。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
Visible (页 1418)	RW	RW	RW	指定是否显示所选对象。
WarningRangeColor (页 1427)	RW	RW	RW	指定“Gauge”对象警告刻度范围的颜色。
WarningRangeStart (页 1428)	RW	RW	RW	指定“Gauge”对象警告范围的起始刻度值。
WarningRangeVisible (页 1428)	RW	RW	RW	指定是否显示“Gauge”对象的警告刻度范围。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。

表格 1-43　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	√	√	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

GraphicIOField

说明



表示“Graphic I/O field”对象。GraphicIOField 对象是 ScreenItems 列表的元素。

下列对象属性的可用性取决于所选的模式：

属性	“输入”(Input)	“输出”	“输入/输出”(Input/Output)	“两种状态”(Two states)
BorderColor	--	--	--	x
Enabled	x	--	x	--
FocusColor	x	--	x	--
FocusWidth	x	--	x	--
HelpText	x	--	x	--
TransparentColor	x	x	x	--
UseTransparentColor	x	x	x	--

VBS 的类型标识符

HMIGraphicIOField

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-44　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AboveUpperLimitColor	-	-	-	-
AdaptPicture	-	-	-	-
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	RW	RW	指定操作权限。
AutoSizing	-	-	-	-
BackColor (页 637)	-	RW	RW	指定背景色。
BackFillStyle	-	-	-	-
BelowLowerLimitColor	-	-	-	-
BitNumber	-	-	-	-
BorderBackColor (页 674)	RW	-	-	指定间断边框线的背景色。
BorderColor (页 678)	RW	-	-	指定线颜色。
BorderFlashingColorOff (页 683)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
BorderFlashingColorOn (页 685)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
BorderFlashingEnabled (页 687)	RW	-	-	指定是否在运行系统中闪烁对象的限值。
BorderFlashingRate	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
BorderStyle (页 695)	RW	-	-	指定边框线类型。
BorderWidth (页 697)	RW	-	-	指定线宽。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (页 764)	RW	-	-	指定角形状。
DrawInsideFrame (页 787)	RW	-	-	指定线宽大于 1 的边框线是显示在边框内，还是与边框对称。
EdgeStyle (页 788)	RW	-	-	指定线样式。
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。
FitToLargest	-	-	-	-
Flashing	-	-	-	-
FlashingEnabled (页 836)	RW	-	-	指定是否为运行系统中的对象激活闪烁。
FlashingOnLimitViolation	-	-	-	-
FlashingRate (页 839)	RW	-	-	指定边界线的闪烁频率。
FlashTransparentColor (页 841)	RW	-	-	指定设置为“透明”的闪烁图形其位图对象的颜色。
FocusColor (页 843)	-	RW	RW	当对象获得焦点时，指定焦点边框的颜色。
FocusWidth (页 844)	-	RW	RW	当对象获得焦点时，指定边框宽度。
Height (页 863)	RW	R	R	指定高度。
HelpText (页 868)	-	RW	RW	指定作为特定对象的用户帮助显示在运行系统中的工具提示。
IsImageMiddleAligned	-	-	-	-
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。

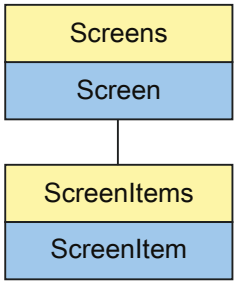
属性	RT Professional	RT Advanced	面板 RT	描述
Location	-	-	-	-
Mode	-	-	-	-
Name	-	-	-	-
OnValue	-	-	-	-
PictureList	-	-	-	-
PictureOff (页 1029)	RW	-	-	指定在“关闭”(Off) 状态下显示的画面。
PictureOn (页 1030)	RW	-	-	指定在“开启”状态下显示的图形文件。
PictureRotation	-	-	-	-
ProcessValue (页 1037)	RW	-	-	指定要显示的数值的默认值。
ScrollBarOrientation	-	-	-	-
ShowScrollBar	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (页 1278)	RW	RW	RW	指定工具提示文本。
Top (页 1279)	RW	RW	RW	指定 Y 坐标的值。
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
TransparentColor (页 1286)	RW	RW	RW	指定将已分配图形（*.bmp 或 *dib）的哪种颜色设置为“透明”。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
UseFlashTransparentColor (页 1359)	RW	-	-	指定是否将闪烁图形位图对象的颜色设置为“透明”(transparent)。
UseTransparentColor (页 1366)	RW	RW	RW	指定“TransparentColor”属性定义的颜色是否为透明。
Visible (页 1418)	RW	RW	RW	指定是否显示所选对象。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。

表格 1-45　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	√	√	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

GraphicView

说明



表示“Graphic view”对象。GraphicView 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIGraphicView

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-46　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
AdaptPicture	-	-	-	-
Authorization (页 626)	RW	-	-	指定操作权限。
AutoSizing	-	-	-	-
BackColor (页 637)	RW	RW	RW	指定背景色。
BackFillStyle (页 643)	RW	-	-	指定填充图案。
BackFlashingColorOff (页 645)	RW	-	-	指定闪烁状态“关闭”(Off) 的背景色。
BackFlashingColorOn (页 646)	RW	-	-	指定闪烁状态“打开”(On) 的背景色。
BackFlashingEnabled (页 648)	RW	-	-	指定是否在运行系统中闪烁背景。
BackFlashingRate (页 649)	RW	-	-	指定背景的闪烁频率。
BorderBackColor (页 674)	RW	-	-	指定间断边框线的背景色。
BorderColor (页 678)	RW	-	-	指定线颜色。
BorderFlashingColorOff (页 683)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。

属性	RT Professional	RT Advanced	面板 RT	描述
BorderFlashingColorOn (页 685)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
BorderFlashingEnabled (页 687)	RW	-	-	指定是否在运行系统中闪烁对象的限值。
BorderFlashingRate (页 689)	RW	-	-	指定边界线的闪烁频率。
BorderStyle (页 695)	RW	-	-	指定边框线类型。
BorderWidth (页 697)	RW	-	-	指定线宽。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (页 764)	RW	-	-	指定角形状。
DrawInsideFrame (页 787)	RW	-	-	指定线宽大于 1 的边框线是显示在边框内，还是与边框对称。
EdgeStyle (页 788)	RW	-	-	指定线样式。
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作对象。
FillingDirection (页 825)	RW	-	-	指定填充方向。
FillPatternColor (页 821)	RW	-	-	确定填充样式的颜色。
FitToLargest	-	-	-	-
Flashing	-	-	-	-
Height (页 863)	RW	R	R	指定高度。
IsImageMiddleAligned	-	-	-	-
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
Location	-	-	-	-
Name	-	-	-	-

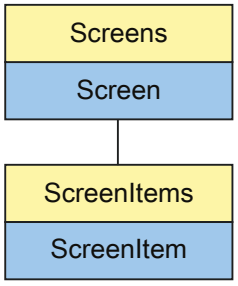
属性	RT Professional	RT Advanced	面板 RT	描述
Picture (页 1026)	RW	-	-	指定将在运行系统的图形对象中显示的 WinCC 项目图形的画面。
PictureRotation	-	-	-	-
RelativeFillLevel (页 1046)	RW	-	-	指定对象的填充百分比。
ShowFillLevel (页 1108)	RW	-	-	指定对象是否填充。
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (页 1278)	RW	RW	RW	指定工具提示文本。
Top (页 1279)	RW	RW	RW	指定 Y 坐标的值。
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
TransparentColor (页 1286)	RW	RW	RW	指定将已分配图形（*.bmp 或 *dib）的哪种颜色设置为“透明”。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
UseTransparentColor (页 1366)	RW	RW	RW	指定"TransparentColor"属性定义的颜色是否为透明。
Visible (页 1418)	RW	RW	RW	指定是否显示所选对象。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。

表格 1-47　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	√	√	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

HTMLBrowser

说明



表示“HTML 浏览器”对象。HTMLBrowser 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIHTMLBrowser

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-48　属性

属性	RT Professional	RT Advanced	面板 RT	描述
Address (页 608)	RW	RW	RW	指定在 HTML 浏览器中将打开的 Web 地址。
AddressEnabled	-	-	-	-
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-
BrowserTypeUsed	-	-	-	-
CanBeGrouped	-	-	-	-
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。
Flashing	-	-	-	-
Height (页 863)	RW	R	R	指定高度。
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定 X 坐标的值。
Location	-	-	-	-
Name	-	-	-	-
ShowStatusBar (页 1120)	RW	-	-	指定是否显示状态栏。
ShowToolBar	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (页 1279)	RW	RW	RW	指定 Y 坐标的值。

属性	RT Professional	RT Advanced	面板 RT	描述
Visible (页 1418)	RW	RW	RW	指定是否显示所选对象。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。

表格 1-49　方法

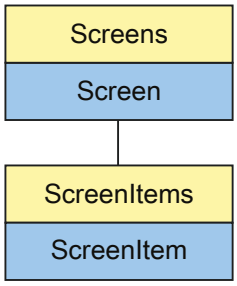
方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	√	√	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。
SetHtml (页 1577)	√	-	-	写入处于 HTML 浏览器显示范围内的 HTML 代码。

参见

SetHTML (页 1577)

IOField

说明



表示“I/O field”对象。IOField 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIIOField

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-50　属性

属性	RT Professional	RT Advanced	Panel RT	说明
AboveUpperLimitColor (页 596)	RW	-	-	针对“超出上限”事件设置指定对象的颜色。
AcceptOnExit (页 597)	RW	-	-	指定是否在保留输入字段内容时，自动对输入字段进行确认。
AcceptOnFull (页 598)	RW	-	-	指定是否在输入定义数量的值时保留输入字段内容并自动进行确认。
AdaptBorder (页 606)	RW	-	-	指定对象的边框是否根据文本的大小进行动态调整。
AllTagTypesAllowed	-	-	-	-
AskOperationMotive (页 624)	RW	-	-	指定是否记录操作此对象的原因。
Authorization (页 626)	R	R	R	指定所选对象在运行系统中的操作权限。
BackColor (页 637)	RW	RW	RW	指定所选对象的背景色。
BackFillStyle (页 643)	RW	-	-	指定所选对象的填充图案。
BackFlashingColorOff (页 645)	RW	-	-	指定闪烁状态“关闭”(Off) 的背景色。
BackFlashingColorOn (页 646)	RW	-	-	指定闪烁状态“打开”(On) 的背景色。
BackFlashingEnabled (页 648)	RW	-	-	指定是否在运行系统中闪烁指定对象的背景。
BackFlashingRate (页 649)	RW	-	-	指定所选对象的背景闪烁的频率。
BelowLowerLimitColor (页 659)	RW	-	-	为“低于下限”(Low limit violated) 事件指定颜色。
BorderBackColor (页 674)	RW	-	-	指定所选对象间断边框线的背景色。
BorderColor	-	-	-	-

属性	RT Professional	RT Advanced	Panel RT	说明
BorderFlashingColorOff (页 683)	RW	-	-	指定闪烁状态“关闭”(Off) 的所选对象的边框线颜色。
BorderFlashingColorOn (页 685)	RW	-	-	指定闪烁状态“打开”(On) 的所选对象的边框线颜色。
BorderFlashingEnabled (页 687)	RW	-	-	指定所选对象的边框线是否在运行系统中闪烁。
BorderFlashingRate (页 689)	RW	-	-	指定所选对象的边框线的闪烁频率。
BorderStyle (页 695)	RW	-	-	指定所选对象边框线的类型。
BorderWidth (页 697)	RW	-	-	指定所选对象的线宽。
BorderWidth3D	-	-	-	-
BottomMargin	-	-	-	-
CanBeGrouped	-	-	-	-
ClearOnError (页 721)	RW	-	-	指定是否自动删除该对象的无效输入。
ClearOnFocus (页 722)	RW	-	-	指定是否在激活 I/O 域后删除字段条目。
CornerRadius	-	-	-	-
CornerStyle (页 764)	RW	-	-	指定特定对象的角形状。
CursorControl (页 769)	RW	-	-	指定是否在鼠标光标离开某字段后按 TAB 顺序跳至下一字段。
DataFormat (页 772)	RW	-	-	指定 I/O 域的显示格式。
DeviceStyle	-	-	-	-
EdgeStyle (页 788)	RW	RW	RW	指定所选对象的线样式。
EditOnFocus (页 791)	RW	-	-	指定是否在使用 <Tab> 键选择输入字段后可以立即输入数据。
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。
FieldLength	-	-	-	-
FillPatternColor (页 821)	RW	-	-	指定所选对象的填充样式的颜色。
FitToLargest	-	-	-	-

属性	RT Professional	RT Advanced	Panel RT	说明
Flashing	-	-	-	-
FlashingColorOff (页 832)	RW	-	-	指定闪烁状态“关闭”(Off) 的所选对象的边框线颜色。
FlashingColorOn (页 834)	RW	-	-	指定闪烁状态“打开”(On) 的所选对象的边框线颜色。
FlashingEnabled (页 836)	RW	-	-	指定所选对象的边框线是否在运行系统中闪烁。
FlashingOnLimitViolation	-	-	-	-
FlashingRate (页 839)	RW	-	-	指定所选对象的边框线的闪烁频率。
Font	-	-	-	-
FontBold (页 849)	RW	-	-	指定所选对象的文本是否以粗体显示。
FontItalic (页 849)	RW	-	-	指定所选对象的文本是否以斜体显示。
FontName (页 851)	RW	-	-	指定所选对象的字体。
FontSize (页 852)	RW	-	-	指定所选对象的文本字体大小。
FontUnderline (页 853)	RW	-	-	指定所选对象的文本是否加下划线。
ForeColor (页 854)	RW	RW	RW	指定所选对象的文本字体颜色。
FormatPattern (页 856)	RW	-	-	指定输出值的格式。
Height (页 863)	RW	R	R	指定所选对象的高度。
HelpText (页 868)	-	RW	RW	指定作为特定对象的用户帮助显示在运行系统中的工具提示。
HiddenInput (页 869)	RW	-	-	指定在输入期间显示输入值还是使用 * 表示每个字符。
HorizontalAlignment (页 881)	RW	RW	RW	指定所选对象内的文本水平对齐。
InputValue (页 894)	RW	-	-	计算输入值。
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定所选对象的 X 坐标的值。
LeftMargin	-	-	-	-

属性	RT Professional	RT Advanced	Panel RT	说明
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
LineWrap	-	-	-	-
Location	-	-	-	-
LogOperation (页 948)	RW	-	-	指定是否在操作对象后将报警输出到报警系统。
LowerLimit (页 950)	RW	-	-	指定输入值的下限值。
Mode (页 981)	RW	-	-	指定所选对象的字段类型。
Name	-	-	-	-
ProcessValue (页 1037)	RW	RW	RW	指定要显示的数值的默认值。
RightMargin	-	-	-	-
ShiftDecimalPoint	-	-	-	-
ShowBadTagState (页 1099)	RW	-	-	定义检测到不良质量代码或变量状态时对象是否变成灰色。
ShowLeadingZeros	-	-	-	-
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TextOrientation (页 1189)	RW	-	-	指定所选对象的文本方向。
ToolTipText (页 1278)	RW	RW	RW	指定工具提示文本。
Top (页 1279)	RW	RW	RW	指定对象的 Y 坐标的值。
TopMargin	-	-	-	-
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
Unit (页 1346)	RW	-	-	指定测量单位。
UpperLimit (页 1350)	RW	-	-	指定输入值的上限值。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。

属性	RT Professional	RT Advanced	Panel RT	说明
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有在激活设计中定义的底纹。
UseTagLimitColors (页 1366)	RW	-	-	指定超出为对象组态的上限和下限时，是否以颜色突出显示。
UseTwoHandOperation (页 1369)	-	R	R	返回关于双手操作是否启用的信息。
VerticalAlignment (页 1413)	RW	RW	RW	指定所选对象中的文本垂直对齐方式。
Visible (页 1418)	RW	RW	RW	指定是否显示所选对象。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。

表格 1-51　方法

方法	RT Professional	RT Advanced	Panel RT	说明
Activate (页 1470)	支持	支持	支持	启用永久区域或根画面。
ActivateDynamic (页 1473)	支持	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	支持	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

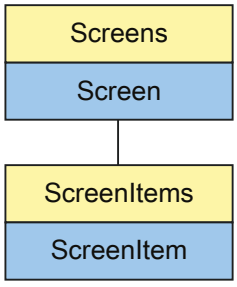
参见

BorderColor (页 678)

1.5.4.3 Objects K-Z

Line

说明



表示"Line"对象。Line 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMILine

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-52　属性

属性	RT Professional	RT Advanced	Panel RT	说明
ActualPointIndex	-	-	-	-
ActualPointLeft	-	-	-	-
ActualPointTop	-	-	-	-
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定所选对象在运行系统中的操作权限。
BackColor (页 637)	RW	RW	RW	指定所选对象的背景色。
BorderEndStyle (页 683)	RW	-	-	指定所选对象线端的类型。
BorderStyle (页 695)	RW	-	-	指定所选对象边框线的类型。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Color (页 723)	RW	RW	RW	指定所选对象的线颜色。
CornerStyle (页 764)	RW	-	-	指定所选对象的拐角形状。
DeviceStyle	-	-	-	-
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。
EndLeft	-	-	-	-
EndStyle (页 800)	RW	RW	RW	指定如何显示所选对象的线端。
EndTop	-	-	-	-
FillStyle (页 824)	-	RW	RW	指定所选对象的填充图案。
Flashing	-	-	-	-
FlashingColorOff (页 832)	RW	-	-	指定闪烁状态“关闭”(Off) 的所选对象的边框线颜色。
FlashingColorOn (页 834)	RW	-	-	指定闪烁状态“打开”(On) 的所选对象的边框线颜色。
FlashingEnabled (页 836)	RW	-	-	指定所选对象的边框线是否在运行系统中闪烁。

属性	RT Professional	RT Advanced	Panel RT	说明
FlashingRate (页 839)	RW	-	-	指定所选对象的边框线的闪烁频率。
Height (页 863)	RW	R	R	指定所选对象的高度。
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定所选对象的 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
LineWidth (页 943)	RW	RW	RW	指定所选对象的线宽。
Location	-	-	-	-
Name	-	-	-	-
Points	-	-	-	-
PointsCount	-	-	-	-
RotationAngle (页 1048)	RW	-	-	以度为单位指定旋转角度。
RotationCenterLeft (页 1049)	RW	-	-	指定运行系统中对象旋转中心点的 X 坐标。
RotationCenterTop (页 1050)	RW	-	-	指定运行系统中对象旋转中心点的 Y 坐标。
Size	-	-	-	-
StartLeft	-	-	-	-
StartStyle (页 1135)	RW	RW	RW	指定如何显示所选对象的直线起始端。
StartTop	-	-	-	-
Style (页 1163)	RW	RW	RW	指定所选对象的线样式。
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (页 1278)	RW	RW	RW	指定工具提示文本。
Top (页 1279)	RW	RW	RW	指定所选对象的 Y 坐标的值。
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。

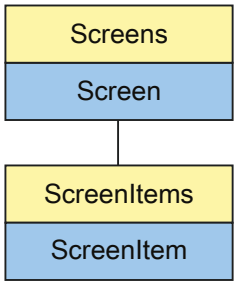
属性	RT Professional	RT Advanced	Panel RT	说明
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有在激活设计中定义的底纹。
Visible (页 1418)	RW	RW	RW	指定是否显示所选对象。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。

表格 1-53　方法

方法	RT Professional	RT Advanced	Panel RT	说明
激活 (页 1470)	支持	支持	支持	启用永久区域或根画面。
ActivateDynamic (页 1473)	支持	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	支持	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

Listbox

描述



表示“Listbox”对象。Listbox 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIListBox

应用

在下面的示例中，名称为“ListBox1”的对象向右移动 10 个像素：

```
'VBS21
Dim objListBox
Set objListBox = ScreenItems("ListBox1")
objListBox.Left = objListBox.Left + 10
```

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
AskOperationMotive (页 624)	RW	-	-	指定是否记录操作此对象的原因。
Authorization (页 626)	RW	-	-	指定操作权限。
BackColor (页 637)	RW	-	-	指定背景色。
BackFillStyle (页 643)	RW	-	-	指定填充图案。
BorderBackColor (页 674)	RW	-	-	指定间断边框线的背景色。
BorderColor (页 678)	RW	-	-	指定线颜色。
BorderStyle (页 695)	RW	-	-	指定边框线类型。
BorderWidth (页 697)	RW	-	-	指定线宽。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (页 764)	RW	-	-	指定角形状。
CountVisibleItems (页 767)	RW	-	-	指定下拉列表中将包含的线数。
EdgeStyle (页 788)	RW	-	-	指定线样式。
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。
FillPatternColor	-	-	-	-
FitToLargest	-	-	-	-
Font	-	-	-	-
FontBold (页 849)	RW	-	-	指定文本是否以粗体显示。

属性	RT Professional	RT Advanced	面板 RT	描述
FontItalic (页 849)	RW	-	-	指定文本是否以斜体显示。
FontName (页 851)	RW	-	-	指定字体。
FontSize (页 852)	RW	-	-	指定文本字体大小。
FontUnderline (页 853)	RW	-	-	指定文本是否应以下划线标出。
ForeColor (页 854)	RW	-	-	指定文本字体颜色。
Height (页 863)	RW	-	-	指定高度。
HorizontalAlignment (页 881)	RW	-	-	指定文本水平对齐。
Index (页 887)	RW	-	-	指定所选文本域的索引。
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
Location	-	-	-	-
LogOperation (页 948)	RW	-	-	指定是否在操作对象后将报警输出到报警系统。
Name	-	-	-	-
SelectedIndex (页 1078)	RW	-	-	指定其关联文本在组合框或列表框中高亮显示的索引。
SelectedText (页 1081)	RW	-	-	指定使用“SelectedIndex”引用的条目的文本。
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Text (页 1185)	RW	-	-	指定文本域的标签。
TextHandles	-	-	-	-
Texts	-	-	-	-
ToolTipText (页 1278)	RW	-	-	指定工具提示文本。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。

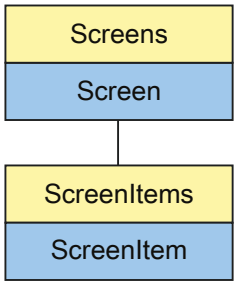
属性	RT Professional	RT Advanced	面板 RT	描述
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-54　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

MediaPlayer

描述



表示“Media Player”对象。MediaPlayer 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIMediaPlayer

应用

在下面的示例中，名称为“Control1”的对象向右移动 16 个像素：

```
'VBS60
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left + 16
```

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-55　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
AspectRatio (页 624)	RW	-	-	指定是否在更改大小时保持媒体播放器的纵横比。
AutoStart	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。
FileName (页 819)	RW	-	-	指定要加载的文件的名称。
Height (页 863)	RW	R	R	指定高度。
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定 X 坐标的值。
Location	-	-	-	-
Name	-	-	-	-
PictureSizeMode (页 1031)	RW	-	-	指定媒体播放器和将要显示的内容之间的尺寸调整。
PlayCount	-	-	-	-
PlayEndless (页 1032)	RW	-	-	指定要循环播放媒体文件。
PopupMenuEnabled	-	-	-	-
ShowControls (页 1101)	RW	-	-	指定显示工具栏。
ShowFeatureBackward (页 1103)	RW	-	-	指定在运行系统中显示“后退”(Backward) 按钮。

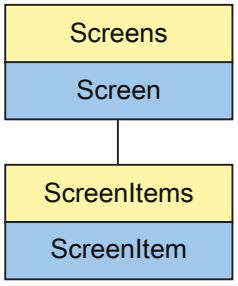
属性	RT Professional	RT Advanced	面板 RT	描述
ShowFeatureForward (页 1104)	RW	-	-	指定在运行系统中显示“快进”(Forward) 按钮。
ShowFeatureFullScreen (页 1105)	RW	-	-	指定可以全屏显示媒体播放器。
ShowFeatureFullVolume	-	-	-	-
ShowFeaturePause (页 1106)	RW	-	-	指定在运行系统中显示“暂停”(Pause) 按钮。
ShowFeaturePlay (页 1106)	RW	-	-	指定在运行系统中显示“播放”(Play) 按钮。
ShowFeatureStop (页 1107)	RW	-	-	指定在运行系统中显示“停止”(Stop) 按钮。
ShowStatusBar	-	-	-	-
ShowTracker (页 1126)	RW	-	-	指定显示播放列表。
Size	-	-	-	-
StepSeconds (页 1161)	RW	-	-	指定在按下“向前”(Forward) 或“向后”(Backward) 键后，以秒为单位的步长间隔。
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (页 1279)	RW	RW	RW	指定 Y 坐标的值。
Visible (页 1418)	RW	RW	RW	指定是否显示所选对象。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。

表格 1-56　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	-	√	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

MultiLineEdit

描述



表示“可编辑的文本域”(Editable text field) 对象。MultiLineEdit 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIMultiLineEdit

应用

在下面的示例中，名称为“MultiLineEdit1”的对象向右移动 10 个像素：

```
'VBS21
Dim objMultiLineEdit
Set objMultiLineEdit = ScreenItems("MultiLineEdit1")
objMultiLineEdit.Left = objMultiLineEdit.Left + 10
```

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-57　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定操作权限。
BackColor (页 637)	RW	-	-	指定背景色。
BorderBackColor (页 674)	RW	-	-	指定间断边框线的背景色。
BorderColor (页 678)	RW	-	-	指定线颜色。
BorderStyle (页 695)	RW	-	-	指定边框线类型。
BorderWidth (页 697)	RW	-	-	指定线宽。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (页 764)	RW	-	-	指定角形状。
EdgeStyle (页 788)	RW	-	-	指定线样式。
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。
Font	-	-	-	-

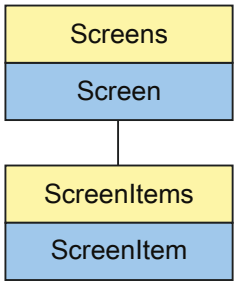
属性	RT Professional	RT Advanced	面板 RT	描述
FontBold (页 849)	RW	-	-	指定文本是否以粗体显示。
FontItalic (页 849)	RW	-	-	指定文本是否以斜体显示。
FontName	-	-	-	-
FontSize (页 852)	RW	-	-	指定文本字体大小。
FontUnderline (页 853)	RW	-	-	指定文本是否应以下划线标出。
ForeColor (页 854)	RW	-	-	指定文本字体颜色。
Height (页 863)	RW	-	-	指定高度。
HorizontalAlignment (页 881)	RW	-	-	指定文本水平对齐。
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Text (页 1185)	RW	-	-	指定文本域的标签。
ToolTipText (页 1278)	RW	-	-	指定工具提示文本。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-58　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。
GetSelectionText	√	-	-	-
SetSelection	√	-	-	-
SetSelectionText	√	-	-	-

OnlineTableControl

说明



表示“Table view”对象。OnlineTableControl 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIOnlineTableControl

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

属性	RT Professional	RT Advanced	面板 RT	说明
AllTagTypesAllowed	-	-	-	-
ApplyProjectSettingsForDesignMode	-	-	-	-
AutoCompleteColumns (页 629)	RW	-	-	指定在控件比组态的列宽时是否显示空列。
AutoCompleteRows (页 629)	RW	-	-	指定在控件比组态的行数长时是否显示空行。
AutoSelectionColors (页 632)	RW	-	-	指定是否将系统定义的颜色用作单元格和行的选择颜色。
AutoSelectionRectColor (页 632)	RW	-	-	指定是否使用系统定义的颜色显示选择框架。
BackColor (页 637)	RW	-	-	指定背景色。
BorderColor (页 678)	RW	-	-	指定线颜色。
BorderWidth (页 697)	RW	-	-	指定线宽。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Caption	-	-	-	-
CellCut (页 713)	RW	-	-	指定单元格过窄时是否缩略单元格内容。
CellSpaceBottom (页 714)	RW	-	-	指定表格单元格的下边距。
CellSpaceLeft (页 715)	RW	-	-	指定表格单元格中采用的左缩进。

属性	RT Professional	RT Advanced	面板 RT	说明
CellSpaceRight (页 716)	RW	-	-	指定表格单元格中采用的右缩进。
CellSpaceTop (页 717)	RW	-	-	指定表格单元格的上边距。
Closeable (页 722)	RW	-	-	指定是否可在运行系统中关闭对象。
ColumnResize (页 742)	RW	-	-	指定是否可以更改列宽。
ColumnScrollbar (页 743)	RW	-	-	指定将显示水平滚动条的时间。
ColumnTitleAlignment (页 754)	RW	-	-	指定引用了"ColumnIndex"的列标题的对齐方式。
ColumnTitles (页 755)	RW	-	-	指定是否显示列标题。
ControlDesignMode (页 762)	RW	-	-	指定控制设计。
Enabled (页 792)	-	-	-	-
EnableEdit (页 797)	RW	-	-	指定是否可以在运行系统中编辑显示的数据。
ExportDelimiter	-	-	-	-
ExportDirectoryChangeable (页 808)	RW	-	-	指定是否可以在运行系统中更改数据导出目录。
ExportDirectoryname (页 809)	RW	-	-	指定导出的运行系统数据的目标目录。
ExportFileExtension (页 810)	RW	-	-	指定导出文件的文件扩展名。
ExportFilename (页 811)	RW	-	-	指定导出的运行系统数据的目标文件名称。
ExportFilenameChangeable (页 812)	RW	-	-	指定是否可以在运行系统中更改导出文件名。
ExportFormat	-	-	-	-
ExportFormatGuid (页 813)	RW	-	-	指定 ID 编号和导出数据源的分配。

属性	RT Professional	RT Advanced	面板 RT	说明
ExportFormatName (页 814)	RW	-	-	指定导出文件格式。
ExportParameters (页 815)	RW	-	-	通过属性对话框指定所选格式的参数。
ExportSelection (页 816)	RW	-	-	指定导出控件的哪些运行系统数据。
ExportShowDialog (页 817)	RW	-	-	指定是否在运行系统中显示数据导出对话框。
FillPattern	-	-	-	-
FillPatternColor	-	-	-	-
Font (页 846)	RW	-	-	指定字体。
GridLineColor (页 860)	RW	-	-	指定网格线颜色。
GridLineWidth (页 862)	RW	-	-	指定分隔线的宽度（以像素为单位）。
Height (页 863)	RW	-	-	指定高度。
HorizontalGridLines (页 883)	RW	-	-	指定是否显示水平分隔线。
IconSpace (页 886)	RW	-	-	指定表格单元格中图标和文本的间距。
Layer	-	-	-	-
Left (页 920)	RW	-	-	指定 X 坐标的值。
LineBackgroundColor	-	-	-	-
LineColor (页 939)	RW	-	-	指定窗口分隔线的颜色。
LineStyle	-	-	-	-
LineWidth (页 943)	RW	-	-	指定线宽。
LoadDataImmediately (页 946)	RW	-	-	指定是否下载日志中记录的在调用画面时显示的时间范围的变量值。
Location	-	-	-	-
MaximumNumberOfTimeColumns	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	说明
MaximumNumberOfValueColumns	-	-	-	-
MinimumNumberOfTimeColumns	-	-	-	-
MinimumNumberOfValueColumns	-	-	-	-
Moveable (页 983)	RW	-	-	指定关于运行系统中是否可以移动对象的信息。
Name	-	-	-	-
Object	-	-	-	-
OcxGuid	-	-	-	-
OcxState	-	-	-	-
OcxStateForEs2Rt	-	-	-	-
Online (页 999)	RW	-	-	指定更新的起始与停止。
PrintJob (页 1036)	RW	-	-	指定一个在“报表”编辑器中创建的打印作业。
RowScrollbar (页 1052)	RW	-	-	指定垂直滚动条将要显示的时间。
RowTitleAlignment (页 1053)	RW	-	-	指定行标题对齐的类型。
RowTitles (页 1054)	RW	-	-	指定是否显示已编号的列标题。
RTPersistence (页 1054)	RW	-	-	指定在画面更改后是否保留在线组态。
RTPersistenceAuthorization (页 1055)	RW	-	-	指定运行系统中在线组态所需的权限。
RTPersistenceType (页 1057)	RW	-	-	指定如何保留 WinCC 的在线组态。
SelectedCellColor (页 1075)	RW	-	-	指定所选单元格的背景色。
SelectedCellForeColor (页 1076)	RW	-	-	指定所选单元格的字体颜色。
SelectedRowColor (页 1079)	RW	-	-	指定所选行的背景色。

属性	RT Professional	RT Advanced	面板 RT	说明
SelectedRowForeColor (页 1080)	RW	-	-	指定所选行的字体颜色。
SelectedTitleColor (页 1081)	RW	-	-	指定所选表格标题的背景色。
SelectedTitleForeColor (页 1082)	RW	-	-	指定所选表格标题的字体颜色。
SelectionColoring (页 1085)	RW	-	-	指定是否对单元格或行使用选择颜色。
SelectionRect (页 1087)	RW	-	-	启用所选单元格或行的选择边框。
SelectionRectColor (页 1088)	RW	-	-	指定报警窗口中选择矩形的颜色。
SelectionRectWidth (页 1089)	RW	-	-	如果 SelectionType 等于“1”，则指定报警窗口中选择矩形的线宽。
SelectionType (页 1090)	RW	-	-	指定可标记的行数。
ShowSortButton (页 1117)	RW	-	-	指定是否在垂直滚动条上显示排序按钮。
ShowSortIcon (页 1118)	RW	-	-	指定是否显示排序图标。
ShowSortIndex (页 1118)	RW	-	-	指定是否显示排序索引。
ShowTitle (页 1124)	RW	-	-	为对象指定窗口边框和窗口标题的样式。
Size	-	-	-	-
Sizeable (页 1129)	RW	-	-	指定可以在运行系统中更改对象的大小。
SortSequence (页 1132)	RW	-	-	如果操作员在运行系统中单击列标题，指定排序顺序会如何改变。
StatusbarBackColor (页 1138)	RW	-	-	指定状态栏的背景色。
StatusbarElementAdd (页 1139)	RW	-	-	创建一个新的用户自定义状态栏元素。

属性	RT Professional	RT Advanced	面板 RT	说明
StatusbarElementAutoSize (页 1140)	RW	-	-	指定是否自动设置引用“StatusbarElementIndex”的状态栏元素的宽度。
StatusbarElementCount (页 1141)	RW	-	-	指定已组态的状态栏元素的数量。
StatusbarElementIconId (页 1142)	RW	-	-	使用其图标 ID 来引用状态栏元素。
StatusbarElementID (页 1143)	RW	-	-	使用其元素 ID 来引用状态栏元素。
StatusbarElementIndex (页 1144)	RW	-	-	引用状态栏元素。
StatusbarElementName (页 1145)	RW	-	-	指定使用“StatusbarElementIndex”引用的状态栏元素的名称。
StatusbarElementRemove (页 1146)	RW	-	-	删除应用其名称的用户自定义状态栏元素。
StatusbarElementRename (页 1147)	RW	-	-	指定使用“StatusbarElementIndex”引用的用户自定义的状态栏元素的新名称。
StatusbarElementRepos (页 1148)	RW	-	-	指定对象状态栏中使用“StatusbarElementIndex”引用的元素位置。
StatusbarElements	-	-	-	-
StatusbarElementText (页 1149)	RW	-	-	指定使用“StatusbarElementIndex”引用的用户自定义的状态栏元素的文本。
StatusbarElementTooltipText (页 1150)	RW	-	-	指定使用“StatusbarElementIndex”引用的用户自定义的状态栏元素的工具提示文本。
StatusbarElementUserDefined (页 1151)	RW	-	-	指定组态工程师是否已添加使用“StatusbarElementIndex”引用的状态栏元素作为新的用户自定义元素。
StatusbarElementVisible (页 1152)	RW	-	-	指定是否在对象中显示使用“StatusbarElementIndex”引用的状态栏元素。
StatusbarElementWidth (页 1153)	RW	-	-	指定使用“StatusbarElementIndex”引用的状态栏元素的宽度（以像素为单位）。
StatusbarFont (页 1154)	RW	-	-	指定状态栏中文本的字体。

属性	RT Professional	RT Advanced	面板 RT	说明
StatusbarFontColor (页 1155)	RW	-	-	指定状态栏中文本的颜色。
StatusbarShowTooltips (页 1156)	RW	-	-	指定是否在运行系统中显示状态栏元素的工具提示。
StatusbarText (页 1157)	RW	-	-	指定状态栏中的默认文本。
StatusbarUseBackColor (页 1158)	RW	-	-	指定是否显示状态栏的背景色。
StatusbarVisible (页 1159)	RW	-	-	指定是否显示控件的状态栏。
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableColor (页 1171)	RW	-	-	指定对象中表格行的背景色。
TableColor2 (页 1172)	RW	-	-	指定对象中表格行的第二种背景色。
TableForeColor (页 1173)	RW	-	-	指定对象表格单元格中采用的文本颜色。
TableForeColor2 (页 1174)	RW	-	-	指定对象表格单元格中采用的第二种文本颜色。
TimeBase (页 1211)	RW	-	-	指定显示时间值的时区。
TimeColumnActualize (页 1213)	RW	-	-	指定是否更新使用“TimeColumnIndex”引用的时间列中的值。
TimeColumnAdd (页 1213)	RW	-	-	创建新的时间列。
TimeColumnAlignment (页 1214)	RW	-	-	指定使用“TimeColumnIndex”引用的时间列中的文本如何对齐。
TimeColumnBackColor (页 1215)	RW	-	-	指定使用“TimeColumnIndex”引用的时间列的背景色。
TimeColumnBeginTime (页 1215)	RW	-	-	为使用“TimeColumnIndex”引用的时间列指定时间范围的起始点。
TimeColumnCaption (页 1216)	RW	-	-	指定使用“TimeColumnIndex”引用的时间列的标签。

属性	RT Professional	RT Advanced	面板 RT	说明
TimeColumnCount (页 1217)	RW	-	-	指定已组态的时间列数。
TimeColumnDateFormat (页 1217)	RW	-	-	指定用于显示使用“TimeColumnIndex”引用的时间列的日期格式。
TimeColumnEndTime (页 1218)	RW	-	-	定义使用“TimeColumnIndex”引用的时间列时间范围的终点。
TimeColumnForeColor (页 1218)	RW	-	-	指定使用“TimeColumnIndex”引用的时间列的字体颜色。
TimeColumnHideText (页 1219)	RW	-	-	指定是否将使用“TimeColumnIndex”引用的时间列的内容显示为文本。
TimeColumnHideTitleText (页 1220)	RW	-	-	指定是否将使用“TimeColumnIndex”引用的时间列的标题显示为文本。
TimeColumnIndex (页 1221)	RW	-	-	引用一个已组态的时间列。
TimeColumnLength (页 1221)	RW	-	-	指定使用“TimeColumnIndex”引用的时间列的宽度。
TimeColumnMeasurePoints (页 1222)	RW	-	-	指定要显示在使用“TimeColumnIndex”引用的时间列中的测量点的数量。
TimeColumnName (页 1222)	RW	-	-	指定使用“TimeColumnIndex”引用的时间列的名称。
TimeColumnRangeType (页 1223)	RW	-	-	为使用“TimeColumnIndex”引用的时间列指定时间范围。
TimeColumnRemove (页 1224)	RW	-	-	移除一个引用了其名称的时间列。
TimeColumnRename (页 1224)	RW	-	-	指定使用“TimeColumnIndex”引用的时间列的新名称。
TimeColumnRepos (页 1225)	RW	-	-	对于带有对应数值列的多个时间列，指定使用“TimeColumnIndex”引用的时间列的位置。
TimeColumns	-	-	-	-
TimeColumnShowDate (页 1226)	RW	-	-	指定是否使用日期和时间显示使用“TimeColumnIndex”引用的时间列。
TimeColumnShowIcon (页 1226)	RW	-	-	指定是否将使用“TimeColumnIndex”引用的时间列的内容显示为图标。

属性	RT Professional	RT Advanced	面板 RT	说明
TimeColumnShowTitleIcon (页 1227)	RW	-	-	指定是否将使用"TimeColumnIndex"引用的时间列的标题显示为标题。
TimeColumnSort (页 1228)	RW	-	-	指定使用"TimeColumnIndex"引用的时间列如何排序。
TimeColumnSortIndex (页 1228)	RW	-	-	指定"TimeColumnIndex"中引用的时间列的排序顺序。
TimeColumnTimeFormat (页 1229)	RW	-	-	指定用于显示使用"TimeColumnIndex"引用的时间列的时间格式。
TimeColumnTimeRangeBase (页 1230)	RW	-	-	指定用于计算在使用"TimeColumnIndex"引用的时间列中显示的时间范围的时间单位。
TimeColumnTimeRangeFactor (页 1230)	RW	-	-	指定用于计算时间范围的系数。
TimeColumnUseValueColumnColors (页 1231)	RW	-	-	指定是否以数值列的颜色显示使用"TimeColumnIndex"引用的时间列。
TimeColumnVisible (页 1232)	RW	-	-	指定是否在列表视图中显示使用"TimeColumnIndex"引用的时间列。
TimeStepBase (页 1235)	RW	-	-	指定在列表中显示的时间戳的精度指定时间单位。
TimeStepFactor (页 1235)	RW	-	-	用"TimeStepBase"时间单位指定时间戳的精度。
TitleColor (页 1236)	RW	-	-	指定表格标题的背景色。
TitleCut (页 1237)	RW	-	-	指定列宽过窄时是否缩减标题栏中的字段内容。
TitleDarkShadowColor (页 1238)	RW	-	-	为对象列表中的列和行标题指定 3D 底纹的暗边颜色。
TitleForeColor (页 1239)	RW	-	-	为对象的表格列和行标题指定文本颜色。
TitleGridLineColor (页 1240)	RW	-	-	指定表格标题栏中分隔线的颜色。
TitleLightShadowColor (页 1241)	RW	-	-	为对象的表格列和行标题指定 3D 底纹的亮边颜色。
TitleSort (页 1242)	RW	-	-	指定如何触发按列标题排序。

属性	RT Professional	RT Advanced	面板 RT	说明
TitleStyle (页 1243)	RW	-	-	指定是否使用列标题文本的底纹颜色。
ToolbarAlignment (页 1252)	RW	-	-	指定工具栏的位置。
ToolbarBackColor (页 1253)	RW	-	-	指定工具栏的背景色。
ToolbarButtonActive (页 1254)	RW	-	-	指定是否在运行系统中激活按钮的相应功能。
ToolbarButtonAdd (页 1255)	RW	-	-	在对象的工具栏中创建一个新的用户自定义按钮。
ToolbarButtonAuthorization (页 1256)	RW	-	-	指定所选按键功能的权限。
ToolbarButtonBeginGroup (页 1257)	RW	-	-	在工具栏上，为所选按键功能插入前导分隔符（垂直线）。
ToolbarButtonClick (页 1258)	RW	-	-	单击工具栏按钮。
ToolbarButtonCount (页 1259)	RW	-	-	指定工具栏中的已组态按钮数目。
ToolbarButtonEnabled (页 1260)	RW	-	-	指定是否可以操作用户自定义工具栏按钮。
ToolbarButtonHotKey (页 1261)	RW	-	-	指定所选对象按钮的热键。
ToolbarButtonID (页 1262)	RW	-	-	使用其 ID 引用按钮。
ToolbarButtonIndex (页 1263)	RW	-	-	引用按钮。
ToolbarButtonLocked (页 1264)	RW	-	-	指定是否显示使用“ToolbarButtonIndex”引用的用户自定义按钮的锁定、按下状态。
ToolbarButtonName (页 1265)	RW	-	-	指定使用“ToolbarButtonIndex”引用的用户自定义按钮的名称。
ToolbarButtonRemove (页 1266)	RW	-	-	移除一个引用了其名称的用户自定义按钮。
ToolbarButtonRename (页 1267)	RW	-	-	指定使用“ToolbarButtonIndex”引用的用户自定义按钮的新名称。

属性	RT Professional	RT Advanced	面板 RT	说明
ToolbarButtonRepos (页 1268)	RW	-	-	在对象的工具栏中指定使用“ToolbarButtonIndex”引用的按钮位置。
ToolbarButtons	-	-	-	-
ToolbarButtonTooltipText (页 1270)	RW	-	-	指定工具栏中用户自定义按钮的工具提示文本。
ToolbarButtonUserDefined (页 1271)	RW	-	-	指定项目工程师是否已将此工具栏按钮添加为新的用户自定义按钮。
ToolbarButtonVisible (页 1272)	RW	-	-	指定是否在工具栏中显示使用“ToolbarButtonIndex”引用的按钮。
ToolbarShowTooltips (页 1273)	RW	-	-	指定是否在运行系统中显示按键功能的工具提示。
ToolbarUseBackColor (页 1275)	RW	-	-	指定是否显示工具栏的背景色。
ToolbarUseHotKeys (页 1276)	RW	-	-	指定是否激活工具栏中按钮的热键。
ToolbarVisible (页 1277)	RW	-	-	指定是否显示控件的工具栏。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。
UseColumnBackColor (页 1351)	RW	-	-	指定列的背景色设置。
UseColumnForeColor (页 1352)	RW	-	-	指定列的字体颜色设置。
UseSelectedTitleColor (页 1362)	RW	-	-	指定是否将选择颜色用于选定表格单元格的标题。
UseTableColor2 (页 1364)	RW	-	-	指定是否用第二种行颜色来表现表格。
ValueColumnAdd (页 1393)	RW	-	-	创建新的数值列。
ValueColumnAlignment (页 1394)	RW	-	-	指定使用“ValueColumnIndex”引用的数值列中的文本如何对齐。
ValueColumnAutoPrecisions (页 1394)	RW	-	-	指定是否自动确定使用“ValueColumnIndex”引用的数值轴的小数位数。

属性	RT Professional	RT Advanced	面板 RT	说明
ValueColumnBackColor (页 1395)	RW	-	-	指定使用“ValueColumnIndex”引用的数值列的背景色。
ValueColumnCaption (页 1396)	RW	-	-	指定使用“ValueColumnIndex”引用的数值列的名称。
ValueColumnCount (页 1397)	RW	-	-	指定所组态的数值列数目。
ValueColumnExponentialFormat (页 1397)	RW	-	-	指定是否在指数计数法中显示使用“ValueColumnIndex”引用的数值列的值。
ValueColumnForeColor (页 1398)	RW	-	-	指定使用“ValueColumnIndex”引用的数值列的字体颜色。
ValueColumnHideText (页 1399)	RW	-	-	指定是否隐藏使用“ValueColumnIndex”引用的数值列的文本。
ValueColumnHideTitleText (页 1399)	RW	-	-	指定是否隐藏使用“ValueColumnIndex”引用的数值列标题的文本。
ValueColumnIndex (页 1400)	RW	-	-	引用数值列。
ValueColumnLength (页 1401)	RW	-	-	指定使用“ValueColumnIndex”引用的数值列的宽度（以字符为单位）。
ValueColumnName (页 1401)	RW	-	-	指定使用“ValueColumnIndex”引用的数值列的名称。
ValueColumnPrecisions (页 1402)	RW	-	-	指定使用“ValueColumnIndex”引用的数值列中数值的小数位数。
ValueColumnProvider (页 1402)	RW	-	-	指定使用“ValueColumnIndex”引用的数值列数据源。
ValueColumnProviderCLSID (页 1403)	RW	-	-	指定使用“ValueColumnIndex”引用的数值列的数据源的 CLSID。
ValueColumnRemove (页 1404)	RW	-	-	移除一个引用了其名称的数值列。
ValueColumnRename (页 1405)	RW	-	-	指定使用“ValueColumnIndex”引用的数值列的新名称。
ValueColumnRepos (页 1405)	RW	-	-	指定使用“ValueColumnIndex”引用的数值列的位置。
ValueColumns	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	说明
ValueColumnSelectTagName (页 1406)	RW	-	-	指定用于选择使用“ValueColumnIndex”引用的数值列的数据源变量名称的对话框在运行系统中初次显示。
ValueColumnShowIcon (页 1407)	RW	-	-	指定是否在使用“ValueColumnIndex”引用的数值列中显示图标。
ValueColumnShowTitleIcon (页 1408)	RW	-	-	指定是否在使用“ValueColumnIndex”引用的数值列的标题中显示图标。
ValueColumnSort (页 1408)	RW	-	-	指定使用“ValueColumnIndex" ”引用的数值列的排序类型。
ValueColumnSortIndex (页 1409)	RW	-	-	指定排序顺序。
ValueColumnTagName (页 1410)	RW	-	-	指定其数值显示在使用“ValueColumnIndex”引用的数值列中的变量的名称。
ValueColumnTimeColumn (页 1410)	RW	-	-	指定相应时间列。
ValueColumnVisible (页 1411)	RW	-	-	指定是否在对象中显示使用“ValueColumnIndex”引用的数值列。
VerticalGridLines (页 1414)	RW	-	-	指定是否显示垂直分隔线。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-59　方法

方法	RT Professional	RT Advanced	面板 RT	说明
激活 (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
AttachDB (页 1477)	√	-	-	执行控件的“连接备份”键功能。

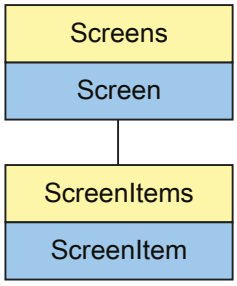
方法	RT Professional	RT Advanced	面板 RT	说明
CalculateStatistic (页 1478)	√	-	-	执行 f(t) 趋势视图和表格视图的“计算统计数据”键功能。
CopyRows (页 1479)	√	-	-	执行控件的“复制行”键功能。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。
DetachDB (页 1484)	√	-	-	执行控件的“断开备份”键功能。
编辑 (页 1485)	√	-	-	执行表格视图的“编辑”键功能。
导出 (页 1485)	√	-	-	执行控件的“导出日志”(Export log) 或“导出数据”(Export data) 按键功能。
GetOperatorMessage (页 1496)	√	-	-	以“ICCAxOperatorMessage”类型的形式返回报警视图中按名称或索引指定的操作员消息对象。
GetOperatorMessageCollection	√	-	-	-
GetRow (页 1498)	√	-	-	以“ICCAxDataRow”类型的形式返回基于表格的控件中的行编号所指定的行对象。
GetRowCollection (页 1499)	√	-	-	以“ICCAxDataRowCollection”类型的形式返回基于表格的控件的所有行对象的列表。
GetSelectedRow (页 1507)	√	-	-	以“ICCAxDataRow”类型的形式返回基于表格的控件的所选行对象。
GetSelectedRows (页 1508)	√	-	-	以“ICCAxDataRow”类型的形式返回用于多项选择的基于表格的控件的所选行对象。
GetStatusbarElement (页 1514)	√	-	-	以“ICCAxStatusbarElement”类型的形式返回按名称或索引指定的控件状态栏元素。
GetStatusbarElementCollection (页 1516)	√	-	-	以类型“ICCAxCollection”形式返回控件的所有状态栏元素的列表。
GetTimeColumn (页 1521)	√	-	-	以“ICCAxTimeColumn”类型的形式返回表视图中具有指定名称或索引的时间列对象。
GetTimeColumnCollection (页 1522)	√	-	-	以“ICCAxCollection”类型的形式返回表视图中所有时间列对象的列表。
GetToolbarButton (页 1523)	√	-	-	以“ICCAxToolbarButton”类型的形式返回控件工具栏上按名称或索引指定的按钮功能。

方法	RT Professional	RT Advanced	面板 RT	说明
GetToolbarButtonCollection (页 1525)	√	-	-	以“ICCAxCollection”类型的形式返回控件工具栏的所有按钮功能的列表。
GetValueColumn (页 1534)	√	-	-	以“ICCAxValueColumn”类型的形式返回 f(t) 趋势视图中具有指定名称或索引的数值列对象。
GetValueColumnCollection (页 1536)	√	-	-	以“ICCAxCollection”类型的形式返回 f(t) 趋势视图中所有数值列对象的列表。
MoveToFirst (页 1547)	√	-	-	执行控件的“第一行”(First line) 按钮功能。
MoveToLast (页 1549)	√	-	-	执行控件的“最后一条数据记录”(Last data record) 按钮功能。
MoveToNext (页 1551)	√	-	-	执行控件的“下一条数据记录”(Next data record) 按钮功能。
MoveToPrevious (页 1552)	√	-	-	执行控件的“前一条数据记录”(Previous data record) 按钮功能。
NextColumn (页 1554)	√	-	-	执行表视图的“下一列”(Next column) 按钮功能。
PreviousColumn (页 1556)	√	-	-	执行表视图的“前一列”(Previous column) 按钮功能。
Print (页 1557)	√	-	-	执行控件的“打印”(Print) 按钮功能。
SelectAll (页 1573)	√	-	-	在基于表格的控件中选择所有行。
SelectRow (页 1575)	√	-	-	在基于表格的控件中选择特定行。
SelectStatisticArea (页 1574)	√	-	-	执行表格视图的“设置统计数据范围”(Set statistics range) 按钮功能。
ShowColumnSelection (页 1578)	√	-	-	执行表视图的“选择列”(Select columns) 按钮功能。
ShowHelp (页 1580)	√	-	-	执行控件的“帮助”(Help) 按钮功能。
ShowInsertValueDialog	√	-	-	-
ShowPropertyDialog (页 1585)	√	-	-	执行控件的“组态对话框”(Configuration dialog) 按钮功能。
ShowTagSelection (页 1589)	√	-	-	执行控件的“选择数据连接”(Select data connection) 按钮功能。

方法	RT Professional	RT Advanced	面板 RT	说明
ShowTimeSelection (页 1590)	√	-	-	执行控件的“选择时间范围”(Select time range) 按钮功能。
StartStopUpdate (页 1592)	√	-	-	执行控件的“开始”(Start) 或“停止”(Stop) 按钮功能。
UnselectAll (页 1595)	√	-	-	在基于表格的控件的单元格中移除所有选择内容。
UnselectRow (页 1595)	√	-	-	在基于表格的控件的特定单元格中移除选择内容。

OnlineTrendControl

说明



表示“f(t) TrendView”对象。OnlineTrendControl 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIOnlineTrendControl

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-60　属性

属性	RT Professional	RT Advanced	面板 RT	说明
AllTagTypesAllowed	-	-	-	-
ApplyProjectSettingsForDesignMode	-	-	-	-
BackColor (页 637)	RW	-	-	指定背景色。
BorderColor (页 678)	RW	-	-	指定线颜色。
BorderWidth (页 697)	RW	-	-	指定线宽。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Caption (页 708)	RW	-	-	指定标题中将要显示的文本。
Closeable (页 722)	RW	-	-	指定是否可在运行系统中关闭对象。
ConnectTrendWindows (页 760)	RW	-	-	指定是否连接已组态的趋势视图。
ControlDesignMode (页 762)	RW	-	-	指定控制设计。
Enabled	-	-	-	-
ExportDelimiter	-	-	-	-
ExportDirectoryChangeable (页 808)	RW	-	-	指定数据导出目录是否可在运行时更改。
ExportDirectoryname (页 809)	RW	-	-	指定导出的运行系统数据的目标目录。

属性	RT Professional	RT Advanced	面板 RT	说明
ExportFileExtension (页 810)	RW	-	-	指定导出文件的文件扩展名。
ExportFilename (页 811)	RW	-	-	指定导出的运行系统数据的目标文件名称。
ExportFilenameChangeable (页 812)	RW	-	-	指定是否可以在运行系统中更改导出文件名。
ExportFormat	-	-	-	-
ExportFormatGuid (页 813)	RW	-	-	指定 ID 编号和导出数据源的分配。
ExportFormatName (页 814)	RW	-	-	指定导出文件格式。
ExportParameters (页 815)	RW	-	-	通过属性对话框指定所选格式的参数。
ExportSelection (页 816)	RW	-	-	指定导出控件的哪些运行系统数据。
ExportShowDialog (页 817)	RW	-	-	指定是否在运行系统中显示数据导出对话框。
Font (页 846)	RW	-	-	指定字体。
GraphDirection (页 859)	RW	-	-	指定当前值在趋势视图的哪个边框上显示。
Height (页 863)	RW	-	-	指定高度。
Layer	-	-	-	-
Left (页 920)	RW	-	-	指定 X 坐标的值。
LineColor (页 939)	RW	-	-	指定窗口分隔线的颜色。
LineWidth (页 943)	RW	-	-	指定线宽。
LoadDataImmediately (页 946)	RW	-	-	指定是否下载日志中记录的在调用画面时显示的时间范围的变量值。
Location	-	-	-	-
MaximumNumberOfTimeAxes	-	-	-	-
MaximumNumberOfValueAxes	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	说明
MinimumNumberOfTimeAxes	-	-	-	-
MinimumNumberOfValueAxes	-	-	-	-
Moveable (页 983)	RW	-	-	指定关于运行系统中是否可以移动对象的信息。
Name	-	-	-	-
Object	-	-	-	-
OcxGuid	-	-	-	-
OcxState	-	-	-	-
OcxStateForEs2Rt	-	-	-	-
Online (页 999)	RW	-	-	指定更新的起始与停止。
PercentageAxis (页 1024)	RW	-	-	指定在趋势视图中显示另外一个具有百分比刻度的轴。
PercentageAxisAlignment (页 1025)	RW	-	-	指定百分比轴对齐方式。
PercentageAxisColor (页 1026)	RW	-	-	指定百分比轴的字体和线颜色。
PrintJob (页 1036)	RW	-	-	指定一个在“报表”编辑器中创建的打印作业。
RTPersistence (页 1054)	RW	-	-	指定在画面更改后是否保留在线组态。
RTPersistenceAuthorization (页 1055)	RW	-	-	指定运行系统中在线组态所需的权限。
RTPersistenceType (页 1057)	RW	-	-	指定如何保留 WinCC 的在线组态。
ShowRuler (页 1114)	RW	-	-	指定是否为轴标签显示刻度分度（帮助线）。
ShowRulerInAxis (页 1114)	RW	-	-	指定是否在时间轴中显示标尺。
ShowScrollbars (页 1116)	RW	-	-	指定是否显示滚动条。
ShowStatisticRuler (页 1119)	RW	-	-	指定是否显示定义统计数据范围的线。

属性	RT Professional	RT Advanced	面板 RT	说明
ShowTitle (页 1124)	RW	-	-	为对象指定窗口边框和窗口标题的样式。
ShowTrendIcon (页 1126)	RW	-	-	指定是否在数值轴下显示图标。
Size	-	-	-	-
Sizeable (页 1129)	RW	-	-	指定可以在运行系统中更改对象的大小。
StatusbarBackColor (页 1138)	RW	-	-	指定状态栏的背景色。
StatusbarElementAdd (页 1139)	RW	-	-	创建一个新的用户自定义状态栏元素。
StatusbarElementAutoSize (页 1140)	RW	-	-	指定是否自动设置所选状态栏元素的宽度。
StatusbarElementCount (页 1141)	RW	-	-	指定已组态的状态栏元素的数量。
StatusbarElementIconId (页 1142)	RW	-	-	指定状态栏元素的 ID 编号和符号分配。
StatusbarElementID (页 1143)	RW	-	-	使用其元素 ID 来引用状态栏元素。
StatusbarElementIndex (页 1144)	RW	-	-	引用状态栏元素。
StatusbarElementName (页 1145)	RW	-	-	指定使用“StatusbarElementIndex”引用的状态栏元素的名称。
StatusbarElementRemove (页 1146)	RW	-	-	删除应用其名称的用户自定义状态栏元素。
StatusbarElementRename (页 1147)	RW	-	-	指定使用“StatusbarElementIndex”引用的用户自定义的状态栏元素的新名称。
StatusbarElementRepos (页 1148)	RW	-	-	指定对象状态栏中使用“StatusbarElementIndex”引用的元素位置。
StatusbarElements	-	-	-	-
StatusbarElementText (页 1149)	RW	-	-	指定所选状态栏元素的文本。
StatusbarElementTooltipText (页 1150)	RW	-	-	指定所选用户自定义状态栏元素的工具提示文本。

属性	RT Professional	RT Advanced	面板 RT	说明
StatusbarElementUserDefined (页 1151)	RW	-	-	指定组态工程师是否已添加状态栏元素作为新的用户自定义元素。
StatusbarElementVisible (页 1152)	RW	-	-	指定是否在对象中显示使用“StatusbarElementIndex”引用的状态栏元素。
StatusbarElementWidth (页 1153)	RW	-	-	指定所引用的状栏元素的宽度（以像素为单位）。
StatusbarFont (页 1154)	RW	-	-	指定状态栏中文本的字体。
StatusbarFontColor (页 1155)	RW	-	-	指定状态栏中文本的颜色。
StatusbarShowTooltips (页 1156)	RW	-	-	指定是否在运行系统中显示状态栏元素的工具提示。
StatusbarText (页 1157)	RW	-	-	指定状态栏中的默认文本。
StatusbarUseBackColor (页 1158)	RW	-	-	指定是否显示状态栏的背景色。
StatusbarVisible (页 1159)	RW	-	-	指定是否显示控件的状态栏。
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TimeAxes	-	-	-	-
TimeAxisAdd (页 1195)	RW	-	-	创建新的时间轴。新创建的时间轴会自动使用“TimeAxisIndex”而引用。
TimeAxisAlignment (页 1196)	RW	-	-	指定时间轴对齐方式。
TimeAxisBeginTime (页 1197)	RW	-	-	指定显示所选趋势的开始时间。
TimeAxisColor (页 1197)	RW	-	-	指定时间轴的颜色。
TimeAxisCount (页 1198)	RW	-	-	指定已组态的时间轴数。

属性	RT Professional	RT Advanced	面板 RT	说明
TimeAxisDateFormat (页 1199)	RW	-	-	定义用于显示所选时间轴的日期格式。
TimeAxisEndTime (页 1200)	RW	-	-	指定显示所选趋势的结束时间。
TimeAxisIndex (页 1200)	RW	-	-	引用时间轴。
TimeAxisInTrendColor (页 1201)	RW	-	-	指定使用"TimeAxisIndex"引用的轴的颜色是否与趋势颜色相对应。
TimeAxisLabel (页 1202)	RW	-	-	指定使用"TimeAxisIndex"引用的 x 轴的标签文本。
TimeAxisMeasurePoints (页 1202)	RW	-	-	为所选时间轴指定要显示的测量点数量。
TimeAxisName (页 1203)	RW	-	-	指定使用"TimeAxisIndex"引用的时间轴的名称。
TimeAxisOnline (页 1204)	RW	-	-	指定时间轴持续更新。
TimeAxisRangeType (页 1204)	RW	-	-	定义所选时间轴的时间范围设置。
TimeAxisRemove (页 1205)	RW	-	-	移除一个引用了其名称的时间轴。
TimeAxisRename (页 1206)	RW	-	-	指定使用"TimeAxisIndex"引用的时间轴的新名称。
TimeAxisRepos (页 1206)	RW	-	-	在对象的趋势图中指定使用"TimeAxisIndex"引用的时间轴位置。
TimeAxisShowDate (页 1207)	RW	-	-	指定是否使用日期和时间显示所选时间轴。
TimeAxisTimeFormat (页 1208)	RW	-	-	指定沿所选趋势的时间轴的信息格式。
TimeAxisTimeRangeBase (页 1209)	RW	-	-	指定用于定义时间范围和时间因素"TimeAxisTimeRangeFactor"的时间单位。
TimeAxisTimeRangeFactor (页 1209)	RW	-	-	指定用于定义时间段和时间单位"TimeAxisTimeRangeBase"的时间因素。

属性	RT Professional	RT Advanced	面板 RT	说明
TimeAxisTrendWindow (页 1210)	RW	-	-	指定使用"TimeAxisIndex"引用的轴在其中显示的趋势图。
TimeAxisVisible (页 1211)	RW	-	-	指定是否在对象中显示使用"TimeAxisIndex"引用的时间轴。
TimeBase (页 1211)	RW	-	-	指定显示时间值的时区。
ToolbarAlignment (页 1252)	RW	-	-	指定工具栏的位置。
ToolbarBackColor (页 1253)	RW	-	-	指定工具栏的背景色。
ToolbarButtonActive (页 1254)	RW	-	-	指定是否在运行系统中激活按钮的相应功能。
ToolbarButtonAdd (页 1255)	RW	-	-	在对象的工具栏中创建一个新的用户自定义按钮。
ToolbarButtonAuthorization (页 1256)	RW	-	-	指定所选按键功能的权限。
ToolbarButtonBeginGroup (页 1257)	RW	-	-	在工具栏上，为所选按键功能插入前导分隔符（垂直线）。
ToolbarButtonClick (页 1258)	RW	-	-	单击工具栏按钮。
ToolbarButtonCount (页 1259)	RW	-	-	指定工具栏中的已组态按钮数目。
ToolbarButtonEnabled (页 1260)	RW	-	-	指定是否可以操作用户自定义工具栏按钮。
ToolbarButtonHotKey (页 1261)	RW	-	-	为对象的所选按钮指定快捷方式。
ToolbarButtonID (页 1262)	RW	-	-	使用其 ID 引用按钮。
ToolbarButtonIndex (页 1263)	RW	-	-	引用按钮。
ToolbarButtonLocked (页 1264)	RW	-	-	指定是否显示使用"ToolbarButtonIndex"引用的用户自定义按钮的锁定、按下状态。
ToolbarButtonName (页 1265)	RW	-	-	指定使用"ToolbarButtonIndex"引用的用户自定义按钮的名称。

属性	RT Professional	RT Advanced	面板 RT	说明
ToolbarButtonRemove (页 1266)	RW	-	-	移除一个引用了其名称的用户自定义按钮。
ToolbarButtonRename (页 1267)	RW	-	-	指定使用"ToolbarButtonIndex"引用的用户自定义按钮的新名称。
ToolbarButtonRepos (页 1268)	RW	-	-	在对象的工具栏中指定使用"ToolbarButtonIndex"引用的按钮的位置。
ToolbarButtons	-	-	-	-
ToolbarButtonTooltipText (页 1270)	RW	-	-	指定工具栏中用户自定义按钮的工具提示文本。
ToolbarButtonUserDefined (页 1271)	RW	-	-	指定项目工程师是否已将此工具栏按钮添加为新的用户自定义按钮。
ToolbarButtonVisible (页 1272)	RW	-	-	指定是否在工具栏中显示使用"ToolbarButtonIndex"引用的按钮。
ToolbarShowTooltips (页 1273)	RW	-	-	指定是否在运行系统中显示按键功能的工具提示。
ToolbarUseBackColor (页 1275)	RW	-	-	指定是否显示工具栏的背景色。
ToolbarUseHotKeys (页 1276)	RW	-	-	指定是否激活工具栏中按钮的热键。
ToolbarVisible (页 1277)	RW	-	-	指定是否显示控件的工具栏。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。
TrendAdd (页 1290)	RW	-	-	创建新趋势。
TrendAutoRangeBeginTagName (页 1290)	RW	-	-	指定定义趋势数据区域的起始值的变量。
TrendAutoRangeBeginValue (页 1291)	RW	-	-	指定趋势数据区域的起始值。
TrendAutoRangeEndTagName (页 1291)	RW	-	-	指定定义趋势数据区域的结束值的变量。
TrendAutoRangeEndValue (页 1292)	RW	-	-	指定趋势数据区域的结束值。

属性	RT Professional	RT Advanced	面板 RT	说明
TrendAutoRangeSource (页 1293)	RW	-	-	指定如何确定趋势数据的自动数据范围。
TrendColor (页 1294)	RW	-	-	指定趋势视图中引用的趋势的颜色。
TrendCount (页 1295)	RW	-	-	指定所组态趋势的数目。
TrendExtendedColorSet (页 1296)	RW	-	-	指定是否显示趋势的点和填充颜色。
TrendFill (页 1297)	RW	-	-	指定趋势下方的区域是否显示为已填充。
TrendFillColor (页 1298)	RW	-	-	定义趋势的填充颜色。
TrendIndex (页 1299)	RW	-	-	引用一个已组态的趋势。
TrendLabel (页 1300)	RW	-	-	为使用“TrendIndex”引用的趋势定义标签文本。
TrendLineStyle (页 1301)	RW	-	-	指定用于趋势显示的线类型。
TrendLineType (页 1302)	RW	-	-	指定如何显示趋势。
TrendLineWidth (页 1303)	RW	-	-	指定所选趋势的线宽（以像素为单位）。
TrendLowerLimit (页 1303)	RW	-	-	指定在对象中作为趋势显示的变量下限。
TrendLowerLimitColor (页 1304)	RW	-	-	指定用于标记低于“TrendLowerLimit”值的趋势值的颜色。
TrendLowerLimitColoring (页 1305)	RW	-	-	指定是否使用“TrendLowerLimitColor”标识小于“TrendLowerLimit”值的变量值。
TrendName (页 1306)	RW	-	-	指定使用“TrendIndex”引用的趋势的名称。
TrendPointColor (页 1307)	RW	-	-	指定引用趋势上的点的颜色。
TrendPointStyle (页 1308)	RW	-	-	指定如何显示趋势点。
TrendPointWidth (页 1309)	RW	-	-	指定点宽度（以像素为单位）。

属性	RT Professional	RT Advanced	面板 RT	说明
TrendProvider (页 1309)	RW	-	-	指定所选趋势的数据源。
TrendProviderCLSID (页 1310)	RW	-	-	指定趋势中数据源的 CLSID。
TrendRemove (页 1312)	RW	-	-	移除一个引用了其名称的趋势。
TrendRename (页 1313)	RW	-	-	指定使用"TrendIndex"引用的趋势的新名称。
TrendRepos (页 1313)	RW	-	-	在对象的趋势窗口中指定使用"TrendIndex"引用的趋势位置。
Trends	-	-	-	-
TrendSelectTagName (页 1314)	RW	-	-	指定用于选择数值轴数据源的变量名称的对话框在运行系统中初次显示。
TrendTagName (页 1316)	RW	-	-	指定用数据补充数值轴的变量名称。
TrendTimeAxis (页 1318)	RW	-	-	指定要用于所选趋势的时间轴。
TrendTrendWindow (页 1320)	RW	-	-	指定显示所选趋势的趋势窗口。
TrendUncertainColor (页 1321)	RW	-	-	指定不确定状态的值的颜色。
TrendUncertainColoring (页 1321)	RW	-	-	指定不确定状态的值将突出显示。
TrendUpperLimit (页 1322)	RW	-	-	指定在特定对象中作为趋势显示的变量上限。
TrendUpperLimitColor (页 1323)	RW	-	-	指定用于标记高于"TrendUpperLimit"值的趋势值的颜色。
TrendUpperLimitColoring (页 1324)	RW	-	-	指定是否使用系统定义的颜色显示选择框架。
TrendValueAlign	-	-	-	-
TrendValueAxis (页 1325)	RW	-	-	指定用于所选趋势的数值轴。

属性	RT Professional	RT Advanced	面板 RT	说明
TrendValueUnit (页 1325)	RW	-	-	对于趋势类型“显示值”，指定之后附加到将显示的值的数值单位，例如，“%”或“°C”。
TrendVisible (页 1326)	RW	-	-	指定是否在对象中显示使用“"TrendIndex”引用的趋势。
TrendWindowAdd (页 1327)	RW	-	-	创建新的趋势视图。
TrendWindowCoarseGrid (页 1327)	RW	-	-	指定是否显示主刻度的网格线。
TrendWindowCoarseGridColor (页 1328)	RW	-	-	指定对象中引用图的主网格的颜色。
TrendWindowCount (页 1329)	RW	-	-	在趋势视图中指定所组态的趋势图。
TrendWindowFineGrid (页 1330)	RW	-	-	指定是否对次刻度显示网格线。
TrendWindowFineGridColor (页 1330)	RW	-	-	指定对象中引用图的辅助网格的颜色。
TrendWindowForegroundTrendGrid (页 1331)	RW	-	-	指定是否在所选趋势窗口中仅显示前景趋势的网格线。
TrendWindowGridInTrendColor (页 1332)	RW	-	-	指定是否使用趋势颜色显示主刻度的网格线。
TrendWindowHorizontalGrid (页 1333)	RW	-	-	指定是否显示水平网格线。
TrendWindowIndex (页 1333)	RW	-	-	引用趋势视图。要访问一个趋势视图的属性，需要设置“TrendWindowIndex”。
TrendWindowName (页 1334)	RW	-	-	指定使用“TrendWindowIndex”引用的趋势视图的名称。
TrendWindowRemove (页 1335)	RW	-	-	移除一个引用了其名称的趋势视图。
TrendWindowRename (页 1335)	RW	-	-	指定使用“TrendWindowIndex”引用的趋势视图的新名称。
TrendWindowRepos (页 1336)	RW	-	-	指定使用“TrendWindowIndex”引用的趋势视图的位置。

属性	RT Professional	RT Advanced	面板 RT	说明
TrendWindowRulerColor (页 1337)	RW	-	-	指定标尺的颜色。
TrendWindowRulerLayer (页 1338)	RW	-	-	指定趋势视图中标尺的显示层。
TrendWindowRulerStyle (页 1339)	RW	-	-	指定标尺的外观。
TrendWindowRulerWidth (页 1339)	RW	-	-	指定标尺的宽度（以像素为单位）。
TrendWindows	-	-	-	-
TrendWindowSpacePortion (页 1340)	RW	-	-	在对象的图表区域中，指定引用图的范围比例。
TrendWindowStatisticRulerColor (页 1341)	RW	-	-	指定统计数据标尺的颜色。
TrendWindowStatisticRulerStyle (页 1342)	RW	-	-	定义线的图形组态能力，以便在趋势窗口中指定统计数据区域。
TrendWindowStatisticRulerWidth (页 1343)	RW	-	-	指定统计数据标尺的线宽。
TrendWindowVerticalGrid (页 1343)	RW	-	-	指定是否显示垂直网格线。
TrendWindowVisible (页 1344)	RW	-	-	指定是否显示使用“TrendWindowIndex”引用的趋势视图。
UseTrendNameAsLabel (页 1369)	RW	-	-	指定使用“名称”还是“标签”属性作为运行系统中趋势的标识。
ValueAxes	-	-	-	-
ValueAxisAdd (页 1379)	RW	-	-	创建新的数值轴。
ValueAxisAlignment (页 1380)	RW	-	-	指定数值轴对齐方式。
ValueAxisAutoPrecisions (页 1381)	RW	-	-	指定是否自动确定使用“ValueAxisIndex”引用的数值轴的小数位数。
ValueAxisAutoRange (页 1381)	RW	-	-	指定是否自动计算使用“ValueAxisIndex”引用的数值轴的取值范围。

属性	RT Professional	RT Advanced	面板 RT	说明
ValueAxisBeginValue (页 1382)	RW	-	-	指定使用"ValueAxisIndex"引用数值轴的取值范围下限。
ValueAxisColor (页 1383)	RW	-	-	指定数值轴的颜色。
ValueAxisCount (页 1384)	RW	-	-	指定所组态的数值轴数目。
ValueAxisEndValue (页 1384)	RW	-	-	指定使用"ValueAxisIndex"引用数值轴的取值范围上限。
ValueAxisExponentialFormat (页 1385)	RW	-	-	指定是否在指数计数法中显示使用"ValueAxisIndex"引用的数值轴的值。
ValueAxisIndex (页 1386)	RW	-	-	引用数值轴。
ValueAxisInTrendColor (页 1386)	RW	-	-	指定使用"ValueAxisIndex"引用的轴的颜色是否与趋势颜色相对应。
ValueAxisLabel (页 1387)	RW	-	-	为使用"ValueAxisIndex"引用的数值轴定义标签文本。
ValueAxisName (页 1388)	RW	-	-	指定使用"ValueAxisIndex"引用的数值轴的名称。
ValueAxisPrecisions (页 1388)	RW	-	-	指定显示的小数位数。
ValueAxisRemove (页 1389)	RW	-	-	移除一个引用了其名称的数值轴。
ValueAxisRename (页 1389)	RW	-	-	指定使用"ValueAxisIndex"引用的数值轴的新名称。
ValueAxisRepos (页 1390)	RW	-	-	指定使用"ValueAxisIndex"引用的数值轴的位置。
ValueAxisScalingType (页 1391)	RW	-	-	指定使用"ValueAxisIndex"引用的数值轴比例缩放的类型。
ValueAxisTrendWindow (页 1392)	RW	-	-	指定使用"ValueAxisIndex"引用的轴在其中显示的趋势图。
ValueAxisVisible (页 1392)	RW	-	-	指定是否在对象中显示使用"ValueAxisIndex"引用的数值轴。

属性	RT Professional	RT Advanced	面板 RT	说明
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-61　方法

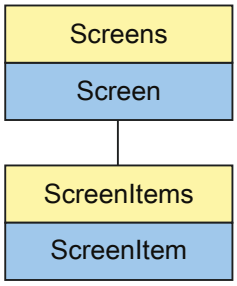
方法	RT Professional	RT Advanced	面板 RT	说明
激活 (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
AttachDB (页 1477)	√	-	-	执行控件的“连接备份”键功能。
CalculateStatistic (页 1478)	√	-	-	执行 f(t) 趋势视图和表格视图的“计算统计数据”键功能。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。
DetachDB (页 1484)	√	-	-	执行控件的“断开备份”键功能。
导出 (页 1485)	√	-	-	执行控件的“导出日志”(Export log) 或“导出数据”(Export data) 按键功能。
GetRulerData (页 1506)	√	-	-	返回标尺位置处被调用趋势的值。
GetStatusbarElement (页 1514)	√	-	-	以“ICCAxStatusbarElement”类型的形式返回按名称或索引指定的控件状态栏元素。
GetStatusbarElementCollection (页 1522)	√	-	-	以类型“ICCAxCollection”形式返回控件的所有状态栏元素的列表。
GetTimeAxis (页 1517)	√	-	-	以“ICCAxTimeAxis”类型的形式返回具有指定名称或索引的 f(t) 趋势视图时间轴对象。
GetTimeAxisCollection (页 1519)	√	-	-	以“ICCAxCollection”类型的形式返回所有 f(t) 趋势视图时间轴对象的列表。
GetToolbarButton (页 1523)	√	-	-	以“ICCAxToolbarButton”类型的形式返回控件工具栏上按名称或索引指定的按钮功能。
GetToolbarButtonCollection (页 1525)	√	-	-	以“ICCAxCollection”类型的形式返回控件工具栏的所有按钮功能的列表。

方法	RT Professional	RT Advanced	面板 RT	说明
GetTrend (页 1526)	√	-	-	以“ICCAxTrend”或“ICCAxFunctionTrend”类型的形式返回按名称或索引指定的 f(t) 或 f(x) 趋势视图趋势。
GetTrendCollection (页 1527)	√	-	-	以“ICCAxCollection”类型的形式返回 f(t) 或 f(x) 趋势视图的所有趋势的列表。
GetTrendWindow (页 1529)	√	-	-	以“ICCAxTrendWindow”类型的形式返回按名称或索引指定的 f(t) 或 f(x) 趋势视图的趋势窗口对象。
GetTrendWindowCollection (页 1530)	√	-	-	以“ICCAxCollection”类型的形式返回 f(t) 或 f(x) 趋势视图的所有趋势窗口对象的列表。
GetValueAxis (页 1531)	√	-	-	以“ICCAxValueAxis”类型的形式返回具有指定名称或索引的 f(t) 趋势视图数值轴对象。
GetValueAxisCollection (页 1533)	√	-	-	以“ICCAxCollection”类型的形式返回所有 f(t) 趋势视图数值轴对象的列表。
MoveAxis (页 1547)	√	-	-	执行 f(t) 和 f(x) 趋势视图的“移动轴区域”(Move axes area) 按钮功能。
MoveRuler	√	-	-	
MoveToFirst (页 1547)	√	-	-	执行控件的“第一行”(First line) 按钮功能。
MoveToLast (页 1549)	√	-	-	执行控件的“最后一条数据记录”(Last data record) 按钮功能。
MoveToNext (页 1551)	√	-	-	执行控件的“下一条数据记录”(Next data record) 按钮功能。
MoveToPrevious (页 1552)	√	-	-	执行控件的“前一条数据记录”(Previous data record) 按钮功能。
NextTrend (页 1554)	√	-	-	执行 f(t) 和 f(x) 趋势视图的“下一个趋势”(Next trend) 按钮功能。
OneToOneView (页 1555)	√	-	-	执行 f(t) 和 f(x) 趋势视图的“原始视图”(Original view) 按钮功能。
PreviousTrend (页 1556)	√	-	-	执行 f(t) 和 f(x) 趋势视图的“上一个趋势”(Previous trend) 按钮功能。
Print (页 1557)	√	-	-	执行控件的“打印”(Print) 按钮功能。
ShowHelp (页 1580)	√	-	-	执行控件的“帮助”(Help) 按钮功能。

方法	RT Professional	RT Advanced	面板 RT	说明
ShowPercentageAxis (页 1585)	√	-	-	执行 f(t) 趋势视图的“相对轴”(Relative axis) 按钮功能。
ShowPropertyDialog (页 1585)	√	-	-	执行控件的“组态对话框”(Configuration dialog) 按钮功能。
ShowTagSelection (页 1589)	√	-	-	执行控件的“选择数据连接”(Select data connection) 按钮功能。
ShowTimeSelection (页 1590)	√	-	-	执行控件的“选择时间范围”(Select time range) 按钮功能。
ZoomArea (页 1600)	√	-	-	执行 f(t) 和 f(x) 趋势视图的“缩放区域”(Zoom area) 按钮功能。
ZoomInOut (页 1600)	√	-	-	执行 f(t) 和 f(x) 趋势视图的“缩放”(Zoom +/-) 按钮功能。
ZoomInOutTime (页 1601)	√	-	-	执行 f(t) 趋势视图的“缩放时间轴”(Zoom time axis +/-) 按钮功能。
ZoomInOutValues (页 1602)	√	-	-	执行 f(t) 趋势视图的“缩放数值轴”(Zoom value axis +/-) 按钮功能。
ZoomMove (页 1603)	√	-	-	执行 f(t) 和 f(x) 趋势视图的“移动趋势区域”(Move trend area) 按钮功能。

OptionGroup

描述



表示“Option button”对象。OptionGroup 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIOptionGroup

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-62　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AdaptBorder (页 606)	RW	-	-	指定对象的边框是否根据文本的大小进行动态调整。
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定操作权限。
BackColor (页 637)	RW	-	-	指定背景色。
BackFillStyle (页 643)	RW	-	-	指定填充图案。
BackFlashingColorOff (页 645)	RW	-	-	指定闪烁状态“关闭”(Off) 的背景色。
BackFlashingColorOn (页 646)	RW	-	-	指定闪烁状态“打开”(On) 的背景色。
BackFlashingEnabled (页 648)	RW	-	-	指定是否在运行系统中闪烁背景。
BackFlashingRate (页 649)	RW	-	-	指定背景的闪烁频率。
BorderBackColor (页 674)	RW	-	-	指定间断边框线的背景色。
BorderColor (页 678)	RW	-	-	指定线颜色。
BorderFlashingColorOff (页 683)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
BorderFlashingColorOn (页 685)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
BorderFlashingEnabled (页 687)	RW	-	-	指定是否在运行系统中闪烁对象的限值。
BorderFlashingRate (页 689)	RW	-	-	指定边界线的闪烁频率。
BorderStyle	-	-	-	-
BorderWidth (页 697)	RW	-	-	指定线宽。

属性	RT Professional	RT Advanced	面板 RT	描述
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CheckMarkAlignment (页 720)	RW	-	-	指定各域是否右对齐。
CheckMarkCount (页 720)	RW	-	-	指定字段数量。
CornerStyle (页 764)	RW	-	-	指定角形状。
DrawInsideFrame (页 787)	RW	-	-	指定线宽大于 1 的将要显示的边框线是在边框内，还是与边框对称。
EdgeStyle (页 788)	RW	-	-	指定线样式。
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。
FillingDirection (页 825)	RW	-	-	指定填充方向。
FillPatternColor (页 821)	RW	-	-	确定填充样式的颜色。
Flashing	-	-	-	-
FlashingColorOff (页 832)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
FlashingColorOn (页 834)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
FlashingEnabled (页 836)	RW	-	-	指定是否为运行系统中的对象激活闪烁。
FlashingRate (页 839)	RW	-	-	指定边界线的闪烁频率。
Font	-	-	-	-
FontBold (页 849)	RW	-	-	指定文本是否以粗体显示。
FontItalic (页 849)	RW	-	-	指定文本是否以斜体显示。
FontName (页 851)	RW	-	-	指定字体。
FontSize (页 852)	RW	-	-	指定文本字体大小。
FontUnderline (页 853)	RW	-	-	指定文本是否应以下划线标出。
ForeColor (页 854)	RW	-	-	指定文本字体颜色。

属性	RT Professional	RT Advanced	面板 RT	描述
Height (页 863)	RW	-	-	指定高度。
HorizontalAlignment (页 881)	RW	-	-	指定文本水平对齐。
Index (页 887)	RW	-	-	指定所选文本域的索引。
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
Location	-	-	-	-
LogOperation (页 948)	RW	-	-	指定是否在操作对象后将报警输出到报警系统。
Name	-	-	-	-
ProcessValue (页 1037)	RW	-	-	指定要显示的数值的默认值。
RelativeFillLevel (页 1046)	RW	-	-	指定对象的填充百分比。
ShowBadTagState (页 1099)	RW	-	-	定义检测到不良质量代码或变量状态时对象是否变成灰色。
ShowFillLevel (页 1108)	RW	-	-	指定所选对象是否填充。
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Text (页 1185)	RW	-	-	指定文本域的标签。
TextHandles	-	-	-	-
TextOrientation (页 1189)	RW	-	-	指定文本方向。
Texts	-	-	-	-
ToolTipText (页 1278)	RW	-	-	指定工具提示文本。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。

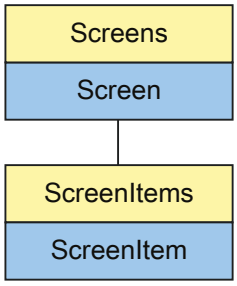
属性	RT Professional	RT Advanced	面板 RT	描述
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
VerticalAlignment (页 1413)	RW	-	-	指定所选对象中的文本垂直对齐方式。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-63　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

PDFview

描述



表示“PDF view”对象。PDFview 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIPDFview

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-64　属性

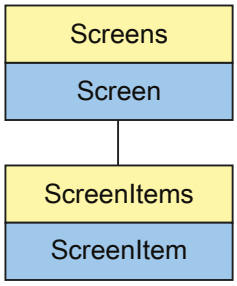
属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Authorization	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Enabled (页 792)	-	RW	RW	指定是否可以在运行系统中操作所选对象。
FileName	-	-	-	-
Height	-	R	R	指定高度。
Layer	-	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	-	RW	RW	指定 X 坐标的值。
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (页 1279)	-	RW	RW	指定 Y 坐标的值。
Visible	-	-	-	-
Width	-	R	R	指定对象的宽度（以像素为单位）。

表格 1-65　方法

方法	RT Professional	RT Advanced	面板 RT	描述
Activate (页 1470)	-	√	√	启用永久区域或根画面。

PLCCodeViewer

说明



显示“PLC 代码查看器”对象 PLCCodeViewer 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIPLCCodeViewer

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-66　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Authorization	-	-	-	-

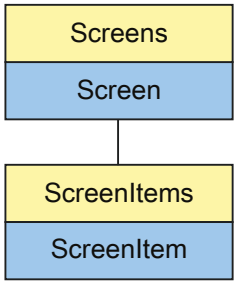
属性	RT Professional	RT Advanced	面板 RT	描述
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Enabled (页 792)	-	RW	RW	指定是否可以在运行系统中操作所选对象。
Height (页 863)	RW	R	R	指定高度。
Layer (页 913)	-	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	-	RW	RW	指定 X 坐标的值。
Location	-	-	-	-
Name	-	-	-	-
NavigateTo (页 988)	RW	RW	RW	指定要跳转至的模块。
NumberOfVisibleLines (页 995)	RW	-	-	在 PLC 代码查看器中指定可见线的数量。
PathHeaderFont (页 1023)	RW	-	-	设置标题字体。
Size	-	-	-	-
SymbolTableFont (页 1166)	RW	-	-	在 PLC 代码查看器中指定符号表的字体。
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolbarButtons	-	-	-	-
Top (页 1279)	-	RW	RW	指定 Y 坐标的值。
TransitionHeaderFont (页 1284)	RW	-	-	指定此 PLC 代码查看器的信息区域中的字体。
Visible (页 1418)	-	RW	RW	指定是否显示所选对象。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。

表格 1-67　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活	√	-	-	启用永久区域或根画面。
ActivateDynamic	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

Polygon

说明



表示“Polygon”对象。Polygon 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIPolygon

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-68　属性

属性	RT Professional	RT Advanced	面板 RT	描述
ActualPointIndex (页 603)	RW	-	-	指定当前角点的数量。
ActualPointLeft (页 604)	RW	-	-	指定当前角点相对于画面原点的 X 坐标。
ActualPointTop (页 605)	RW	-	-	指定当前角点相对于画面原点的 Y 坐标。
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定操作权限。
BackColor (页 637)	RW	RW	RW	指定背景色。
BackFillStyle (页 643)	RW	RW	RW	指定填充图案。
BorderBackColor (页 674)	RW	-	-	指定间断边框线的背景色。
BorderColor (页 678)	RW	RW	RW	指定线颜色。
BorderFlashingColorOff (页 683)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
BorderFlashingColorOn (页 685)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
BorderFlashingEnabled (页 687)	RW	-	-	指定是否在运行系统中闪烁对象的限值。

属性	RT Professional	RT Advanced	面板 RT	描述
BorderFlashingRate (页 689)	RW	-	-	指定边界线的闪烁频率。
BorderStyle (页 695)	RW	-	-	指定边框线类型。
BorderWidth (页 697)	RW	RW	RW	指定线宽。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (页 764)	RW	-	-	指定角形状。
DeviceStyle	-	-	-	-
EdgeStyle (页 788)	RW	RW	RW	指定线样式。
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。
FillingDirection (页 825)	RW	-	-	指定填充方向。
FillPatternColor	RW	-	-	-
Flashing	-	-	-	-
FlashingColorOff (页 832)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
FlashingColorOn (页 834)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
FlashingEnabled (页 836)	RW	-	-	指定是否为运行系统中的对象激活闪烁。
FlashingRate (页 839)	RW	-	-	指定边界线的闪烁频率。
Height (页 863)	RW	R	R	指定高度。
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
LineWidth	-	-	-	-
Location	-	-	-	-
Name	-	-	-	-
Points	-	-	-	-

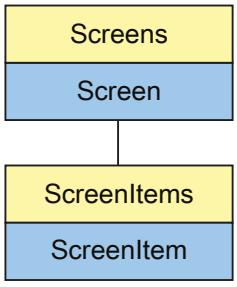
属性	RT Professional	RT Advanced	面板 RT	描述
PointsCount (页 1034)	RW	-	-	指定折线或多边形角点的个数。
RelativeFillLevel (页 1046)	RW	-	-	指定对象的填充百分比。
RotationAngle (页 1048)	RW	-	-	以度为单位指定旋转角度。
RotationCenterLeft (页 1049)	RW	-	-	指定运行系统中对象旋转中心点的 X 坐标。
RotationCenterTop (页 1050)	RW	-	-	指定运行系统中对象旋转中心点的 Y 坐标。
ShowFillLevel (页 1108)	RW	-	-	指定所选对象是否填充。
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (页 1278)	RW	RW	RW	指定工具提示文本。
Top (页 1279)	RW	RW	RW	指定 Y 坐标的值。
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
Visible (页 1418)	RW	RW	RW	指定是否显示所选对象。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。

表格 1-69　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	√	√	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

Polyline

说明



表示“Polyline”对象。Polyline 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIPolyline

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-70　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
ActualPointIndex (页 603)	RW	-	-	确定当前隅角点的数量。
ActualPointLeft (页 604)	RW	-	-	指定当前角点相对于画面原点的 X 坐标。
ActualPointTop (页 605)	RW	-	-	指定当前角点相对于画面原点的 Y 坐标。
Authorization (页 626)	RW	-	-	指定操作权限。
BackColor (页 637)	RW	RW	RW	指定背景色。
BorderEndStyle (页 683)	RW	-	-	指定线端的类型。
BorderStyle (页 695)	RW	-	-	指定边框线类型。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Color (页 723)	RW	RW	RW	指定线颜色。
CornerStyle (页 764)	RW	-	-	指定角形状。
DeviceStyle	-	-	-	-
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。
EndStyle (页 800)	RW	RW	RW	指定如何显示线端。
FillStyle (页 824)	-	RW	RW	指定是否以虚线显示背景颜色。

属性	RT Professional	RT Advanced	面板 RT	描述
Flashing	-	-	-	-
FlashingColorOff (页 832)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
FlashingColorOn (页 834)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
FlashingEnabled (页 836)	RW	-	-	指定是否为运行系统中的对象激活闪烁。
FlashingRate (页 839)	RW	-	-	指定边界线的闪烁频率。
Height (页 863)	RW	R	R	指定高度。
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定对象的 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
LineWidth (页 943)	RW	RW	RW	指定线宽。
Location	-	-	-	-
Name	-	-	-	-
Points	-	-	-	-
PointsCount (页 1034)	RW	-	-	指定折线或多边形角点的个数。
RotationAngle (页 1048)	RW	-	-	以度为单位指定旋转角度。
RotationCenterLeft (页 1049)	RW	-	-	指定运行系统中对象旋转中心点的 X 坐标。
RotationCenterTop (页 1050)	RW	-	-	指定运行系统中对象旋转中心点的 Y 坐标。
Size	-	-	-	-
StartStyle (页 1135)	RW	RW	RW	指定如何显示直线起始端。
Style (页 1163)	RW	RW	RW	指定线样式。
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-

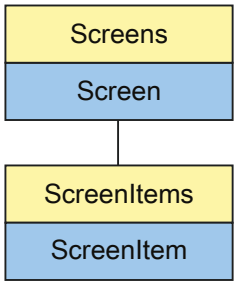
属性	RT Professional	RT Advanced	面板 RT	描述
ToolTipText (页 1278)	RW	RW	RW	指定工具提示文本。
Top (页 1279)	RW	RW	RW	指定对象的 Y 坐标的值。
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
Visible (页 1418)	RW	RW	RW	指定是否显示对象。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。

表格 1-71　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	√	√	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

ProDiag 概览

说明



表示“ProDiagOverview”对象。ProDiagOverview 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIProDiagOverview

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-72　属性

属性	RT Professional	RT Advanced	面板 RT	描述
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。
Height (页 863)	RW	R	R	指定高度。

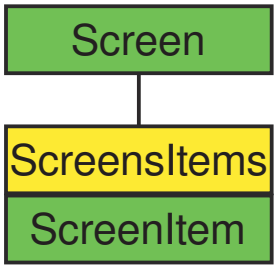
属性	RT Professional	RT Advanced	面板 RT	描述
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定 X 坐标的值。
Name	-	-	-	-
Top (页 1279)	RW	-	-	指定 Y 坐标的值。
Visible (页 1418)	RW	RW	RW	指定是否显示所选对象。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。

表格 1-73　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

ProtectedAreaNameView

描述



表示“EffectiveRangeName”(RFID) 对象。ProtectedAreaNameView 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIProtectedAreaNameView

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-74　属性

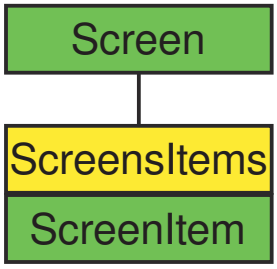
属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Authorization	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Font	-	-	-	-
Height	-	R	R	指定高度。
Layer (页 913)	-	-	RW	在包含对象的画面中指定图层。
Left (页 920)	-	-	RW	指定 X 坐标的值。
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (页 1279)	-	-	RW	指定 Y 坐标的值。
Visible (页 1418)	-	-	RW	指定是否显示所选对象。
Width	-	R	R	指定宽度。

表格 1-75　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	-	-	√	启用永久区域或根画面。

RangeLabelView

描述



表示“EffectiveRangeName”对象。RangeLabelView 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIRangeLabelView

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-76　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Authorization	-	-	-	-
Bounds	-	-	-	-

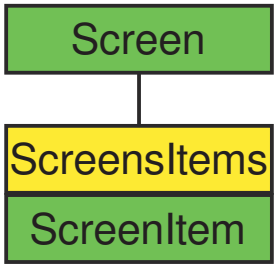
属性	RT Professional	RT Advanced	面板 RT	描述
CanBeGrouped	-	-	-	-
Font	-	-	-	-
Height	-	R	R	指定高度。
Layer (页 913)	-	-	RW	在包含对象的画面中指定图层。
Left (页 920)	-	-	RW	指定 X 坐标的值。
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (页 1279)	-	-	RW	指定 Y 坐标的值。
Visible (页 1418)	-	-	RW	指定是否显示所选对象。
Width	-	R	R	指定宽度。

表格 1-77　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	-	-	√	启用永久区域或根画面。

RangeQualityView

描述



表示“EffectiveRangeSignal”对象。RangeQualityView 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIRangeQualityView

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-78　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Height	-	R	R	指定高度。
Layer (页 913)	-	-	RW	在对象所在的画面中，返回图层。
Left (页 920)	-	-	RW	指定 X 坐标的值。
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (页 1279)	-	-	RW	指定 Y 坐标的值。
Visible (页 1418)	-	-	RW	指定是否显示所选对象。
Width	-	R	R	指定宽度。

表格 1-79　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	-	-	√	启用永久区域或根画面。

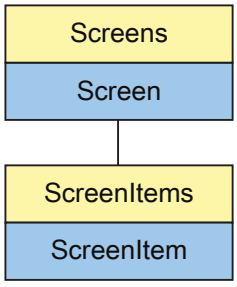
参见

Height (页 863)

Width (页 1432)

RecipeView

描述



显示“配方视图”对象 RecipeView 对象是 ScreenItems 列表的元素。

如果通过用户自定义函数更改此对象的设置，则即使再次调用画面后，更改的设置仍然会保留。

说明

无法通过用户自定义函数实现"Simple RecipeView"对象的动态化。

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-80　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AdvancedButtonPositions	-	-	-	-
AdvancedView	-	-	-	-
AllowEdit	-	-	-	-
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	-	RW	RW	指定所选对象在运行系统中的操作权限。
BackButtonVisible	-	-	-	-
BackColor (页 637)	-	RW	RW	指定背景色。
BorderBackColor	-	-	-	-
BorderColor	-	-	-	-
BorderWidth	-	-	-	-
Bounds	-	-	-	-
ButtonBackColor	-	-	-	-
ButtonBackFillStyle	-	-	-	-
ButtonBorderBackColor	-	-	-	-
ButtonBorderColor	-	-	-	-
ButtonBorderWidth	-	-	-	-
ButtonCornerRadius	-	-	-	-
ButtonEdgeStyle	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
ButtonFirstGradientColor	-	-	-	-
ButtonFirstGradientOffset	-	-	-	-
ButtonMiddleGradientColor	-	-	-	-
ButtonPositions	-	-	-	-
ButtonSecondGradientColor	-	-	-	-
ButtonSecondGradientOffset	-	-	-	-
CanBeGrouped	-	-	-	-
ColumnOrder	-	-	-	-
ColumnWidth	-	-	-	-
ComboboxFont	-	-	-	-
CornerRadius	-	-	-	-
DataRecordNameCaption	-	-	-	-
DataRecordNrCaption	-	-	-	-
DeviceStyle	-	-	-	-
Display3D	-	-	-	-
DisplayButton2Plc	-	-	-	-
DisplayButtonComparison	-	-	-	-
DisplayButtonDelete	-	-	-	-
DisplayButtonFromPlc	-	-	-	-
DisplayButtonHelp	-	-	-	-
DisplayButtonNew	-	-	-	-
DisplayButtonSave	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
DisplayButtonSaveAs	-	-	-	-
DisplayComboBox	-	-	-	-
DisplayGridLines	-	-	-	-
DisplayLabeling	-	-	-	-
DisplayNumbers	-	-	-	-
DisplaySize	-	-	-	-
DisplayStatusBar	-	-	-	-
DisplayTable	-	-	-	-
EdgeStyle	-	-	-	-
Enabled (页 792)	-	RW	RW	指定是否可以在运行系统中操作所选对象。
EntryNameCaption	-	-	-	-
EntryNameColumnWidth	-	-	-	-
EntryValueColFirst	-	-	-	-
EntryValueColumnWidth	-	-	-	-
EntryValueFieldLength	-	-	-	-
EntryValuePos	-	-	-	-
ES2RT_ButtonPositions	-	-	-	-
ES2RT_ColumnOrder	-	-	-	-
ES2RT_ColumnWidth	-	-	-	-
ES2RT_EntryNameColumnWidth	-	-	-	-
ES2RT_EntryValueColumnWidth	-	-	-	-
ES2RT_ListAreaHeight	-	-	-	-
ES2RT_ListAreaWidth	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
FitToSize	-	-	-	-
Flashing	-	-	-	-
FocusColor (页 843)	-	RW	RW	当对象获得焦点时，指定焦点边框的颜色。
FocusWidth (页 844)	-	RW	RW	当对象获得焦点时，指定边框宽度。
Font	-	-	-	-
ForeColor (页 854)	-	RW	RW	指定文本字体颜色。
HeaderFont	-	-	-	-
Height	-	R	R	指定高度。
IsRunningUnderCE	-	-	-	-
IsVerticalScrollBarEnabled	-	-	-	-
KeyboardOnline	-	-	-	-
Layer (页 913)	-	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	-	RW	RW	指定 X 坐标的值。
ListAreaHeight	-	-	-	-
ListAreaLeft	-	-	-	-
ListAreaTop	-	-	-	-
ListAreaWidth	-	-	-	-
Location	-	-	-	-
MaxNumberOfComboBoxCharacters	-	-	-	-
MenuButtonVisible	-	-	-	-
Name	-	-	-	-
NameColumnWidth	-	-	-	-
NumberOfButtons	-	-	-	-
NumberOfLines	-	-	-	-
PaddingBottom	-	-	-	-
PaddingLeft	-	-	-	-
PaddingRight	-	-	-	-
PaddingTop	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
PlcUDTFilter	-	-	-	-
ProhibitDataRecordTagInOnlySimpleView	-	-	-	-
Recipe	-	-	-	-
RecipeName (页 1043)	-	RW	RW	指定在“配方视图”(Recipe view) 中显示的配方名称。
RecipeNameCaption	-	-	-	-
RecipeNrCaption	-	-	-	-
RecipeNrColFirst	-	-	-	-
RecipeNumber (页 1044)	-	RW	RW	指定在“配方视图”(Recipe view) 中显示的配方编号。
RecordName (页 1045)	-	RW	RW	指定在“配方视图”(Recipe view) 中显示的配方数据记录的名称。
RecordNrColFirst	-	-	-	-
RecordNumber (页 1045)	-	R	R	返回当前在“配方视图”(Recipe view) 中显示的配方数据记录的编号。
RenameButtonVisible	-	-	-	-
SelectionBackColor (页 1084)	-	R	R	返回所选单元格的背景色。
SelectionForeColor (页 1086)	-	R	R	返回所选单元格的前景色。
SimpleView	-	-	-	-
Size	-	-	-	-
StatuslineFont	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableBackColor (页 1170)	-	R	R	返回表格单元格的背景色。
TableEvenRowBackColor	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
TableForeColor (页 1173)	-	R	R	返回对象表格单元格中采用的字体颜色。
TableGridLineColor (页 1175)	-	R	R	返回表格中网格线的颜色。
TableHeaderBackColor (页 1176)	-	R	R	返回表格标题的背景色。
TableHeaderBackFillStyle	-	-	-	-
TableHeaderBorderBackColor	-	-	-	-
TableHeaderBorderColor	-	-	-	-
TableHeaderBorderWidth	-	-	-	-
TableHeaderCornerRadius	-	-	-	-
TableHeaderEdgeStyle	-	-	-	-
TableHeaderFirstGradientColor	-	-	-	-
TableHeaderFirstGradientOffset	-	-	-	-
TableHeaderForeColor (页 1179)	-	R	R	返回表格标题的文本颜色。
TableHeaderMiddleGradientColor	-	-	-	-
TableHeaderPaddingBottom	-	-	-	-
TableHeaderPaddingLeft	-	-	-	-
TableHeaderPaddingRight	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
TableHeaderPaddingTop	-	-	-	-
TableHeaderSecondGradientColor	-	-	-	-
TableHeaderSecondGradientOffset	-	-	-	-
Tag4DataRecord	-	-	-	-
Tag4RecipeNumber	-	-	-	-
TextualObjectPositions	-	-	-	-
TextualObjectsAutoSize	-	-	-	-
TextualObjectsBorderBackColor	-	-	-	-
TextualObjectsBorderColor	-	-	-	-
TextualObjectsBorderWidth	-	-	-	-
TextualObjectsCornerRadius	-	-	-	-
TextualObjectsEdgeStyle	-	-	-	-
TextualObjectsPaddingBottom	-	-	-	-
TextualObjectsPaddingLeft	-	-	-	-
TextualObjectsPaddingRight	-	-	-	-
TextualObjectsPaddingTop	-	-	-	-
Top (页 1279)	-	R	R	返回 Y 坐标的值。

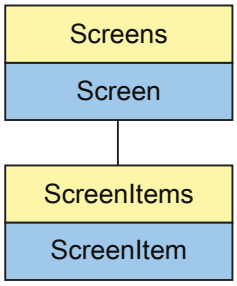
属性	RT Professional	RT Advanced	面板 RT	描述
UseButtonFirstGradient	-	-	-	-
UseButtonSecondGradient	-	-	-	-
UseDesignColorSchema	-	-	-	-
UseTableHeaderFirstGradient	-	-	-	-
UseTableHeaderSecondGradient	-	-	-	-
ValueCaption	-	-	-	-
ValueColumnWidth	-	-	-	-
VerticalScrolling	-	-	-	-
ViewType	-	-	-	-
ViewTypeForSaveStream	-	-	-	-
Visible (页 1418)	-	RW	RW	指定是否显示所选对象。
VisibleItems	-	-	-	-
Width	-	R	R	指定宽度。

表格 1-81　方法

方法	RT Professional	RT Advanced	面板 RT	描述
Activate (页 1470)	-	√	√	启用永久区域或根画面。

Rectangle

说明



表示“Rectangle”对象。Rectangle 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIRectangle

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-82　属性

属性	RT Professional	RT Advanced	Panel RT	说明
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定所选对象在运行系统中的操作权限。
BackColor (页 637)	RW	RW	RW	指定所选对象的背景色。
BackFillStyle (页 643)	RW	RW	RW	指定所选对象的填充图案。
BorderBackColor (页 674)	RW	-	-	指定所选对象间断边框线的背景色。
BorderColor (页 678)	RW	RW	RW	指定所选对象的线颜色。
BorderFlashingColorOff (页 683)	RW	-	-	指定闪烁状态“关闭”(Off) 的所选对象的边框线颜色。
BorderFlashingColorOn (页 685)	RW	-	-	指定闪烁状态“打开”(On) 的所选对象的边框线颜色。
BorderFlashingEnabled (页 687)	RW	-	-	指定所选对象的边框线是否在运行系统中闪烁。
BorderFlashingRate (页 689)	RW	-	-	指定所选对象的边框线的闪烁频率。
BorderStyle (页 695)	RW	-	-	指定所选对象边框线的类型。
BorderWidth (页 697)	RW	RW	RW	指定所选对象的线宽。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (页 764)	RW	-	-	指定特定对象的角形状。
DeviceStyle	-	-	-	-
DrawInsideFrame (页 787)	RW	-	-	指定线宽大于 1 的所选对象的边框线是在边框内描绘，还是与边框对称。
EdgeStyle (页 788)	RW	RW	RW	指定所选对象的线样式。
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。
FillingDirection (页 825)	RW	-	-	指定填充方向。

属性	RT Professional	RT Advanced	Panel RT	说明
FillPatternColor (页 821)	RW	-	-	指定所选对象的填充样式的颜色。
Flashing	-	-	-	-
FlashingColorOff (页 832)	RW	-	-	指定闪烁状态“关闭”(Off) 的所选对象的边框线颜色。
FlashingColorOn (页 834)	RW	-	-	指定闪烁状态“打开”(On) 的所选对象的边框线颜色。
FlashingEnabled (页 836)	RW	-	-	指定所选对象的边框线是否在运行系统中闪烁。
FlashingRate (页 839)	RW	-	-	指定所选对象的边框线的闪烁频率。
Height (页 863)	RW	RW	RW	指定所选对象的高度。
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定所选对象的 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
Location	-	-	-	-
Name	-	-	-	返回 STRING 形式的对象名称。
RelativeFillLevel (页 1046)	RW	-	-	指定对象的填充百分比。
RoundCornerHeight (页 1051)	RW	-	-	指定转角半径。
RoundCornerWidth (页 1051)	RW	-	-	指定转角半径。
ShowFillLevel (页 1108)	RW	-	-	指定所选对象是否填充。
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (页 1278)	RW	RW	RW	指定工具提示文本。
Top (页 1279)	RW	RW	RW	指定所选对象的 Y 坐标的值。

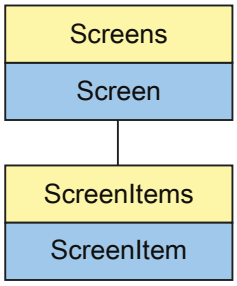
属性	RT Professional	RT Advanced	Panel RT	说明
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有在激活设计中定义的底纹。
Visible (页 1418)	RW	RW	RW	指定是否显示所选对象。
Width (页 1432)	RW	RW	RW	指定对象的宽度（以像素为单位）。

表格 1-83　方法

方法	RT Professional	RT Advanced	Panel RT	说明
激活 (页 1470)	支持	支持	支持	启用永久区域或根画面。
ActivateDynamic (页 1473)	支持	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	支持	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

RoundButton

说明



表示“Round button”对象。RoundButton 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIRoundButton

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-84　属性

属性	RT Professional	RT Advanced	Panel RT	说明
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定所选对象在运行系统中的操作权限。
BackColor (页 637)	RW	-	-	指定所选对象的背景色。
BackFillStyle (页 643)	RW	-	-	指定所选对象的填充图案。
BackFlashingColorOff (页 645)	RW	-	-	指定闪烁状态“关闭”(Off) 的背景色。
BackFlashingColorOn (页 646)	RW	-	-	指定闪烁状态“打开”(On) 的背景色。
BackFlashingEnabled (页 648)	RW	-	-	指定是否在运行系统中闪烁指定对象的背景。
BackFlashingRate (页 649)	RW	-	-	指定所选对象的背景闪烁的频率。
BorderBackColor (页 674)	RW	-	-	指定所选对象间断边框线的背景色。
BorderBrightColor3D (页 677)	RW	-	-	指定所选对象 3D 显示中以下边框部分的边框颜色：外边框的顶部和底部；内边框的底部和右部
BorderColor (页 678)	RW	-	-	指定线颜色。
BorderFlashingColorOff (页 683)	RW	-	-	指定闪烁状态“关闭”(Off) 的所选对象的边框线颜色。
BorderFlashingColorOn (页 685)	RW	-	-	指定闪烁状态“打开”(On) 的所选对象的边框线颜色。
BorderFlashingEnabled (页 687)	RW	-	-	指定所选对象的边框线是否在运行系统中闪烁。
BorderFlashingRate (页 689)	RW	-	-	指定所选对象的边框线的闪烁频率。
BorderShadeColor3D	-	-	-	-
BorderStyle (页 695)	RW	-	-	指定所选对象边框线的类型。

属性	RT Professional	RT Advanced	Panel RT	说明
BorderWidth (页 697)	RW	-	-	指定所选对象的线宽。
BorderWidth3D (页 700)	RW	-	-	指定所选对象 3D 显示的边框宽度。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (页 764)	RW	-	-	指定特定对象的角形状。
DrawInsideFrame (页 787)	RW	-	-	指定线宽大于 1 的所选对象的边框线是在边框内描绘，还是与边框对称。
EdgeStyle (页 788)	RW	-	-	指定所选对象的线样式。
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。
FillingDirection (页 825)	RW	-	-	指定填充方向。
FillPatternColor (页 821)	RW	-	-	指定所选对象的填充样式的颜色。
Flashing	-	-	-	-
FlashingColorOff (页 832)	RW	-	-	指定闪烁状态“关闭”(Off) 的所选对象的边框线颜色。
FlashingColorOn (页 834)	RW	-	-	指定闪烁状态“打开”(On) 的所选对象的边框线颜色。
FlashingEnabled (页 836)	RW	-	-	指定所选对象的边框线是否在运行系统中闪烁。
FlashingRate (页 839)	RW	-	-	指定所选对象的边框线的闪烁频率。
Font	-	-	-	-
FontBold (页 849)	RW	-	-	指定所选对象的文本是否以粗体显示。
FontItalic (页 849)	RW	-	-	指定所选对象的文本是否以斜体显示。
FontName (页 851)	RW	-	-	指定所选对象的字体。
FontSize (页 852)	RW	-	-	指定所选对象的文本字体大小。
FontUnderline (页 853)	RW	-	-	指定所选对象的文本是否加下划线。
ForeColor (页 854)	R	-	-	指定所选对象的文本字体颜色。

属性	RT Professional	RT Advanced	Panel RT	说明
Height (页 863)	R	-	-	指定所选对象的高度。
HorizontalAlignment (页 881)	R	-	-	指定所选对象内的文本水平对齐。
Layer (页 913)	R	-	-	返回画面中对象所在的层。
Left (页 920)	R	-	-	返回对象的 X 坐标的值。
LineEndShapeStyle (页 940)	R	-	-	指定线端的形状。
Location	-	-	-	-
Mode (页 981)	R	-	-	返回指定对象的字段类型。
Name	-	-	-	-
PictureAlignment (页 1027)	R	-	-	返回过程映像中背景画面的显示类型。
PictureDeactivated (页 1028)	R	-	-	返回在“已禁用”(Disabled) 状态下显示的图形。
PictureOff (页 1029)	R	-	-	返回在“关闭”(Off) 状态下显示的图形。
PictureOn (页 1030)	R	-	-	返回在“开启”(On) 状态下显示的画面。
Pressed (页 1036)	R	-	-	返回关于指定的对象是否显示为已按下状态的信息。
Radius (页 1040)	R	-	-	返回特定“Circle”对象的半径。
RelativeFillLevel (页 1046)	R	-	-	返回对象的填充百分比。
ShowFillLevel (页 1108)	R	-	-	返回关于特定的对象是否已填充的信息。
Size	-	-	-	-
StyleSettings (页 1164)	R	-	-	返回对象的显示样式。
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Text (页 1185)	R	-	-	返回文本域的标签。
TextOrientation (页 1189)	R	-	-	返回特定对象的文本方向。

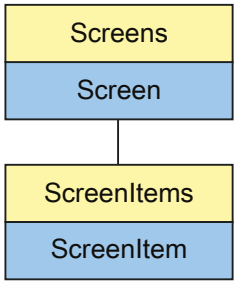
属性	RT Professional	RT Advanced	Panel RT	说明
Toggle (页 1244)	R	-	-	返回关于在运行系统中对特定对象操作完后是否仍占用该对象的信息。
ToolTipText (页 1278)	R	-	-	返回工具提示文本。
Top (页 1279)	R	-	-	返回对象的 Y 坐标的值。
Transparency (页 1284)	R	-	-	返回对象透明度（百分数形式）。
TransparentColorDeactivatedPicture (页 1287)	R	-	-	返回在“已禁用”(Disabled) 状态下要设置为“透明”(transparent) 的所分配位图对象颜色。
TransparentColorPictureOff (页 1287)	R	-	-	返回在“关闭”(Off) 状态下要设置为“透明”(transparent) 的所分配位图对象颜色。
TransparentColorPictureOn (页 1288)	R	-	-	返回在“开启”(On) 状态下要设置为“透明”(transparent) 的所分配位图对象颜色。
UseDesignColorSchema (页 1353)	R	-	-	返回关于在当前设计的全局颜色方案中定义的颜色是否用于此对象的信息。
UseDesignShadowSettings (页 1355)	R	-	-	返回关于显示的对象是否带有在激活设计中定义的底纹的信息。
UseTransparentColorDeactivatedPicture (页 1367)	R	-	-	返回关于通过“TransparentColorDeactivatedPicture”属性指定的透明色是否用于“已禁用”(Disabled) 状态的信息。
UseTransparentColorPictureOff (页 1367)	R	-	-	返回关于通过属性“TransparentColorPictureOff”定义的透明色是否用于“关闭”(Off) 状态的信息。
UseTransparentColorPictureOn (页 1368)	R	-	-	返回关于通过属性“TransparentColorPictureOn”定义的透明色是否用于“开启”(On) 状态的信息。
VerticalAlignment (页 1413)	R	-	-	返回特定对象的文本垂直对齐方式。
Visible (页 1418)	R	-	-	返回关于是否显示特定对象的信息。
Width (页 1432)	R	-	-	返回对象的宽度（以像素为单位）。

表格 1-85　方法

方法	RT Professional	RT Advanced	Panel RT	说明
激活 (页 1470)	支持	支持	支持	启用永久区域或根画面。
ActivateDynamic (页 1473)	支持	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	支持	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

S7GraphOverview

说明



表示“GRAPH overview”对象。S7GraphOverview 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIS7GraphOverview

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-86　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
AssociatedS7GraphDBName	-	-	-	-
AssociatedS7GraphDBTag	-	-	-	-
Authorization (页 626)	RW	-	-	指定操作权限。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Enabled (页 792)	-	RW	RW	指定是否可以在运行系统中操作所选对象。
ErrorColor (页 803)	RW	-	-	指定 GRAPH 概览中错误的颜色。
Errorflag (页 805)	RW	-	-	指定是否在设备 / 详细信息视图中显示错误描述。
Height (页 863)	-	R	R	指定高度。
HighlightColor (页 869)	RW	-	-	指定图形概览中的高亮颜色。
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	-	RW	RW	指定 X 坐标的值。
Location	-	-	-	-
Name	-	-	-	-

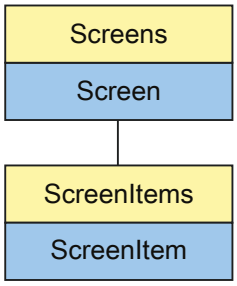
属性	RT Professional	RT Advanced	面板 RT	描述
PathHeaderBackColor (页 1023)	-	-	-	
PathHeaderFont (页 1023)	RW	-	-	设置标题字体。
PathHeaderTextColor (页 1023)	RW	-	-	指定 GRAPH 概览标题中的字体颜色。
SeparatorColor (页 1092)	RW	-	-	指定选择列表中分隔线的颜色。
Size	-	-	-	-
StepBackColor (页 1160)	RW	-	-	指定步的背景色。
StepFont (页 1161)	RW	-	-	指定步的前景色。
StepTextColor (页 1162)	RW	-	-	指定步的文本颜色。
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (页 1279)	-	RW	RW	指定 Y 坐标的值。
Visible (页 1418)	-	RW	RW	指定是否显示所选对象。
Width (页 1432)	-	R	R	指定宽度。

表格 1-87　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活	√	-	-	启用永久区域或根画面。
ActivateDynamic	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

ScreenWindow

说明



表示“ScreenWindow”对象。ScreenWindow 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIScreenWindow

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

属性	RT Professional	RT Advanced	面板 RT	描述
AdaptScreenToWindow (页 607)	R	-	-	指定画面窗口中显示的画面是否根据运行系统中画面窗口的大小进行调整。
AdaptWindowtoScreen (页 607)	R	-	-	指定画面窗口是否根据运行系统中显示的画面进行调整。
AllTagTypesAllowed	-	-	-	-
BorderEnabled (页 682)	R	-	-	指定窗口在运行系统中是否带边框显示。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CaptionText (页 712)	RW	-	-	指定标题中将要显示的文本。
Flashing	-	-	-	-
Height (页 863)	RW	-	-	指定高度。
HorizontalScrollBarPosition (页 884)	RW	-	-	指定水平滚动条中的滑块位置。
IndipendentWindow	-	-	-	-
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
LeftOffset (页 926)	RW	-	-	设置画面大于画面窗口时其画面零点的水平位移。
Location	-	-	-	-
MenuToolBarConfig (页 955)	RW	-	-	确定包含用户自定义菜单和工具栏的组态文件。
MonitorNumber (页 982)	R	-	-	指定其内容显示在画面窗口中的监视器数量。
Name	-	-	-	-
ProcessTag	-	-	-	-
ScreenName (页 1069)	R	-	-	指定在运行系统画面窗口中显示的画面。
ScreenScaleMode (页 1071)	R	-	-	指定所显示画面的缩放模式。

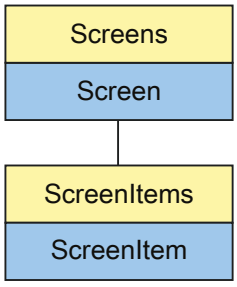
属性	RT Professional	RT Advanced	面板 RT	描述
ServerPrefix	RW	-	-	指定在运行系统画面窗口中显示的画面所在的服务器，或返回服务器名称。
ShowCaption (页 1100)	R	-	-	指定是否显示工具栏。
ShowScrollbars (页 1116)	R	-	-	指定是否显示滚动条。
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TagPrefix (页 1181)	RW	-	-	指定变量前缀，该前缀加在了画面窗口中包含的所有变量前。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。
TopOffset (页 1282)	RW	-	-	设置画面大于画面窗口时其画面零点的垂直位移。
VerticalScrollbarPosition (页 1415)	RW	-	-	指定对象垂直滚动条中的滑块位置。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。
WindowCloseEnabled (页 1436)	R	-	-	指定是否可在运行系统中关闭窗口。
WindowMaximizeEnabled (页 1436)	R	-	-	指定运行系统中是否可以最大化对象。
WindowMovingEnabled (页 1437)	R	-	-	指定运行系统中是否可以移动对象。
WindowOnTop (页 1438)	R	-	-	指定对象是否始终位于运行系统的前景中。
WindowSizingEnabled (页 1439)	R	-	-	指定是否可更改大小。
WindowStartupPosition (页 1441)	R	-	-	指定组态独立的画面窗口时画面窗口的位置和模式。
ZoomFactor (页 1469)	RW	-	-	指定画面或画面窗口的缩放因子。

表格 1-88　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

ScriptDiagnostics

说明



表示“ApplicationWindow”对象。ApplicationWindow 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIApplicationWindow

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-89　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
BorderEnabled (页 682)	RW	-	-	指定窗口在运行系统中是否带边框显示。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
GSCRuntimeAllowed	-	-	-	-
Height (页 863)	RW	-	-	指定高度。
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
Location	-	-	-	-
Name	-	-	-	-
ShowCaption (页 1100)	RW	-	-	指定是否显示工具栏。
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Template (页 1184)	RW	-	-	指定用于显示“打印作业/脚本诊断”(Print job/ Script diagnostics) 对象窗口内容的模板。
Top (页 1279)	-	-	-	-
Visible (页 1418)	RW	-	-	指定是否显示所选对象。

属性	RT Professional	RT Advanced	面板 RT	描述
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。
WindowCloseEnabled (页 1436)	RW	-	-	指定是否可在运行系统中关闭窗口。
WindowMaximizeEnabled (页 1436)	RW	-	-	指定运行系统中是否可以最大化对象。
WindowMovingEnabled (页 1437)	RW	-	-	指定运行系统中是否可以移动对象。
WindowOnTop (页 1438)	RW	-	-	指定对象是否始终位于运行系统的前景中。
WindowsContents (页 1439)	RW	-	-	指定打印作业或脚本诊断的内容。
WindowSizingEnabled (页 1439)	RW	-	-	指定是否可更改大小。

表格 1-90　方法

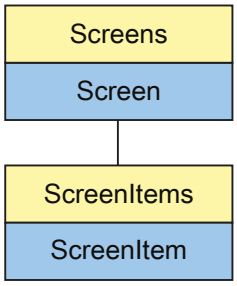
方法	RT Professional	RT Advanced	面板 RT	描述
Activate (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

参见

Name (页 985)

Slider

说明



表示“Slider”对象。Slider 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMISlider

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-91　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AlarmLowerLimitColor	-	-	-	-
AlarmUpperLimitColor	-	-	-	-
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	RW	RW	指定操作权限。
BackColor (页 637)	RW	RW	RW	指定背景色。
BackFillStyle (页 643)	RW	-	-	指定所选对象的填充图案。
BackPicture (页 651)	RW	-	-	指定背景图形。
BarBackColor (页 653)	RW	RW	RW	确定所选对象的棒图的背景色。
BarColor (页 656)	RW	RW	RW	指定工具栏的颜色。
BarOrientation	-	-	-	-
BorderBackColor	-	-	-	-
BorderBrightColor3D (页 677)	RW	RW	RW	指定 3D 显示中以下边框部分的边框颜色。
BorderColor	-	-	-	-
BorderInnerStyle3D (页 691)	RW	-	-	指定对象边框的内侧部分的外观。
BorderInnerWidth3D (页 692)	RW	RW	RW	以 3D 形式表现所选择的对象时，指定内轮廓的宽度。
BorderOuterStyle3D	-	-	-	-
BorderOuterWidth3D (页 693)	RW	RW	RW	以 3D 形式表现所选择的对象时，指定外轮廓的宽度。
BorderShadeColor3D (页 694)	RW	RW	RW	指定 3D 显示中以下边框部分的边框颜色。
BorderWidth (页 697)	RW	RW	RW	指定线宽。

属性	RT Professional	RT Advanced	面板 RT	描述
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Caption (页 708)	RW	RW	RW	指定标题中将要显示的文本。
CompatibilityMode	-	-	-	-
ContinousChange (页 762)	RW	-	-	指定是在释放鼠标按钮后还是在运行系统中更改滚动条位置后立即传送“ProcessValue”的值。
CornerRadius	-	-	-	-
DeviceStyle	-	-	-	-
EdgeStyle	-	-	-	-
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。
Flashing (页 830)	RW	-	-	指定激活闪烁功能。
FocusColor (页 843)	RW	RW	RW	当对象获得焦点时，指定焦点边框的颜色。
FocusWidth (页 844)	RW	RW	RW	当对象获得焦点时，指定边框宽度。
Font (页 846)	RW	-	-	指定字体。
ForeColor (页 854)	RW	RW	RW	指定文本字体颜色。
ForeColorTransparency	-	-	-	-
Height (页 863)	RW	R	R	指定高度。
LabelColor (页 907)	RW	RW	RW	指定刻度标签的颜色。
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定 X 坐标的值。
LimitRangeCollection	-	-	-	-
Location	-	-	-	-
MaximumValue (页 953)	RW	RW	RW	指定所选对象的刻度范围的最大值。
MinimumValue (页 979)	RW	RW	RW	指定所选对象的刻度范围的最小值。
Name (页 985)	RW	-	-	指定对象名称。
PositionFont (页 1035)	RW	-	-	设置滑块标签的字体。

属性	RT Professional	RT Advanced	面板 RT	描述
ProcessValue (页 1037)	RW	RW	RW	指定要显示的数值的默认值。
ScaleLabelFieldLength	-	-	-	-
ScalePosition	-	-	-	-
ShowBar (页 1100)	RW	RW	RW	指定显示的过程值是否也可通过填充的棒图显示。
ShowLimitLines	-	-	-	-
ShowLimitMarkers	-	-	-	-
ShowLimitRanges	-	-	-	-
ShowPosition (页 1113)	RW	RW	RW	添加滚动条位置的值的数字显示。
ShowScale	-	-	-	-
ShowThumb (页 1121)	R	RW	RW	显示"Slider"对象的滚动条。
ShowTickLabels (页 1122)	R	RW	RW	指定是否以刻度显示标签。
ShowTicks (页 1123)	R	RW	RW	指定是否显示刻度中的标记。
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ThumbBackColor (页 1192)	R	RW	RW	指定滑块的背景色。
ThumbPicture (页 1193)	R	-	-	返回滑块中滑块元素的图形。
TickStyle (页 1194)	R	-	-	返回标记在刻度中的显示方式。
Top (页 1279)	R	RW	RW	指定 Y 坐标的值。
Transparency (页 1284)	R	-	-	返回对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	R	-	-	返回关于在当前设计的全局颜色方案中定义的颜色是否用于此对象的信息。

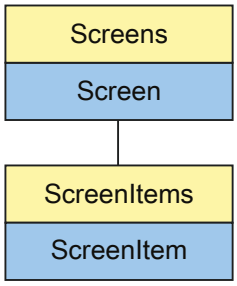
属性	RT Professional	RT Advanced	面板 RT	描述
UseDesignShadowSettings (页 1355)	R	-	-	返回关于显示的对象是否带有全局阴影的信息。
UseTwoHandOperation	-	-	-	-
Visible (页 1418)	R	RW	RW	指定是否显示所选对象。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。

表格 1-92　方法

方法	RT Professional	RT Advanced	面板 RT	描述
Activate (页 1470)	√	√	√	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

SmartClientView

说明



表示“Sm@rtClient View”对象。SmartClientView 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMISmartClientView

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-93　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AdressPreview	-	-	-	-
AllowMenu	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
ConnectionType	-	-	-	-
ConnectOnStart	-	-	-	-
Enabled (页 792)	-	RW	RW	指定是否可以在运行系统中操作所选对象。
EncryptCommunication	-	-	-	-
Flashing	-	-	-	-
Font	-	-	-	-
Height	-	R	R	指定高度。
Layer (页 913)	-	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	-	RW	RW	指定 X 坐标的值。
LocalCursor	-	-	-	-
Location	-	-	-	-
Machine	-	-	-	-
MachineName (页 951)	-	RW	RW	设置要监视的设备的网络 ID。
Name	-	-	-	-
Password (页 1021)	-	RW	RW	设置远程监视的密码。
PasswordsMustBeEncrypted	-	-	-	-
ScaleDenominator	-	-	-	-
ScaleNumerator	-	-	-	-
Scaling	-	-	-	-
ServerScale	-	-	-	-
Shared	-	-	-	-
ShowControls	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-

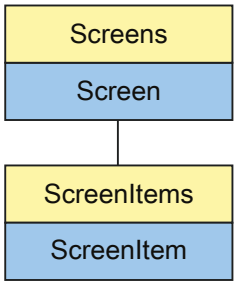
属性	RT Professional	RT Advanced	面板 RT	描述
TabIndexAlpha	-	-	-	-
TcpPortNr	-	-	-	-
Top (页 1279)	-	RW	RW	指定 Y 坐标的值。
UseCurserKeyScroll	-	-	-	-
ViewOnly (页 1416)	-	RW	RW	指定 Sm@rtClient 显示用于远程监视还是远程维护。
Visible (页 1418)	-	RW	RW	指定是否显示所选对象。
Width	-	R	R	指定宽度。

表格 1-94　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	-	√	√	启用永久区域或根画面。

StatusForce

描述



表示“Watch table”对象。StatusForce 对象是 ScreenItems 列表的元素。

如果通过用户自定义函数更改此对象的设置，则即使再次调用画面后，更改的设置仍然会保留。

VBS 的类型标识符

HMIStatusForce

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-95　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Appearance	-	-	-	-
Authorization (页 626)	-	RW	RW	指定操作权限。
BackColor (页 637)	-	RW	RW	指定背景色。
BorderBackColor	-	-	-	-
BorderColor	-	-	-	-
BorderWidth	-	-	-	-
Bounds	-	-	-	-
ButtonBackColor	-	-	-	-
ButtonBackFillStyle	-	-	-	-
ButtonBorderBackColor	-	-	-	-
ButtonBorderColor	-	-	-	-
ButtonBorderWidth	-	-	-	-
ButtonCornerRadius	-	-	-	-
ButtonEdgeStyle	-	-	-	-
ButtonFirstGradientColor	-	-	-	-
ButtonFirstGradientOffset	-	-	-	-
ButtonMiddleGradientColor	-	-	-	-
ButtonPositions	-	-	-	-
ButtonSecondGradientColor	-	-	-	-
ButtonSecondGradientOffset	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
CanBeGrouped	-	-	-	-
ColumnOrder	-	-	-	-
ColumnsMoveable	-	-	-	-
ColumnTextBit	-	-	-	-
ColumnTextConnection	-	-	-	-
ColumnTextDataType	-	-	-	-
ColumnTextDbNumber	-	-	-	-
ColumnTextFormat	-	-	-	-
ColumnTextOffset	-	-	-	-
ColumnTextRead	-	-	-	-
ColumnTextType	-	-	-	-
ColumnTextWrite	-	-	-	-
ColumnWidth	-	-	-	-
CornerRadius	-	-	-	-
CountVisibleItems	-	-	-	-
DeviceStyle	-	-	-	-
EdgeStyle	-	-	-	-
Enabled (页 792)	-	RW	RW	指定是否可以在运行系统中操作所选对象。
ES2RT_ColumnOrder	-	-	-	-
ES2RT_ColumnWidth	-	-	-	-
FitToSize	-	-	-	-
Flashing	-	-	-	-
FocusColor (页 843)	-	RW	RW	当对象获得焦点时，指定焦点边框的颜色。
FocusWidth (页 844)	-	RW	RW	当对象获得焦点时，指定边框宽度。
Height	-	R	R	指定高度。
IsRunningUnderCE	-	-	-	-
Layer (页 913)	-	RW	RW	在包含对象的画面中指定图层。

属性	RT Professional	RT Advanced	面板 RT	描述
Left (页 920)	-	RW	RW	指定 X 坐标的值。
ListAreaHeight	-	-	-	-
ListAreaLeft	-	-	-	-
ListAreaTop	-	-	-	-
ListAreaWidth	-	-	-	-
Location	-	-	-	-
Name	-	-	-	-
Object	-	-	-	-
PaddingBottom	-	-	-	-
PaddingLeft	-	-	-	-
PaddingRight	-	-	-	-
PaddingTop	-	-	-	-
PLCFilter	-	-	-	-
SelectionBackColor (页 1084)	-	RW	RW	指定所选单元格的背景色。
SelectionForeColor (页 1086)	-	RW	RW	指定所选单元格的前景色。
SetOfVisibleColumns	-	-	-	-
ShowReadButton	-	-	-	-
ShowTableGridlines (页 1121)	-	RW	RW	指定是否在表格中显示网格线。
ShowWriteButton	-	-	-	-
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableBackColor (页 1170)	-	RW	RW	指定表格单元格的背景色。
TableEvenRowBackColor	-	-	-	-
TableFont	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
TableForeColor (页 1173)	-	RW	RW	指定对象表格单元格中采用的文本颜色。
TableGridlineColor	-	-	-	-
TableHeaderBackColor (页 1176)	-	RW	RW	指定表格标题的背景色。
TableHeaderBackFillStyle	-	-	-	-
TableHeaderBorderBackColor	-	-	-	-
TableHeaderBorderColor	-	-	-	-
TableHeaderBorderWidth	-	-	-	-
TableHeaderCornerRadius	-	-	-	-
TableHeaderEdgeStyle	-	-	-	-
TableHeaderFirstGradientColor	-	-	-	-
TableHeaderFirstGradientOffset	-	-	-	-
TableHeaderFont	-	-	-	-
TableHeaderForeColor (页 1179)	-	R	R	返回特定对象的表格标题的文本颜色。
TableHeaderMiddleGradientColor	-	-	-	-
TableHeaderPaddingBottom	-	-	-	-
TableHeaderPaddingLeft	-	-	-	-
TableHeaderPaddingRight	-	-	-	-

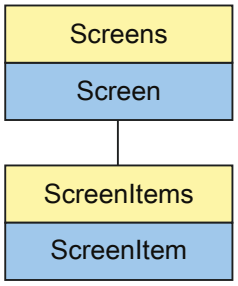
属性	RT Professional	RT Advanced	面板 RT	描述
TableHeaderPaddingTop	-	-	-	-
TableHeaderSecondGradientColor	-	-	-	-
TableHeaderSecondGradientOffset	-	-	-	-
ToolTipText (页 1278)	-	R	R	指定工具提示文本。
Top (页 1279)	-	R	R	返回 Y 坐标的值。
UseButtonFirstGradient	-	-	-	-
UseButtonSecondGradient	-	-	-	-
UseDesignColorSchema	-	-	-	-
UseTableHeaderFirstGradient	-	-	-	-
UseTableHeaderSecondGradient	-	-	-	-
Visible (页 1418)	-	R	R	返回关于是否显示特定对象的信息。
Width	-	R	R	指定宽度。

表格 1-96　方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	-	√	√	启用永久区域或根画面。

Switch

说明



表示“Switch”对象。Switch 对象是 ScreenItems 列表的元素。

下列对象属性的可用性取决于所选的模式：

属性	“带有文本的开关”	“带有图形的开关”	“开关”
CaptionColor (页 710)	--	--	x
CaptionText (页 712)	--	--	x
HorizontalAlignment (页 881)	X	--	--
InnerBackColorOff (页 890)	--	--	x
InnerBackColorOn (页 891)	--	--	x
TextOn (页 1188)	x	--	x
TextOff (页 1187)	x	--	x
VerticalAlignment (页 1413)	x	--	--

VBS 的类型标识符

HMISwitch

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-97　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AboveUpperLimitColor	-	-	-	-
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	-	RW	RW	指定操作权限。
BackColor (页 637)	-	RW	RW	指定背景色。
BackFillStyle	-	-	-	-
BackFlashingColorOff	-	-	-	-
BackFlashingColorOn	-	-	-	-
BackFlashingEnabled	-	-	-	-
BackFlashingRate	-	-	-	-
BelowLowerLimitColor	-	-	-	-
BorderBackColor	-	-	-	-
BorderBrightColor3D	-	-	-	-
BorderColor	-	-	-	-
BorderFlashingColorOff	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
BorderFlashingColorOn	-	-	-	-
BorderFlashingEnabled	-	-	-	-
BorderFlashingRate	-	-	-	-
BorderShadeColor3D	-	-	-	-
BorderWidth	-	-	-	-
BorderWidth3D	-	-	-	-
CanBeGrouped	-	-	-	-
CaptionColor (页 710)	-	RW	RW	指定将要在标题栏中显示文本的颜色。
CaptionFont	-	-	-	-
CaptionText (页 712)	-	RW	RW	指定标题中将要显示的文本。
CornerRadius	-	-	-	-
CornerStyle	-	-	-	-
DeviceStyle	-	-	-	-
DrawInsideFrame	-	-	-	-
EdgeStyle	-	-	-	-
Enabled (页 792)	-	RW	RW	指定是否可以在运行系统中操作所选对象。
FirstGradientColor	-	-	-	-
FirstGradientOffset	-	-	-	-
FitToLargest	-	-	-	-
Flashing	-	-	-	-
FlashingColorOff	-	-	-	-
FlashingColorOn	-	-	-	-
FlashingEnabled	-	-	-	-
FlashingOnLimitViolation	-	-	-	-
FlashingRate	-	-	-	-
FocusColor (页 843)	-	RW	RW	当对象获得焦点时，指定焦点边框的颜色。

属性	RT Professional	RT Advanced	面板 RT	描述
FocusWidth (页 844)	-	RW	RW	当对象获得焦点时，指定边框宽度。
Font	-	-	-	-
ForeColor (页 854)	-	RW	RW	指定文本字体颜色。
Height	-	R	R	指定高度。
HelpText (页 868)	-	RW	RW	指定作为特定对象的用户帮助显示在运行系统中的工具提示。
HorizontalAlignment (页 881)	-	RW	RW	指定文本水平对齐。
HorizontalPictureAlignment	-	-	-	-
HotKey	-	-	-	-
InnerBackColorOff (页 890)	-	RW	RW	在“Switch”对象处于“关闭”(OFF) 状态时设置其滚动条下方的颜色。
InnerBackColorOn (页 891)	-	RW	RW	在“Switch”对象处于“打开”(ON) 状态时设置其滚动条下方的颜色。
Layer (页 913)	-	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	-	RW	RW	指定 X 坐标的值。
LineEndShapeStyle	-	-	-	-
Location	-	-	-	-
MiddleGradientColor	-	-	-	-
Mode	-	-	-	-
Name	-	-	-	-
OnValue	-	-	-	-
PictureAreaBottomMargin	-	-	-	-
PictureAreaLeftMargin	-	-	-	-
PictureAreaRightMargin	-	-	-	-
PictureAreaTopMargin	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
PictureAutoSizing	-	-	-	-
PictureOff	-	-	-	-
PictureOn	-	-	-	-
ProcessValue	-	-	-	-
SecondGradientColor	-	-	-	-
SecondGradientOffset	-	-	-	-
ShowCaption	-	-	-	-
Size	-	-	-	-
StyleItem	-	-	-	-
SwitchOrientation	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TextAreaBottomMargin	-	-	-	-
TextAreaLeftMargin	-	-	-	-
TextAreaRightMargin	-	-	-	-
TextAreaTopMargin	-	-	-	-
TextOff (页 1187)	-	R	R	返回在“关闭”(Off) 状态下显示的文本。
TextOn (页 1188)	-	R	R	返回在“开启”(On) 状态下显示的文本。
TextOrientation	-	-	-	-
ToolTipText (页 1278)	-	R	R	返回工具提示文本。
Top (页 1279)	-	R	R	返回 Y 坐标的值。
UseDesignColorSchema	-	-	-	-
UseFirstGradient	-	-	-	-
UseSecondGradient	-	-	-	-
VerticalAlignment (页 1413)	-	R	R	返回特定对象中的文本垂直对齐方式。

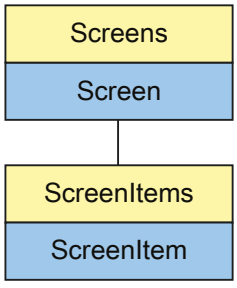
属性	RT Professional	RT Advanced	面板 RT	描述
VerticalPictureAlignment	-	-	-	-
Visible (页 1418)	-	R	R	返回关于是否显示特定对象的信息。
Width	-	R	R	指定宽度。

表格 1-98　方法

方法	RT Professional	RT Advanced	面板 RT	描述
Activate (页 1470)	-	√	√	启用永久区域或根画面。

SymbolicIOField

说明



表示“SymbolicIOField”对象。SymbolicIOField 对象是 ScreenItems 列表的元素。

下列对象属性的可用性取决于所选的模式：

属性	“输入”(Input)	“输出”	“输入/输出”(Input/Output)	“两种状态”(Two states)
BackColor (页 637)	x	x	x	x
BorderColor (页 678)	--	x	--	x
BorderWidth (页 697)	--	--	--	x
Enabled (页 792)	x	--	x	--
HelpText (页 868)	x	--	x	--
VerticalAlignment (页 1413)	--	x	--	x
HorizontalAlignment (页 881)	--	x	--	x

VBS 的类型标识符

HMISymbolicIOField

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-99　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AboveUpperLimitColor	-	-	-	-
AcceptOnExit (页 597)	RW	-	-	指定是否在保留输入字段内容时，自动对输入字段进行确认。

属性	RT Professional	RT Advanced	面板 RT	描述
AdaptBorder (页 606)	RW	-	-	指定对象的边框是否根据文本的大小进行动态调整。
AllTagTypesAllowed	-	-	-	-
AskOperationMotive (页 624)	RW	-	-	指定是否记录操作此对象的原因。
Assignments (页 625)	RW	-	-	指定包含输出值与实际要输出的输出文本之间分配关系的列表。
Authorization (页 626)	RW	RW	RW	指定所选对象在运行系统中的操作权限。
BackColor (页 637)	RW	RW	RW	指定所选对象的背景色。
BackFillStyle (页 643)	RW	-	-	指定所选对象的填充图案。
BackFlashingColorOff (页 645)	RW	-	-	指定闪烁状态“关闭”(Off) 的背景色。
BackFlashingColorOn (页 646)	RW	-	-	指定闪烁状态“打开”(On) 的背景色。
BackFlashingEnabled (页 648)	RW	-	-	指定是否在运行系统中闪烁指定对象的背景。
BackFlashingRate (页 649)	RW	-	-	指定所选对象的背景闪烁的频率。
BelowLowerLimitColor	-	-	-	-
BitNumber (页 660)	RW	-	-	指定必须更改状态才能触发值更改的位。
BorderBackColor	-	-	-	-
BorderColor (页 678)	RW	RW	RW	指定所选对象的线颜色。
BorderFlashingColorOff (页 683)	RW	-	-	指定闪烁状态“关闭”(Off) 的所选对象的边框线颜色。
BorderFlashingColorOn (页 685)	RW	-	-	指定闪烁状态“打开”(On) 的所选对象的边框线颜色。
BorderFlashingEnabled (页 687)	RW	-	-	指定所选对象的边框线是否在运行系统中闪烁。

属性	RT Professional	RT Advanced	面板 RT	描述
BorderFlashingRate (页 689)	RW	-	-	指定所选对象的边界线的闪烁频率。
BorderStyle (页 695)	RW	-	-	指定所选对象边框线的类型。
BorderWidth (页 697)	RW	RW	RW	指定所选对象的线宽。
BottomMargin	-	-	-	-
CanBeGrouped	-	-	-	-
CaptionBackColor (页 709)	RW	-	-	指定所选对象的说明的背景色。
CaptionColor (页 710)	RW	-	-	指定显示在所选对象标题中的文本的颜色。
CornerRadius	-	-	-	-
CornerStyle (页 764)	RW	-	-	指定特定对象的角形状。
CountVisibleItems (页 767)	RW	-	-	指定下拉列表中将包含的线数。
CursorControl (页 769)	RW	-	-	指定是否在鼠标光标离开某字段后按 TAB 顺序跳至下一字段。
DeviceStyle	-	-	-	-
DrawInsideFrame (页 787)	RW	-	-	指定线宽大于 1 的所选对象的边框线是在边框内描绘，还是与边框对称。
EdgeStyle (页 788)	RW	-	-	指定所选对象的线样式。
EditOnFocus (页 791)	RW	-	-	指定是否在使用“Tab”键选择输入字段后可以立即输入数据。
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。
EvenRowBackColor	-	-	-	-
ExtraHeightOffset	-	-	-	-
FillPatternColor (页 821)	R	-	-	返回特定对象的填充样式的颜色。
FitToLargest	-	-	-	-
Flashing	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
FlashingColorOff (页 832)	R	-	-	返回闪烁状态为 “关闭”(Off) 时特定对象的边框线颜色。
FlashingColorOn (页 834)	R	-	-	返回闪烁状态为 “开启”(On) 时特定对象的边框线颜色。
FlashingEnabled (页 836)	R	-	-	返回关于特定对象的边框线是否在运行系统中闪烁的信息。
FlashingOnLimitViolation	-	-	-	-
FlashingRate (页 839)	R	-	-	返回特定对象的边框线的闪烁频率。
FocusColor	-	-	-	-
FocusWidth	-	-	-	-
Font	-	-	-	-
FontBold (页 849)	R	-	-	返回关于特定对象的文本是否以粗体显示的信息。
FontItalic (页 849)	R	-	-	返回关于特定对象的文本是否以斜体显示的信息。
FontName (页 851)	R	-	-	返回特定对象的字体。
FontSize (页 852)	R	-	-	返回特定对象的文本字体大小。
FontUnderline (页 853)	R	-	-	返回关于特定对象的文本是否以下划线的形式显示的信息。
ForeColor (页 854)	R	RW	RW	指定所选对象的文本字体颜色。
Height (页 863)	RW	R	R	指定所选对象的高度。
HelpText (页 868)	-	RW	RW	指定作为特定对象的用户帮助显示在运行系统中的工具提示。
HorizontalAlignment (页 881)	R	RW	RW	指定所选对象内的文本水平对齐。
InputValue (页 894)	R	-	-	返回输入值。
ItemBorderStyle (页 897)	R	-	-	返回特定对象的选择列表中分隔线的线型。
Layer (页 913)	R	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	R	RW	RW	指定所选对象的 X 坐标的值。

属性	RT Professional	RT Advanced	面板 RT	描述
LeftMargin	-	-	-	-
LineEndShapeStyle (页 940)	R	-	-	返回线端的形状。
Location	-	-	-	-
LogOperation (页 948)	R	-	-	返回关于操作完该对象后报警是否会输出在报警系统中的信息。
Mode (页 981)	R	-	-	返回指定对象的字段类型。
Name	-	-	-	-
OnValue	-	-	-	-
ProcessValue (页 1037)	R	-	-	返回要显示的数值的默认值。
RightMargin	-	-	-	-
SelectBackColor (页 1074)	R	-	-	返回特定对象的所选文本条目的背景色。
SelectForeColor (页 1083)	R	-	-	返回特定对象的所选文本条目的颜色。
SeparatorBackColor (页 1091)	R	-	-	返回特定对象的选择列表中虚线的背景色。
SeparatorColor (页 1092)	R	-	-	返回特定对象的选择列表中分隔线的颜色。
SeparatorCornerStyle (页 1093)	R	-	-	返回指定对象的角形状。
SeparatorLineEndShapeStyle (页 1093)	R	-	-	返回指定对象的线端形状。
SeparatorStyle (页 1094)	R	-	-	返回特定对象的选择列表中分隔线的线型。
SeparatorWidth (页 1095)	R	-	-	返回特定对象的选择列表中的分隔线宽度。
ShowBadTagState (页 1099)	R	-	-	返回关于出现不良质量代码或变量状态时对象是否呈灰显的信息。
ShowDropDownButton (页 1103)	R	-	-	返回是否显示选择列表的按钮。

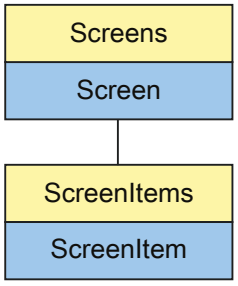
属性	RT Professional	RT Advanced	面板 RT	描述
ShowDropDownList	-	-	-	-
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TextList (页 1186)	R	-	-	返回其中包含各输出值实际要输出的输出文本的列表。
TextOff	-	-	-	-
TextOn	-	-	-	-
TextOrientation (页 1189)	R	-	-	返回特定对象的文本方向。
ToolTipText (页 1278)	R	R	R	返回工具提示文本。
Top (页 1279)	R	R	R	返回特定对象的 Y 坐标的值。
TopMargin	-	-	-	-
Transparency (页 1284)	R	-	-	返回对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	R	-	-	返回关于在当前设计的全局颜色方案中定义的颜色是否用于此对象的信息。
UseDesignShadowSettings (页 1355)	R	-	-	返回关于显示的对象是否带有在激活设计中定义的底纹的信息。
VerticalAlignment (页 1413)	R	RW	RW	指定所选对象中的文本垂直对齐方式。
Visible (页 1418)	R	RW	RW	指定是否显示所选对象。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。

表格 1-100　方法

方法	RT Professional	RT Advanced	面板 RT	描述
Activate (页 1470)	√	√	√	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

SymbolLibrary

说明



表示“SymbolLibrary”对象。SymbolLibrary 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMISymbolLibrary

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-101 属性

属性	RT Professional	RT Advanced	面板 RT	描述
AboveUpperLimitColor	-	-	-	-
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	-	RW	RW	指定操作权限。
BackColor (页 637)	RW	RW	RW	指定背景色。
BackFillStyle (页 643)	RW	-	-	指定填充图案。
BelowLowerLimitColor	-	-	-	-
BlinkColor (页 661)	RW	RW	RW	指定对象在运行系统中闪烁的颜色。
BlinkMode (页 662)	RW	-	-	指定所选对象的闪烁图形的类型。
BlinkSpeed (页 663)	RW	-	-	设置闪烁速度。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
ChangeMouseCursor (页 719)	RW	-	-	指定光标置于图标上方时，它在运行系统中的外观如何变化。
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。
FillColorMode (页 820)	RW	-	-	指定所选对象的前景类型。

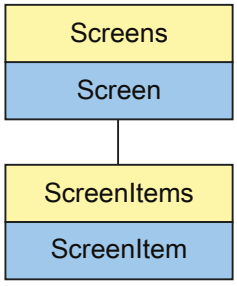
属性	RT Professional	RT Advanced	面板 RT	描述
FixedAspectRatio (页 829)	RW	-	-	指定纵横比在图标大小更改后保持原样还是动态更改。
Flashing	-	-	-	-
FlashingOnLimitViolation	-	-	-	-
Flip (页 842)	RW	-	-	指定是否沿符号的垂直和/或水平中心轴翻转符号。
ForeColor (页 854)	RW	RW	RW	指定文本字体颜色。
Height (页 863)	RW	R	R	指定高度。
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定 X 坐标的值。
Location	-	-	-	-
Name	-	-	-	-
Object	-	-	-	-
OcxGuid	-	-	-	-
OcxState	-	-	-	-
OcxStateForEs2Rt	-	-	-	-
ProcessValue	-	-	-	-
Rotation (页 1047)	RW	-	-	以度为单位指定旋转角度。
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (页 1279)	RW	RW	RW	指定 Y 坐标的值。
Visible (页 1418)	RW	RW	RW	指定是否显示所选对象。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。

表格 1-102 方法

方法	RT Professional	RT Advanced	面板 RT	描述
Activate (页 1470)	√	√	√	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

SysDiagControl

说明



表示“系统诊断视图”对象。SystemDiagnoseView 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMISysDiagView

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Authorization	-	-	-	-
BackgroundColor	-	-	-	-
BorderBackColor	-	-	-	-
BorderColor	-	-	-	-
BorderWidth	-	-	-	-
Bounds	-	-	-	-
BufferViewColumnOrder	-	-	-	-
BufferViewInternalRowOrder	-	-	-	-
BusyText	-	-	-	-
ButtonBackColor	-	-	-	-
ButtonBackFillStyle	-	-	-	-
ButtonBorderBackColor	-	-	-	-
ButtonBorderColor	-	-	-	-
ButtonBorderWidth	-	-	-	-
ButtonCornerRadius	-	-	-	-
ButtonEdgeStyle	-	-	-	-
ButtonFirstGradientColor	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
ButtonFirstGradientOffset	-	-	-	-
ButtonMiddleGradientColor	-	-	-	-
ButtonSecondGradientColor	-	-	-	-
ButtonSecondGradientOffset	-	-	-	-
BV_ColumnWidth_Date	-	-	-	-
BV_ColumnWidth_Event	-	-	-	-
BV_ColumnWidth_EventSeverity	-	-	-	-
BV_ColumnWidth_EventState	-	-	-	-
BV_ColumnWidth_Number	-	-	-	-
BV_ColumnWidth_Time	-	-	-	-
BV_ItemText_Date	-	-	-	-
BV_ItemText_Event	-	-	-	-
BV_ItemText_EventSeverity	-	-	-	-
BV_ItemText_EventState	-	-	-	-
BV_ItemText_Number	-	-	-	-
BV_ItemText_Time	-	-	-	-
BV_ShowItem_Date	-	-	-	-
BV_ShowItem_Event	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
BV_ShowItem_EventSeverity	-	-	-	-
BV_ShowItem_EventState	-	-	-	-
BV_ShowItem_Number	-	-	-	-
BV_ShowItem_Time	-	-	-	-
CanBeGrouped	-	-	-	-
ColumnSettings	-	-	-	-
ColumnSettingsBufferView	-	-	-	-
ComplexViewToolbar	-	-	-	-
ComplexViewToolbarBounds	-	-	-	-
ComponentInfoText	-	-	-	-
CornerRadius	-	-	-	-
DeviceStyle	-	-	-	-
EdgeStyle	-	-	-	-
Enabled (页 792)	R	R	R	返回关于特定对象是否可在运行系统中进行操作的信息。
EnterButtonVisible	-	-	-	-
Es2RtTableBounds	-	-	-	-
EscButtonVisible	-	-	-	-
EsPreviewType	-	-	-	-
FitToSize	-	-	-	-
FitToSizeLowerRows	-	-	-	-
FitToSizeUpperRows	-	-	-	-
Flashing	-	-	-	-
Height (页 863)	RW	R	R	指定高度。
HomeButtonVisible	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
InfoArea_BackgroundColor	-	-	-	-
InfoArea_ColumnsMovable	-	-	-	-
InfoArea_DefaultTextColor	-	-	-	-
InfoArea_ErrorTextBackgroundColor	-	-	-	-
InfoArea_ErrorTextColor	-	-	-	-
InfoArea_FocusFrameColor	-	-	-	-
InfoArea_FocusFrameWidth	-	-	-	-
InfoArea_Font	-	-	-	-
InfoArea_RootNodeText	-	-	-	-
InfoArea_SelectionBackgroundColor	-	-	-	-
InfoArea_SelectionForegroundColor	-	-	-	-
InfoArea_ShowGridLines	-	-	-	-
InfoArea_TableHeaderBackgroundColor	-	-	-	-
InfoArea_TableHeaderTextColor	-	-	-	-
InputAddressText	-	-	-	-
InspectorViewInternalColumnOrder	-	-	-	-
InspectorViewRowOrder	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
ItemText_AKZ	-	-	-	-
ItemText_Descriptor	-	-	-	-
ItemText_ErrorText	-	-	-	-
ItemText_HardwareRevision	-	-	-	-
ItemText_IMDataVersion	-	-	-	-
ItemText_InstallationDate	-	-	-	-
ItemText_LADDR	-	-	-	-
ItemText_ManufacturerID	-	-	-	-
ItemText_Name	-	-	-	-
ItemText_OKZ	-	-	-	-
ItemText_OperationState	-	-	-	-
ItemText_OrderID	-	-	-	-
ItemText_ProfileID	-	-	-	-
ItemText_Rack	-	-	-	-
ItemText_RevisionCounter	-	-	-	-
ItemText_SerialNumber	-	-	-	-
ItemText_Slot	-	-	-	-
ItemText_SoftwareRevision	-	-	-	-
ItemText_SpecificProfileData	-	-	-	-
ItemText_State	-	-	-	-
ItemText_Station	-	-	-	-
ItemText_SubAddress	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
ItemText_SubSlot	-	-	-	-
ItemText_SubSystem	-	-	-	-
ItemText_Type	-	-	-	-
IV_ShowItem_AKZ	-	-	-	-
IV_ShowItem_Descriptor	-	-	-	-
IV_ShowItem_ErrorText	-	-	-	-
IV_ShowItem_HardwareRevision	-	-	-	-
IV_ShowItem_IMDataVersion	-	-	-	-
IV_ShowItem_InstallationDate	-	-	-	-
IV_ShowItem_LADDR	-	-	-	-
IV_ShowItem_ManufacturerID	-	-	-	-
IV_ShowItem_Name	-	-	-	-
IV_ShowItem_OKZ	-	-	-	-
IV_ShowItem_OperationState	-	-	-	-
IV_ShowItem_OrderID	-	-	-	-
IV_ShowItem_ProfileID	-	-	-	-
IV_ShowItem_Rack	-	-	-	-
IV_ShowItem_RevisionCounter	-	-	-	-
IV_ShowItem_SerialNumber	-	-	-	-
IV_ShowItem_Slot	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
IV_ShowItem_SoftwareRevision	-	-	-	-
IV_ShowItem_SpecificProfileData	-	-	-	-
IV_ShowItem_State	-	-	-	-
IV_ShowItem_Station	-	-	-	-
IV_ShowItem_SubAddress	-	-	-	-
IV_ShowItem_SubSlot	-	-	-	-
IV_ShowItem_SubSystem	-	-	-	-
IV_ShowItem_Type	-	-	-	-
Layer (页 913)	R	R	R	在对象所在的画面中，返回图层。
Left (页 920)	R	R	R	返回 X 坐标的值。
LinesPerDiagEntry	-	-	-	-
Location	-	-	-	-
MaxToolbarRows	-	-	-	-
Name	-	-	-	-
NavigateTo (页 988)	RW	-	-	-
NavigationButtons	-	-	-	-
NavigationPath_Font	-	-	-	-
NavigationPath_RootText	-	-	-	-
NavigationPath_TextColor	-	-	-	-
NavigationpathDiagbufferDetailText	-	-	-	-
NavigationpathDiagbufferText	-	-	-	-
OutputAddressText	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
PaddingBottom	-	-	-	-
PaddingLeft	-	-	-	-
PaddingRight	-	-	-	-
PaddingTop	-	-	-	-
RTPersistence	-	-	-	-
RTPersistenceAuthorization	-	-	-	-
RTPersistenceType	-	-	-	-
ShowMilliseconds	-	-	-	-
ShowNavigationButtons	-	-	-	-
ShowPathInformation	-	-	-	-
ShowSplittedView	-	-	-	-
ShowToolbarBackgroundColor	-	-	-	-
SimpleViewToolbar	-	-	-	-
Size	-	-	-	-
SplittedViewRatio	-	-	-	-
StyleItem	-	-	-	-
SupportsInplaceEdit	-	-	-	-
SysDiagBuffButtonVisible	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableEvenRowBackColor	-	-	-	-
TableGridlineColor	-	-	-	-
TableHeaderBackFillStyle	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
TableHeaderBorderBackColor	-	-	-	-
TableHeaderBorderColor	-	-	-	-
TableHeaderBorderWidth	-	-	-	-
TableHeaderCornerRadius	-	-	-	-
TableHeaderEdgeStyle	-	-	-	-
TableHeaderFirstGradientColor	-	-	-	-
TableHeaderFirstGradientOffset	-	-	-	-
TableHeaderFont	-	-	-	-
TableHeaderMiddleGradientColor	-	-	-	-
TableHeaderPaddingBottom	-	-	-	-
TableHeaderPaddingLeft	-	-	-	-
TableHeaderPaddingRight	-	-	-	-
TableHeaderPaddingTop	-	-	-	-
TableHeaderSecondGradientColor	-	-	-	-
TableHeaderSecondGradientOffset	-	-	-	-
ToolBar_ButtonsHeight	-	-	-	-
ToolBar_ButtonsWidth	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
ToolbarAlignment	-	-	-	-
ToolbarBackgroundColor	-	-	-	-
ToolbarButtonClick	-	-	-	-
ToolbarButtonHeight	-	-	-	-
ToolbarButtonSettings	-	-	-	-
ToolbarButtonWidth	-	-	-	-
ToolbarIconStyle	-	-	-	-
Top (页 1279)	R	R	R	返回 Y 坐标的值。
UnitViewColumnOrder	-	-	-	-
UpdateButtonVisible	-	-	-	-
UseButtonFirstGradient	-	-	-	-
UseButtonSecondGradient	-	-	-	-
UseDesignColorSchema	-	-	-	-
UseScadaRendererStyle	-	-	-	-
UseSystemScrollbarWidth	-	-	-	-
UseTableHeaderFirstGradient	-	-	-	-
UseTableHeaderSecondGradient	-	-	-	-
UV_ColumnWidth_AKZ	-	-	-	-
UV_ColumnWidth_Descriptor	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
UV_ColumnWidth_InstallationDate	-	-	-	-
UV_ColumnWidth_LADDR	-	-	-	-
UV_ColumnWidth_Name	-	-	-	-
UV_ColumnWidth_OKZ	-	-	-	-
UV_ColumnWidth_OperationState	-	-	-	-
UV_ColumnWidth_OrderID	-	-	-	-
UV_ColumnWidth_ProfileID	-	-	-	-
UV_ColumnWidth_Rack	-	-	-	-
UV_ColumnWidth_Slot	-	-	-	-
UV_ColumnWidth_SoftwareRevision	-	-	-	-
UV_ColumnWidth_SpecificProfileData	-	-	-	-
UV_ColumnWidth_State	-	-	-	-
UV_ColumnWidth_Station	-	-	-	-
UV_ColumnWidth_SubAddress	-	-	-	-
UV_ColumnWidth_SubSlot	-	-	-	-
UV_ColumnWidth_SubSystem	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
UV_ColumnWidth_Type	-	-	-	-
UV_ShowItem_AKZ	-	-	-	-
UV_ShowItem_Descriptor	-	-	-	-
UV_ShowItem_InstallationDate	-	-	-	-
UV_ShowItem_LADDR	-	-	-	-
UV_ShowItem_Name	-	-	-	-
UV_ShowItem_OKZ	-	-	-	-
UV_ShowItem_OperationState	-	-	-	-
UV_ShowItem_OrderID	-	-	-	-
UV_ShowItem_ProfileID	-	-	-	-
UV_ShowItem_Rack	-	-	-	-
UV_ShowItem_Slot	-	-	-	-
UV_ShowItem_SoftwareRevision	-	-	-	-
UV_ShowItem_SpecificProfileData	-	-	-	-
UV_ShowItem_State	-	-	-	-
UV_ShowItem_Station	-	-	-	-
UV_ShowItem_SubAddress	-	-	-	-
UV_ShowItem_SubSlot	-	-	-	-

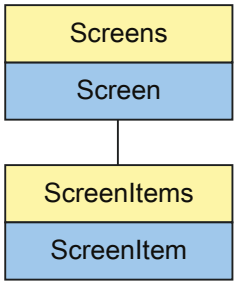
属性	RT Professional	RT Advanced	面板 RT	描述
UV_ShowItem_SubSystem	-	-	-	-
UV_ShowItem_Type	-	-	-	-
ViewType	-	-	-	-
Visible (页 1418)	RW	RW	RW	指定是否显示所选对象。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。

表格 1-103　方法

方法	RT Professional	RT Advanced	面板 RT	描述
Activate (页 1470)	√	√	√	启用永久区域或根画面。

TextField

说明



表示“TextField”对象。TextField 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMITextField

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-104 属性

属性	RT Professional	RT Advanced	面板 RT	描述
AdaptBorder (页 606)	RW	-	-	指定对象的边框是否根据文本的大小进行动态调整。
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定操作权限。
BackColor (页 637)	RW	RW	RW	指定背景色。
BackFillStyle (页 643)	RW	-	-	指定填充图案。
BackFlashingColorOff (页 645)	RW	-	-	指定闪烁状态“关闭”(Off) 的背景色。
BackFlashingColorOn (页 646)	RW	-	-	指定闪烁状态“打开”(On) 的背景色。
BackFlashingEnabled (页 648)	RW	-	-	指定是否在运行系统中闪烁背景。
BackFlashingRate (页 649)	RW	-	-	指定背景的闪烁频率。
BorderBackColor (页 674)	RW	-	-	指定间断边框线的背景色。

属性	RT Professional	RT Advanced	面板 RT	描述
BorderColor (页 678)	RW	-	RW	指定线颜色。
BorderFlashingColorOff (页 683)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
BorderFlashingColorOn (页 685)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
BorderFlashingEnabled (页 687)	RW	-	-	指定是否在运行系统中闪烁对象的限值。
BorderFlashingRate (页 689)	RW	-	-	指定边界线的闪烁频率。
BorderStyle (页 695)	RW	-	-	指定边框线类型。
BorderWidth (页 697)	RW	RW	RW	指定线宽。
BottomMargin	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerRadius	-	-	-	-
CornerStyle (页 764)	RW	-	-	指定角形状。
DrawInsideFrame (页 787)	RW	-	-	指定线宽大于 1 的将要显示的边框线是在边框内，还是与边框对称。
EdgeStyle (页 788)	RW	RW	RW	指定线样式。
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。
FillPatternColor (页 821)	RW	-	-	确定填充图案的颜色。
FitToLargest	-	-	-	-
Flashing	-	-	-	-
FlashingColorOff (页 832)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
FlashingColorOn (页 834)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。

属性	RT Professional	RT Advanced	面板 RT	描述
FlashingEnabled (页 836)	RW	-	-	指定是否为运行系统中的对象激活闪烁。
FlashingRate (页 839)	RW	-	-	指定边界线的闪烁频率。
Font	-	-	-	-
FontBold (页 849)	RW	-	-	指定文本是否以粗体显示。
FontItalic (页 849)	RW	-	-	指定文本是否以斜体显示。
FontName (页 851)	RW	-	-	指定字体。
FontSize (页 852)	RW	-	-	指定文本字体大小。
FontUnderline (页 853)	RW	-	-	指定文本是否应以下划线标出。
ForeColor (页 854)	RW	RW	RW	指定文本字体颜色。
Height (页 863)	RW	R	R	指定高度。
HorizontalAlignment (页 881)	RW	RW	RW	指定文本水平对齐。
Layer (页 913)	R	R	R	在对象所在的画面中，返回图层。
Left (页 920)	RW	RW	RW	指定 X 坐标的值。
LeftMargin	-	-	-	-
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
LineWrap	-	-	-	-
Location	-	-	-	-
Name	-	-	-	-
RelativeFillLevel (页 1046)	RW	-	-	指定对象的填充百分比。
ReSizeable	-	-	-	-
RightMargin	-	-	-	-
RotationAngle (页 1048)	RW	-	-	以度为单位指定旋转角度。
RotationCenterLeft (页 1049)	RW	-	-	指定运行系统中对象旋转中心点的 X 坐标。

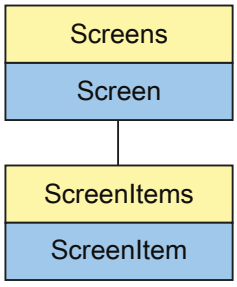
属性	RT Professional	RT Advanced	面板 RT	描述
RotationCenterTop (页 1050)	RW	-	-	指定运行系统中对象旋转中心点的 Y 坐标。
ShowFillLevel (页 1108)	RW	-	-	指定所选对象是否填充。
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Text (页 1185)	RW	RW	RW	指定文本域的标签。
TextOrientation (页 1189)	RW	-	-	指定文本方向。
ToolTipText (页 1278)	RW	R	R	指定工具提示文本。
Top (页 1279)	RW	RW	RW	指定 Y 坐标的值。
TopMargin	-	-	-	-
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
VerticalAlignment (页 1413)	RW	RW	RW	指定所选对象中的文本垂直对齐方式。
Visible (页 1418)	RW	RW	RW	指定是否显示所选对象。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。

表格 1-105 方法

方法	RT Professional	RT Advanced	面板 RT	描述
Activate (页 1470)	√	√	√	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

TrendRulerControl

描述



表示“Value table”对象。TrendRulerControl 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMITrendRulerControl

示例

在下面的示例中，名称为"Control1"的对象向右移动 16 个像素：

```
'VBS60
Dim objControl
Set objControl = ScreenItems("Control1")
objControl.Left = objControl.Left + 16
```

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
ApplyProjectSettingsForDesignMode	-	-	-	-
AutoCompleteColumns (页 629)	RW	-	-	指定在控件比组态的列宽时是否显示空列。
AutoCompleteRows (页 629)	RW	-	-	指定在控件比组态的行数长时是否显示空行。
AutoPosition (页 630)	RW	-	-	指定数值表是否自动放置于向数值表提供数据的对象下。
AutoSelectionColors (页 632)	RW	-	-	指定是否将系统定义的颜色用作单元格和行的选择颜色。
AutoSelectionRectColor (页 632)	RW	-	-	指定是否使用系统定义的颜色显示选择框架。
AutoShow (页 633)	RW	-	-	指定是否自动显示数值表。
BackColor (页 637)	RW	-	-	指定背景色。
BlockAlignment (页 664)	RW	-	-	指定引用块的列标题内的文本对齐方式。

属性	RT Professional	RT Advanced	面板 RT	描述
BlockAutoPrecision (页 664)	RW	-	-	指定是否自动调整当前块中显示的小数位数。
BlockCaption (页 665)	RW	-	-	指定当前块的标题。
BlockCount (页 666)	RW	-	-	设置块数。
BlockDateFormat (页 666)	RW	-	-	指定当前块的日期格式。
BlockExponentialFormat (页 667)	RW	-	-	指定是否在指数计数法中显示当前块中的值。
BlockHideText	-	-	-	-
BlockHideTitleText (页 668)	RW	-	-	指定该块的标题是否以文本形式显示。
BlockId (页 668)	RW	-	-	使用其 ID 引用一个块。
BlockIndex (页 669)	RW	-	-	引用块。
BlockLength (页 669)	RW	-	-	指定当前块的字符数。
BlockName (页 670)	RW	-	-	指定当前块的名称。
BlockPrecision (页 671)	RW	-	-	指定当前块中的小数位数。
Blocks	-	-	-	-
BlockShowDate (页 671)	RW	-	-	指定在当前块中将显示的日期。
BlockShowIcon (页 672)	RW	-	-	指定是否将当前块的内容显示为图标。
BlockShowTitleIcon (页 673)	RW	-	-	指定当前块的标题是否以图标形式显示。
BlockTimeFormat (页 673)	RW	-	-	指定当前块的时间格式。
BlockUseSourceFormat (页 674)	RW	-	-	指定是否在当前块中使用所连接控件的格式。

属性	RT Professional	RT Advanced	面板 RT	描述
BorderColor (页 678)	RW	-	-	指定线颜色。
BorderWidth (页 697)	RW	-	-	指定线宽。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Caption (页 708)	RW	-	-	指定标题中将要显示的文本。
CellCut (页 713)	RW	-	-	指定单元格过窄时是否缩略单元格内容。
CellSpaceBottom (页 714)	RW	-	-	指定表格单元格的下边距。
CellSpaceLeft (页 715)	RW	-	-	指定表格单元格中采用的左缩进。
CellSpaceRight (页 716)	RW	-	-	指定表格单元格中采用的右缩进。
CellSpaceTop (页 717)	RW	-	-	指定表格单元格的上边距。
Closeable (页 722)	RW	-	-	指定是否可在运行系统中关闭对象。
ColumnAdd (页 726)	RW	-	-	创建一个新列。
ColumnCount (页 729)	RW	-	-	指定所组态列的数目。
ColumnIndex (页 734)	RW	-	-	引用一个列。
ColumnName (页 737)	RW	-	-	指定使用“ColumnIndex”引用的列的名称。
ColumnRemove (页 740)	RW	-	-	移除一个引用了其名称的列。
ColumnRepos (页 741)	RW	-	-	对于多个列，指定使用“ColumnIndex”引用的列的位置。
ColumnResize (页 742)	RW	-	-	指定是否可以更改列宽。
Columns	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
ColumnScrollbar (页 743)	RW	-	-	指定将显示水平滚动条的时间。
ColumnSort (页 746)	RW	-	-	指定使用“ColumnIndex”引用的列的排序类型。
ColumnSortIndex (页 747)	RW	-	-	指定使用“ColumnIndex”引用的列的排序顺序，其中列以一个接一个的方式排序。
ColumnTitleAlignment (页 754)	RW	-	-	指定使用“ColumnIndex”引用的列标题的对齐方式。
ColumnTitles (页 755)	RW	-	-	指定是否显示列标题。
ColumnVisible (页 757)	RW	-	-	指定是否在对象中显示使用“ColumnIndex”引用的列。
ControlDesignMode (页 762)	RW	-	-	指定控制设计。
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。
ExportDelimiter	-	-	-	-
ExportDirectoryChangeable (页 808)	RW	-	-	指定是否可以在运行系统中更改数据导出目录。
ExportDirectoryname (页 809)	RW	-	-	指定导出的运行系统数据的目标目录。
ExportFileExtension (页 810)	RW	-	-	指定导出文件的文件扩展名。
ExportFilename (页 811)	RW	-	-	指定导出的运行系统数据的目标文件名称。
ExportFilenameChangeable (页 812)	RW	-	-	指定是否可以在运行系统中更改导出文件名。
ExportFormat	-	-	-	-
ExportFormatGuid (页 813)	RW	-	-	指定 ID 编号和导出数据源的分配。
ExportFormatName (页 814)	RW	-	-	指定导出文件格式。

属性	RT Professional	RT Advanced	面板 RT	描述
ExportParameters (页 815)	RW	-	-	通过属性对话框指定所选格式的参数。
ExportSelection (页 816)	RW	-	-	指定导出控件的哪些运行系统数据。
ExportShowDialog (页 817)	RW	-	-	指定是否在运行系统中显示数据导出对话框。
FillPattern	-	-	-	-
FillPatternColor	-	-	-	-
Font (页 846)	RW	-	-	指定字体。
GridLineColor (页 860)	RW	-	-	指定网格线颜色。
GridLineWidth (页 862)	RW	-	-	指定分隔线的宽度（以像素为单位）。
Height (页 863)	RW	-	-	指定高度。
HorizontalGridLines (页 883)	RW	-	-	指定是否显示水平分隔线。
IconSpace (页 886)	RW	-	-	指定表格单元格中图标和文本的间距。
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
LineBackgroundColor	-	-	-	-
LineColor (页 939)	RW	-	-	指定窗口分隔线的颜色。
LineStyle	-	-	-	-
LineWidth (页 943)	RW	-	-	指定线宽。
Location	-	-	-	-
Moveable (页 983)	RW	-	-	指定运行系统中是否可以移动对象。
Name	-	-	-	-
Object	-	-	-	-
OcxGuid	-	-	-	-
OcxState	-	-	-	-
OcxStateForEs2Rt	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
PrintJob (页 1036)	RW	-	-	指定一个在“报表”编辑器中创建的打印作业。
RowScrollbar (页 1052)	RW	-	-	指定垂直滚动条将要显示的时间。
RowTitleAlignment (页 1053)	RW	-	-	指定行标题对齐的类型。
RowTitles (页 1054)	RW	-	-	指定是否显示已编号的列标题。
RTPersistence (页 1054)	RW	-	-	指定在画面更改后是否保留在线组态。
RTPersistenceAuthorization (页 1055)	RW	-	-	指定运行系统中在线组态所需的权限。
RTPersistenceType (页 1057)	RW	-	-	指定如何保留 WinCC 的在线组态。
RulerColumns	-	-	-	-
RulerType (页 1059)	RW	-	-	指定数值表的显示模式。
SelectedCellColor (页 1075)	RW	-	-	指定所选单元格的背景色。
SelectedCellForeColor (页 1076)	RW	-	-	指定所选单元格的字体颜色。
SelectedRowColor (页 1079)	RW	-	-	指定所选行的背景色。
SelectedRowForeColor (页 1080)	RW	-	-	指定所选行的字体颜色。
SelectedTitleColor (页 1081)	RW	-	-	指定所选表格标题的背景色。
SelectedTitleForeColor (页 1082)	RW	-	-	指定所选表格标题的字体颜色。
SelectionColoring (页 1085)	RW	-	-	指定是否对单元格或行使用选择颜色。
SelectionRect (页 1087)	RW	-	-	指定是否对所选单元格或行使用选择框架。
SelectionRectColor (页 1088)	RW	-	-	指定报警窗口中选择矩形的颜色。

属性	RT Professional	RT Advanced	面板 RT	描述
SelectionRectWidth (页 1089)	RW	-	-	如果 SelectionType 等于"1"，则指定报警窗口中选择矩形的线宽。
SelectionType (页 1090)	RW	-	-	指定可标记的行数。
ShareSpaceWithSourceControl (页 1097)	RW	-	-	指定数据源的显示区域应根据数值表的大小而调整。
ShowSortButton (页 1117)	RW	-	-	指定是否在垂直滚动条上显示排序按钮。
ShowSortIcon (页 1118)	RW	-	-	指定是否显示排序图标。
ShowSortIndex (页 1118)	RW	-	-	指定是否显示排序索引。
ShowTitle (页 1124)	RW	-	-	为对象指定窗口边框和窗口标题的样式。
Size	-	-	-	-
Sizeable (页 1129)	RW	-	-	指定可以在运行系统中更改对象的大小。
SortSequence (页 1132)	RW	-	-	如果操作员在运行系统中单击列标题，指定排序顺序会如何改变。
SourceControl (页 1133)	RW	-	-	指定数据表所连接的趋势或表格视图。
SourceControlType (页 1133)	RW	-	-	设置数据源的类型。
StatisticAreaColumns	-	-	-	-
StatisticResultColumns	-	-	-	-
StatusbarBackColor (页 1138)	RW	-	-	指定状态栏的背景色。
StatusbarElementAdd (页 1139)	RW	-	-	创建一个新的用户自定义状态栏元素。
StatusbarElementAutoSize (页 1140)	RW	-	-	指定是否自动设置引用"StatusbarElementIndex"的状态栏元素的宽度。

属性	RT Professional	RT Advanced	面板 RT	描述
StatusbarElementCount (页 1141)	RW	-	-	指定已组态的状态栏元素的数量。
StatusbarElementIconId (页 1142)	RW	-	-	使用其图标 ID 来引用状态栏元素。
StatusbarElementID (页 1143)	RW	-	-	使用其元素 ID 来引用状态栏元素。
StatusbarElementIndex (页 1144)	RW	-	-	引用状态栏元素。
StatusbarElementName (页 1145)	RW	-	-	指定使用"StatusbarElementIndex"引用的状态栏元素的名称。
StatusbarElementRemove (页 1146)	RW	-	-	删除引用其名称的用户自定义状态栏元素。
StatusbarElementRename (页 1147)	RW	-	-	指定使用"StatusbarElementIndex"引用的用户自定义的状态栏元素的新名称。
StatusbarElementRepos (页 1148)	RW	-	-	指定对象状态栏中使用"StatusbarElementIndex"引用的元素位置。
StatusbarElements (页 1149)	-	-	-	-
StatusbarElementText (页 1149)	RW	-	-	指定使用"StatusbarElementIndex"引用的用户自定义的状态栏元素的文本。
StatusbarElementTooltipText (页 1150)	RW	-	-	指定使用"StatusbarElementIndex"引用的用户自定义的状态栏元素的工具提示文本。
StatusbarElementUserDefined (页 1151)	RW	-	-	指定组态工程师是否已添加使用"StatusbarElementIndex"引用的状态栏元素作为新的用户自定义元素。
StatusbarElementVisible (页 1152)	RW	-	-	指定是否在对象中显示使用"StatusbarElementIndex"引用的状态栏元素。
StatusbarElementWidth (页 1153)	RW	-	-	指定使用"StatusbarElementIndex"引用的状态栏元素的宽度（以像素为单位）。
StatusbarFont (页 1154)	RW	-	-	指定状态栏中文本的字体。
StatusbarFontColor (页 1155)	RW	-	-	指定状态栏中文本的颜色。

属性	RT Professional	RT Advanced	面板 RT	描述
StatusbarShowTooltips (页 1156)	RW	-	-	指定是否在运行系统中显示状态栏元素的工具提示。
StatusbarText (页 1157)	RW	-	-	指定状态栏中的默认文本。
StatusbarUseBackColor (页 1158)	RW	-	-	指定是否显示状态栏的背景色。
StatusbarVisible (页 1159)	RW	-	-	指定是否显示控件的状态栏。
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableColor (页 1171)	RW	-	-	指定对象中表格行的背景色。
TableColor2 (页 1172)	RW	-	-	指定对象中表格行的第二种背景色。
TableForeColor (页 1173)	RW	-	-	指定对象表格单元格中采用的文本颜色。
TableForeColor2 (页 1174)	RW	-	-	指定对象表格单元格中采用的第二种文本颜色。
TitleColor (页 1236)	RW	-	-	指定表格标题的背景色。
TitleCut (页 1237)	RW	-	-	指定列宽过窄时是否缩减标题栏中的字段内容。
TitleDarkShadowColor (页 1238)	RW	-	-	为对象列表中的列和行标题指定 3D 底纹的暗边颜色。
TitleForeColor (页 1239)	RW	-	-	为对象的表格列和行标题指定文本颜色。
TitleGridLineColor (页 1240)	RW	-	-	指定表格标题栏中分隔线的颜色。
TitleLightShadowColor (页 1241)	RW	-	-	为对象的表格列和行标题指定 3D 底纹的亮边颜色。
TitleSort (页 1242)	RW	-	-	指定如何触发按列标题排序。
TitleStyle (页 1243)	RW	-	-	指定是否使用列标题文本的底纹颜色。
ToolbarAlignment (页 1252)	RW	-	-	指定工具栏的位置。

属性	RT Professional	RT Advanced	面板 RT	描述
ToolbarBackColor (页 1253)	RW	-	-	指定工具栏的背景色。
ToolbarButtonActive (页 1254)	RW	-	-	指定是否在运行系统中激活按钮的相应功能。
ToolbarButtonAdd (页 1255)	RW	-	-	在对象的工具栏中创建一个新的用户自定义按钮。
ToolbarButtonAuthorization (页 1256)	RW	-	-	指定所选按键功能的权限。
ToolbarButtonBeginGroup (页 1257)	RW	-	-	在工具栏上，为所选按键功能插入前导分隔符（垂直线）。
ToolbarButtonClick (页 1258)	RW	-	-	单击工具栏按钮。
ToolbarButtonCount (页 1259)	RW	-	-	指定工具栏中的已组态按钮数目。
ToolbarButtonEnabled (页 1260)	RW	-	-	指定是否可以操作用户自定义工具栏按钮。
ToolbarButtonHotKey (页 1261)	RW	-	-	指定所选对象按钮的热键。
ToolbarButtonID (页 1262)	RW	-	-	使用其 ID 引用按钮。
ToolbarButtonIndex (页 1263)	RW	-	-	引用按钮。
ToolbarButtonLocked (页 1264)	RW	-	-	指定是否显示使用“ToolbarButtonIndex”引用的用户自定义按钮的锁定、按下状态。
ToolbarButtonName (页 1265)	RW	-	-	指定使用“ToolbarButtonIndex”引用的用户自定义按钮的名称。
ToolbarButtonRemove (页 1266)	RW	-	-	移除一个引用了其名称的用户自定义按钮。
ToolbarButtonRename (页 1267)	RW	-	-	指定使用“ToolbarButtonIndex”引用的用户自定义按钮的新名称。
ToolbarButtonRepos (页 1268)	RW	-	-	在对象的工具栏中指定使用“ToolbarButtonIndex”引用的按钮位置。
ToolbarButtons	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
ToolbarButtonTooltipText (页 1270)	RW	-	-	指定工具栏中用户自定义按钮的工具提示文本。
ToolbarButtonUserDefined (页 1271)	RW	-	-	指定项目工程师是否已将此工具栏按钮添加为新的用户自定义按钮。
ToolbarButtonVisible (页 1272)	RW	-	-	指定是否在工具栏中显示使用“ToolbarButtonIndex”引用的按钮。
ToolbarShowTooltips (页 1273)	RW	-	-	指定是否在运行系统中显示按键功能的工具提示。
ToolbarUseBackColor (页 1275)	RW	-	-	指定是否显示工具栏的背景色。
ToolbarUseHotKeys (页 1276)	RW	-	-	指定是否激活工具栏中按钮的热键。
ToolbarVisible (页 1277)	RW	-	-	指定是否显示控件的工具栏。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。
UseSelectedTitleColor (页 1362)	RW	-	-	指定是否将选择颜色用于选定表格单元格的标题。
UseSourceBackColors (页 1363)	RW	-	-	指定使用数据源的背景色。
UseSourceForeColors (页 1363)	RW	-	-	指定使用数据源的字体颜色。
UseTableColor2 (页 1364)	RW	-	-	指定是否用第二种行颜色来表现表格。
VerticalGridLines (页 1414)	RW	-	-	指定是否显示垂直分隔线。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

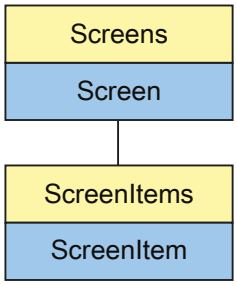
表格 1-106 方法

方法	RT Professional	RT Advanced	面板 RT	描述
激活 (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。
导出 (页 1485)	√	-	-	执行控件的“导出日志”(Export log) 或“导出数据”(Export data) 按键功能。
GetRow (页 1498)	√	-	-	以“ICCAxDataRow”类型的形式返回基于表格的控件中的行编号所指定的行对象。
GetRowCollection (页 1499)	√	-	-	以“ICCAxDataRowCollection”类型的形式返回基于表格的控件的所有行对象的列表。
GetRulerBlock (页 1501)	√	-	-	以“ICCAxRulerBlock”类型的形式返回评估表中具有指定名称或索引的块对象。
GetRulerBlockCollection (页 1502)	√	-	-	以“ICCAxCollection”类型的形式返回评估表中所有块对象的列表。
GetRulerColumn (页 1503)	√	-	-	以“ICCAxRulerColumn”类型的形式返回评估表中具有指定名称或索引的列对象。
GetRulerColumnCollection (页 1504)	√	-	-	以 ICCAxCollection”类型的形式返回评估表中所有列对象的列表。
GetSelectedRow (页 1507)	√	-	-	以“ICCAxDataRow”类型的形式返回基于表格的控件的所选行对象。
GetSelectedRows (页 1508)	√	-	-	以“ICCAxDataRow”类型的形式返回用于多项选择的基于表格的控件的所选行对象。
GetStatisticAreaColumn (页 1510)	√	-	-	以“ICCAxRulerColumn”类型的形式返回评估表的统计区域窗口中具有指定名称或索引的列对象。
GetStatisticAreaColumnCollection (页 1511)	√	-	-	以“ICCAxCollection”类型的形式返回评估表的统计区域窗口中所有列对象的列表。

方法	RT Professional	RT Advanced	面板 RT	描述
GetStatisticResultColumn (页 1512)	√	-	-	以“ICCAxRulerColumn”类型的形式返回评估表中具有指定名称或索引的统计窗口的列对象。
GetStatisticResultColumnCollection (页 1513)	√	-	-	以“ICCAxCollection”类型的形式返回评估表的统计窗口中所有列对象的列表。
GetStatusbarElement (页 1514)	√	-	-	以“ICCAxStatusbarElement”类型的形式返回按名称或索引指定的控件状态栏元素。
GetStatusbarElementCollection (页 1516)	√	-	-	以类型“ICCAxCollection”形式返回控件的所有状态栏元素的列表。
GetToolbarButton (页 1523)	√	-	-	以“ICCAxToolbarButton”类型的形式返回控件工具栏上按名称或索引指定的按钮功能。
GetToolbarButtonCollection (页 1525)	√	-	-	以“ICCAxCollection”类型的形式返回控件工具栏的所有按钮功能的列表。
Print (页 1557)	√	-	-	执行控件的 “打印”(Print) 按钮功能。
SelectAll (页 1573)	√	-	-	在基于表格的控件中选择所有行。
SelectRow (页 1575)	√	-	-	在基于表格的控件中选择特定行。
ShowHelp (页 1580)	√	-	-	执行控件的 “帮助”(Help) 按钮功能。
ShowPropertyDialog (页 1585)	√	-	-	执行控件的 “组态对话框”(Configuration dialog) 按钮功能。
UnselectAll (页 1595)	√	-	-	在基于表格的控件的单元格中移除所有选择内容。
UnselectRow (页 1595)	√	-	-	在基于表格的控件的特定单元格中移除选择内容。

TrendView

说明



表示“TrendView”对象。TrendView 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMITrendView

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-107 属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	-	RW	RW	指定所选对象在运行系统中的操作权限。
AxisXBunchCount	-	-	-	-
AxisXMarkCount	-	-	-	-
AxisXNoOfDigits	-	-	-	-
AxisXShowBunchValues	-	-	-	-
AxisXStyle	-	-	-	-
AxisY1BunchCount	-	-	-	-
AxisY1MarkCount	-	-	-	-
AxisY1ShowBunchValues	-	-	-	-
AxisY2BunchCount	-	-	-	-
AxisY2MarkCount	-	-	-	-
AxisY2ShowBunchValues	-	-	-	-
BackColor	-	-	-	-
BorderBackColor	-	-	-	-
BorderColor	-	-	-	-
BorderWidth	-	-	-	-
Bounds	-	-	-	-
ButtonBackColor	-	-	-	-
ButtonBackFillStyle	-	-	-	-
ButtonBarHeight	-	-	-	-
ButtonBorderBackColor	-	-	-	-
ButtonBorderColor	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
ButtonBorderWidth	-	-	-	-
ButtonCornerRadius	-	-	-	-
ButtonEdgeStyle	-	-	-	-
ButtonFirstGradientColor	-	-	-	-
ButtonFirstGradientOffset	-	-	-	-
ButtonMiddleGradientColor	-	-	-	-
ButtonPositions	-	-	-	-
ButtonSecondGradientColor	-	-	-	-
ButtonSecondGradientOffset	-	-	-	-
CanBeGrouped	-	-	-	-
ColumnOrder	-	-	-	-
ColumnsMoveable	-	-	-	-
ColumnTextDateTime	-	-	-	-
ColumnTextTagConnection	-	-	-	-
ColumnTextTrend	-	-	-	-
ColumnTextValue	-	-	-	-
ColumnTextXValue	-	-	-	-
ColumnWidth	-	-	-	-
CornerRadius	-	-	-	-
CountVisibleItems	-	-	-	-
Curves	-	-	-	-
DeviceStyle	-	-	-	-
DiagramAreaHeight	-	-	-	-
DiagramAreaLeft	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
DiagramAreaTop	-	-	-	-
DiagramAreaWidth	-	-	-	-
EdgeStyle	-	-	-	-
Enabled (页 792)	-	RW	RW	指定是否可以在运行系统中操作所选对象。
EnableNavigateButtons	-	-	-	-
EnableNavigateKeys	-	-	-	-
ES2RT_ColumnOrder	-	-	-	-
ES2RT_ColumnWidth	-	-	-	-
FitToSize	-	-	-	-
Flashing	-	-	-	-
FocusColor (页 843)	-	RW	RW	指定聚焦所选对象时其焦点框的颜色。
FocusWidth (页 844)	-	RW	RW	指定聚焦指定对象时其边框宽度。
Font	-	-	-	-
GridlineAxis	-	-	-	-
GridlineColor	-	-	-	-
GridlineEnabled	-	-	-	-
GridlineFillColor	-	-	-	-
GridlineStyle	-	-	-	-
Height	-	R	R	指定高度。
HelpText (页 868)	-	RW	RW	指定作为特定对象的用户帮助显示在运行系统中的工具提示。
IsRunningUnderCE	-	-	-	-
Layer (页 913)	-	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	-	RW	RW	指定所选对象的 X 坐标的值。
Location	-	-	-	-
Look3D	-	-	-	-
MaxNrOfCurves	-	-	-	-
MinNrOfCurves	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
Name	-	-	-	-
PaddingBottom	-	-	-	-
PaddingLeft	-	-	-	-
PaddingRight	-	-	-	-
PaddingTop	-	-	-	-
RulerColor (页 1058)	-	RW	RW	定义特定对象中轴标签的刻度分度（帮助线）颜色。
ScaleColor (页 1060)	-	RW	RW	指定所选对象的刻度的颜色。
SelectionBackColor	-	-	-	-
SelectionForeColor	-	-	-	-
ShowRuler (页 1114)	-	RW	RW	定义是否为特定对象的轴标签显示刻度分度（帮助线）。
ShowTableGridLines	-	-	-	-
ShowTimeAxis	-	-	-	-
ShowTimeAxisLabeling	-	-	-	-
ShowValueAxis1	-	-	-	-
ShowValueAxis1Label	-	-	-	-
ShowValueAxis2	-	-	-	-
ShowValueAxis2Label	-	-	-	-
ShowValueTable	-	-	-	-
ShowY1HlpLine	-	-	-	-
ShowY2HlpLine	-	-	-	-
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableBackColor (页 1170)	-	RW	RW	指定所选对象的表格单元格的背景色。

属性	RT Professional	RT Advanced	面板 RT	描述
TableColumnsWidthAndOrder	-	-	-	-
TableEvenRowBackColor	-	-	-	-
TableFont	-	-	-	-
TableGridLineColor (页 1175)	-	R	R	返回所选对象的表格的网格线颜色。
TableHeaderBackColor (页 1176)	-	R	R	返回所选对象表格标题的背景色。
TableHeaderBackFillStyle	-	-	-	-
TableHeaderBorderBackColor	-	-	-	-
TableHeaderBorderColor	-	-	-	-
TableHeaderBorderWidth	-	-	-	-
TableHeaderCornerRadius	-	-	-	-
TableHeaderEdgeStyle	-	-	-	-
TableHeaderFirstGradientColor	-	-	-	-
TableHeaderFirstGradientOffset	-	-	-	-
TableHeaderFont	-	-	-	-
TableHeaderForeColor (页 1179)	-	R	R	返回所选对象表格标题的文本颜色。
TableHeaderMiddleGradientColor	-	-	-	-
TableHeaderPaddingBottom	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
TableHeaderPaddingLeft	-	-	-	-
TableHeaderPaddingRight	-	-	-	-
TableHeaderPaddingTop	-	-	-	-
TableHeaderSecondGradientColor	-	-	-	-
TableHeaderSecondGradientOffset	-	-	-	-
TagForExternalTime	-	-	-	-
TimeAxisBegin	-	-	-	-
TimeAxisBeginTime	-	-	-	-
TimeAxisCountPoints	-	-	-	-
TimeAxisEnd	-	-	-	-
TimeAxisMode	-	-	-	-
TimeAxisRange	-	-	-	-
TimeAxisSide	-	-	-	-
TimeAxisTimeRange	-	-	-	-
ToolbarButtons	-	-	-	-
ToolbarButtonsForMigration	-	-	-	-
ToolbarEnabled	-	-	-	-
ToolbarHeight	-	-	-	-
ToolbarLeft	-	-	-	-
ToolbarStyle	-	-	-	-
ToolbarTop	-	-	-	-
ToolbarWidth	-	-	-	-
Top (页 1279)	-	R	R	返回所选对象的 Y 坐标的值。
TrendsForPrinting	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
UseButtonFirstGradient	-	-	-	-
UseButtonSecondGradient	-	-	-	-
UseDesignColorSchema	-	-	-	-
UseTableHeaderFirstGradient	-	-	-	-
UseTableHeaderSecondGradient	-	-	-	-
ValueAxis1AutoRange	-	-	-	-
ValueAxis1Begin	-	-	-	-
ValueAxis1End	-	-	-	-
ValueAxis1LabelLength	-	-	-	-
ValueAxis1Style	-	-	-	-
ValueAxis2AutoRange	-	-	-	-
ValueAxis2Begin	-	-	-	-
ValueAxis2End	-	-	-	-
ValueAxis2LabelLength	-	-	-	-
ValueAxis2Style	-	-	-	-
ValueTableHeight	-	-	-	-
ValueTableLeft	-	-	-	-
ValueTableTop	-	-	-	-
ValueTableWidth	-	-	-	-
ValueY1HlpLine	-	-	-	-
ValueY2HlpLine	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
Visible (页 1418)	-	R	R	指定是否显示所选对象。
Width	-	R	R	指定宽度。

表格 1-108 方法

方法	RT Professional	RT Advanced	面板 RT	描述
Activate (页 1470)	-	√	√	启用永久区域或根画面。

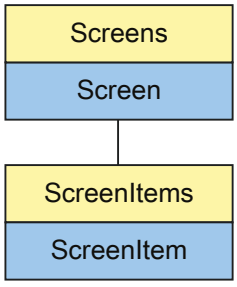
参见

Height (页 863)

Width (页 1432)

TubeArcObject

描述



表示“TubeArc”对象。TubeArcObject 是 ScreenItems 列表的元素。

VBS 的类型标识符

HMITubeArcObject

应用

在下面的示例中，名称为“TubeArcObject1”的对象向右移动 10 个像素：

```
'VBS24
Dim objTubeArcObject
Set objTubeArcObject = ScreenItems("TubeArcObject1")
objTubeArcObject.Left = objTubeArcObject.Left + 10
```

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-109 属性

属性	RT Professional	RT Advanced	Panel RT	描述
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定操作权限。
BackColor	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Color (页 723)	RW	-	-	指定线颜色。
CornerStyle	-	-	-	-
DrawInsideFrame	-	-	-	-
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。

属性	RT Professional	RT Advanced	Panel RT	描述
EndAngle (页 799)	RW	-	-	指定所选对象端点偏离零位置 (0°) 的角度。
Flashing	-	-	-	-
FlashingColorOff	-	-	-	-
FlashingColorOn	-	-	-	-
FlashingEnabled	-	-	-	-
FlashingRate	-	-	-	-
Height (页 863)	RW	-	-	指定高度。
层 (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
LineEndShapeStyle	-	-	-	-
LineWidth (页 943)	RW	-	-	指定线宽。
Location	-	-	-	-
Name	-	-	-	-
RadiusHeight (页 1041)	RW	-	-	指定副轴。
RadiusWidth (页 1042)	RW	-	-	指定主轴。
Size	-	-	-	-
StartAngle (页 1134)	RW	-	-	指定起始点偏离零位置 (0°) 的角度。
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (页 1278)	RW	-	-	指定工具提示文本。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。

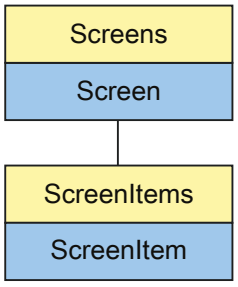
属性	RT Professional	RT Advanced	Panel RT	描述
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-110 方法

方法	RT Professional	RT Advanced	Panel RT	描述
Activate (页 1470)	可用	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	可用	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	可用	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

TubeDoubleTeeObject

描述



表示“DoubleTee”对象。TubeDoubleTeeObject 是 ScreenItems 列表的元素。

VBS 的类型标识符

HMITubeDoubleTeeObject

应用

在下面的示例中，名称为“TubeDoubleTeeObject1”的对象向右移动 10 个像素：

```
'VBS21
Dim objTubeDoubleTeeObject
Set objTubeDoubleTeeObject = ScreenItems("TubeDoubleTeeObject1")
objTubeDoubleTeeObject.Left = objTubeDoubleTeeObject.Left + 10
```

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-111 属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定操作权限。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Color (页 723)	RW	-	-	指定线颜色。
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。
Height (页 863)	RW	-	-	指定高度。
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
LineWidth (页 943)	RW	-	-	指定线宽。

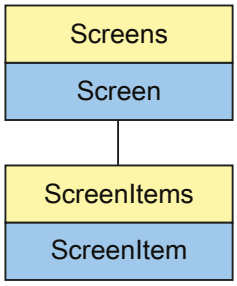
属性	RT Professional	RT Advanced	面板 RT	描述
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (页 1278)	RW	-	-	指定工具提示文本。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-112 方法

方法	RT Professional	RT Advanced	面板 RT	描述
Activate (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

TubePolyline

说明



表示“Pipe”对象。TubePolyline 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMITubePolyline

应用

在下面的示例中，名称为“TubePolyline1”的对象向右移动 10 个像素：

```
'VBS24
Dim objTubePolyline
Set objTubePolyline = ScreenItems("TubePolyline1")
objTubePolyline.Left = objTubePolyline.Left + 10
```

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-113 属性

属性	RT Professional	RT Advanced	Panel RT	说明
ActualPointIndex (页 603)	RW	-	-	确定所选对象的当前隅角点的数量。
ActualPointLeft (页 604)	RW	-	-	指定当前角点相对于画面原点的 X 坐标。
ActualPointTop (页 605)	RW	-	-	指定当前角点相对于画面原点的 Y 坐标。
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定操作权限。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Color (页 723)	RW	-	-	指定线颜色。
CornerStyle (页 764)	RW	-	-	指定角形状。
DeviceStyle	-	-	-	-
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。
Height (页 863)	RW	-	-	指定高度。
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
LineWidth (页 943)	RW	-	-	指定线宽。
Location	-	-	-	-
Name	-	-	-	-

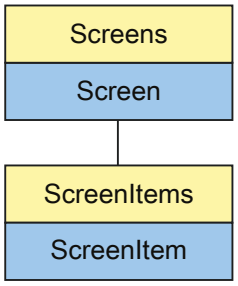
属性	RT Professional	RT Advanced	Panel RT	说明
Points	-	-	-	-
PointsCount (页 1034)	RW	-	-	指定折线或多边形角点的个数。
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (页 1278)	RW	-	-	指定工具提示文本。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-114 方法

方法	RT Professional	RT Advanced	Panel RT	说明
Activate (页 1470)	支持	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	支持	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	支持	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

TubeTeeObject

描述



表示“Tee”对象。TubeTeeObject 是 ScreenItems 列表的元素。

VBS 的类型标识符

HMITubeTeeObject

应用

在下面的示例中，名称为“TubeTeeObject1”的对象向右移动 10 个像素：

```
'VBS21
Dim objTubeTeeObject
Set objTubeTeeObject = ScreenItems("TubeTeeObject1")
objTubeTeeObject.Left = objTubeTeeObject.Left + 10
```

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-115 属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Authorization (页 626)	RW	-	-	指定操作权限。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Color (页 723)	RW	-	-	指定线颜色。
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。
Height (页 863)	RW	-	-	指定高度。
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
LineWidth (页 943)	RW	-	-	指定线宽。
Location	-	-	-	-
Name	-	-	-	-
RotationAngle (页 1048)	RW	-	-	以度为单位指定旋转角度。
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
ToolTipText (页 1278)	RW	-	-	指定工具提示文本。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。

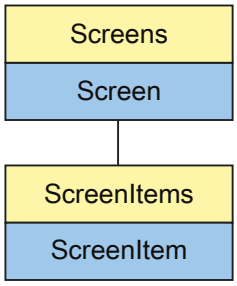
属性	RT Professional	RT Advanced	面板 RT	描述
Transparency (页 1284)	RW	-	-	指定对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	RW	-	-	指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。
UseDesignShadowSettings (页 1355)	RW	-	-	指定显示的对象是否带有全局阴影。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-116 方法

方法	RT Professional	RT Advanced	面板 RT	描述
Activate (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

用户归档控件

说明



显示“配方视图”对象 UserArchiveControl 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIUserArchiveControl

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-117 属性

属性	RT Professional	RT Advanced	面板 RT	说明
AllTagTypesAllowed	-	-	-	-
ApplyProjectSettingsForDesignMode	-	-	-	-
ArchiveName (页 622)	RW	-	-	设置日志名称。
ArchiveType (页 623)	RW	-	-	设置日志类型。
AutoCompleteColumns (页 629)	RW	-	-	指定在控件比组态的列宽时是否显示空列。
AutoCompleteRows (页 629)	RW	-	-	指定在控件比组态的行数长时是否显示空行。
AutoSelectionColors (页 632)	RW	-	-	指定是否将系统定义的颜色用作单元格和行的选择颜色。
AutoSelectionRectColor (页 632)	RW	-	-	指定是否使用系统定义的颜色显示选择框架。
BackColor (页 637)	RW	-	-	指定背景色。
Blocks	-	-	-	-
BorderColor (页 678)	RW	-	-	指定线颜色。
BorderWidth (页 697)	RW	-	-	指定线宽。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Caption (页 708)	RW	-	-	指定标题中将要显示的文本。
CellCut	-	-	-	-
CellSpaceBottom (页 714)	RW	-	-	指定表格单元格的下边距。

属性	RT Professional	RT Advanced	面板 RT	说明
CellSpaceLeft (页 715)	RW	-	-	指定表格单元格中采用的左缩进。
CellSpaceRight (页 716)	RW	-	-	指定表格单元格中采用的右缩进。
CellSpaceTop (页 717)	RW	-	-	指定表格单元格的上边距。
Closeable (页 722)	RW	-	-	指定是否可在运行系统中关闭对象。
ColumnAlias (页 726)	RW	-	-	指定使用“ColumnIndex”引用的列的显示名称。
ColumnAlignment (页 727)	RW	-	-	指定使用“ColumnIndex”引用的列内容的对齐方式。
ColumnAutoPrecisions (页 728)	RW	-	-	指定是否自动确定使用“ColumnIndex”引用的列的小数位数。
ColumnCaption (页 728)	RW	-	-	指定当前列的标题。
ColumnCount (页 729)	RW	-	-	指定所组态趋势的数目。
ColumnDateFormat (页 729)	RW	-	-	指定日期格式。
ColumnDMVarName (页 730)	RW	-	-	指定变量的名称。
ColumnExponentialFormat (页 731)	RW	-	-	指定是否在指数计数法中显示使用“ColumnIndex”引用的列的值。
ColumnFlagNotNull (页 731)	RW	-	-	指定分配给使用“ColumnIndex”引用的列的用户归档字段是否必须有一个值。
ColumnFlagUnique (页 732)	RW	-	-	指定分配给使用“ColumnIndex”引用的列的用户归档字段是否必须有唯一值。
ColumnHideText (页 733)	RW	-	-	指定是否隐藏使用“ColumnIndex”引用的列的文本。
ColumnHideTitleText (页 733)	RW	-	-	指定是否隐藏使用“ColumnIndex”引用的列的标题。

属性	RT Professional	RT Advanced	面板 RT	说明
ColumnIndex (页 734)	RW	-	-	引用一个列。
ColumnLeadingZeros (页 735)	RW	-	-	指定使用"ColumnIndex"引用的列的值所显示的前导零数量。
ColumnLength (页 735)	RW	-	-	指定使用"ColumnIndex"引用的列中所显示的字符数量。
ColumnMaxValue (页 736)	RW	-	-	指定使用"ColumnIndex"引用的列的用户归档中所定义的最大值。
ColumnMinValue (页 736)	RW	-	-	指定使用"ColumnIndex"引用的列的用户归档中所定义的最小值。
ColumnName (页 737)	RW	-	-	指定使用"ColumnIndex"引用的列的名称。
ColumnPosition (页 738)	RW	-	-	指定使用"ColumnIndex"引用的列的位置。
ColumnPrecisions (页 738)	RW	-	-	指定使用"ColumnIndex"引用的列中的小数位数。
ColumnReadAccess (页 739)	RW	-	-	指定使用"ColumnIndex"引用的列的用户归档中所定义的读访问权限。
ColumnReadonly (页 740)	RW	-	-	指定是否只可读取使用"ColumnIndex"引用的列的值。
ColumnRepos (页 741)	RW	-	-	对于多个列，指定使用"ColumnIndex"引用的列的位置。
ColumnResize (页 742)	RW	-	-	指定是否可以更改列宽。
Columns	-	-	-	-
ColumnScrollbar (页 743)	RW	-	-	指定将显示水平滚动条的时间。
ColumnShowDate (页 744)	RW	-	-	指定使用"ColumnIndex"引用的列中将显示日期。
ColumnShowIcon (页 745)	RW	-	-	指定使用"ColumnIndex"引用的列中将显示符号。
ColumnShowTitleIcon (页 745)	RW	-	-	指定使用"ColumnIndex"引用的列中将显示符号。

属性	RT Professional	RT Advanced	面板 RT	说明
ColumnSort (页 746)	RW	-	-	指定使用“ColumnIndex”引用的列的排序类型。
ColumnSortIndex (页 747)	RW	-	-	指定使用“ColumnIndex”引用的列的排序顺序，其中列以一个接一个的方式排序。
ColumnStartValue (页 748)	RW	-	-	指定使用“ColumnIndex”引用的列的用户归档中所定义的起始值。
ColumnStringLength (页 748)	RW	-	-	指定使用“ColumnIndex”引用的列的用户归档中所定义的字符串长度。
ColumnTimeFormat (页 753)	RW	-	-	指定使用“ColumnIndex”引用的列的时间格式。
ColumnTitleAlignment (页 754)	RW	-	-	指定使用“ColumnIndex”引用的列标题的对齐方式。
ColumnTitles (页 755)	RW	-	-	指定是否显示列标题。
ColumnType (页 756)	RW	-	-	指定使用“ColumnIndex”引用的列的用户归档中所定义的数据类型。
ColumnVisible (页 757)	RW	-	-	指定是否在对象中显示使用“ColumnIndex”引用的列。
ColumnWriteAccess (页 758)	RW	-	-	指定使用“ColumnIndex”引用的列的用户归档中所定义的写访问权限。
ControlDesignMode (页 762)	RW	-	-	指定控制设计。
DataProviderGuid (页 773)	R	-	-	确定控件的 GUID。
DataSource	-	-	-	-
DefaultFilterEom	-	-	-	-
Enabled	-	-	-	-
EnableDelete (页 797)	RW	-	-	指定是否可在运行系统中删除数据。
EnableEdit (页 797)	RW	-	-	指定是否可以在运行系统中编辑显示的数据。

属性	RT Professional	RT Advanced	面板 RT	说明
EnableInsert (页 798)	RW	-	-	指定是否可在运行系统中插入数据。
ExportDelimiter	-	-	-	-
ExportDirectoryChangeable (页 808)	RW	-	-	指定是否可以在运行系统中更改数据导出目录。
ExportDirectoryname (页 809)	RW	-	-	指定导出的运行系统数据的目标目录。
ExportFileExtension (页 810)	RW	-	-	指定导出文件的文件扩展名。
ExportFilename (页 811)	RW	-	-	指定导出的运行系统数据的目标文件名称。
ExportFilenameChangeable (页 812)	RW	-	-	指定是否可以在运行系统中更改导出文件名。
ExportFormat	-	-	-	-
ExportFormatGuid (页 813)	RW	-	-	指定 ID 编号和导出数据源的分配。
ExportFormatName (页 814)	RW	-	-	指定导出文件格式。
ExportParameters (页 815)	RW	-	-	通过属性对话框指定所选格式的参数。
ExportSelection (页 816)	RW	-	-	指定导出控件的哪些运行系统数据。
ExportShowDialog (页 817)	RW	-	-	指定是否在运行系统中显示数据导出对话框。
FillPattern	-	-	-	-
FillPatternColor	-	-	-	-
Filter	-	-	-	-
FilterSQL (页 826)	RW	-	-	针对过滤条件设置 SQL 语句。
Font (页 846)	RW	-	-	指定字体。

属性	RT Professional	RT Advanced	面板 RT	说明
GridLineColor (页 860)	RW	-	-	指定网格线颜色。
GridLineWidth (页 862)	RW	-	-	指定分隔线的宽度（以像素为单位）。
Height (页 863)	RW	-	-	指定高度。
HorizontalGridLines (页 883)	RW	-	-	指定是否显示水平分隔线。
IconSpace (页 886)	RW	-	-	指定表格单元格中图标和文本的间距。
Layer	-	-	-	-
Left (页 920)	RW	-	-	指定 X 坐标的值。
LineBackgroundColor	-	-	-	-
LineColor (页 939)	RW	-	-	指定窗口分隔线的颜色。
LineStyle	-	-	-	-
LineWidth (页 943)	RW	-	-	指定线宽。
Location	-	-	-	-
Moveable (页 983)	RW	-	-	指定关于运行系统中是否可以移动对象的信息。
Name	-	-	-	-
Object	-	-	-	-
OcxGuid	-	-	-	-
OcxState	-	-	-	-
OcxStateForEs2Rt	-	-	-	-
PrintJob (页 1036)	RW	-	-	指定一个在“报表”编辑器中创建的打印作业。
RowScrollbar (页 1052)	RW	-	-	指定垂直滚动条将要显示的时间。

属性	RT Professional	RT Advanced	面板 RT	说明
RowTitleAlignment (页 1053)	RW	-	-	指定行标题对齐的类型。
RowTitles (页 1054)	RW	-	-	指定是否显示已编号的列标题。
RTPersistence (页 1054)	RW	-	-	指定在画面更改后是否保留在线组态。
RTPersistenceAuthorization (页 1055)	RW	-	-	指定运行系统中在线组态所需的权限。
RTPersistenceType (页 1057)	RW	-	-	指定如何保留 WinCC 的在线组态。
SelectArchiveName (页 1074)	RW	-	-	指定配方视图的数据源的选择对话框在运行系统中初次显示。
SelectedCellColor (页 1075)	RW	-	-	指定所选单元格的背景色。
SelectedCellForeColor (页 1076)	RW	-	-	指定所选单元格的字体颜色。
SelectedID (页 1077)	RW	-	-	指定在配方视图中所选的数据记录的 ID。
SelectedRowColor (页 1079)	RW	-	-	指定所选行的背景色。
SelectedRowForeColor (页 1080)	RW	-	-	指定所选行的字体颜色。
SelectedTitleColor (页 1081)	RW	-	-	指定所选表格标题的背景色。
SelectedTitleForeColor (页 1082)	RW	-	-	指定所选表格标题的字体颜色。
SelectionColoring (页 1085)	RW	-	-	指定是否对单元格或行使用选择颜色。
SelectionRect (页 1087)	RW	-	-	指定是否对所选单元格或行使用选择框架。
SelectionRectColor (页 1088)	RW	-	-	指定报警窗口中选择矩形的颜色。

属性	RT Professional	RT Advanced	面板 RT	说明
SelectionRectWidth (页 1089)	RW	-	-	如果 SelectionType 等于“1”，则指定报警窗口中选择矩形的线宽。
SelectionType (页 1090)	RW	-	-	指定可标记的行数。
ShowSortButton (页 1117)	RW	-	-	指定是否在垂直滚动条上显示排序按钮。
ShowSortIcon (页 1118)	RW	-	-	指定是否显示排序图标。
ShowSortIndex (页 1118)	RW	-	-	指定是否显示排序索引。
ShowTitle (页 1124)	RW	-	-	为对象指定窗口边框和窗口标题的样式。
Size	-	-	-	-
Sizeable (页 1129)	RW	-	-	指定可以在运行系统中更改对象的大小。
SortSequence (页 1132)	RW	-	-	如果操作员在运行系统中单击列标题，指定排序顺序会如何改变。
StatusbarBackColor (页 1138)	RW	-	-	指定状态栏的背景色。
StatusbarElementAdd (页 1139)	RW	-	-	创建一个新的用户自定义状态栏元素。
StatusbarElementAutoSize (页 1140)	RW	-	-	指定是否自动设置引用“StatusbarElementIndex”的状态栏元素的宽度。
StatusbarElementCount (页 1141)	RW	-	-	指定已组态的状态栏元素的数量。
StatusbarElementIconId (页 1142)	RW	-	-	使用其图标 ID 来引用状态栏元素。
StatusbarElementID (页 1143)	RW	-	-	使用其元素 ID 来引用状态栏元素。
StatusbarElementIndex (页 1144)	RW	-	-	引用状态栏元素。

属性	RT Professional	RT Advanced	面板 RT	说明
StatusbarElementName (页 1145)	RW	-	-	指定使用"StatusbarElementIndex"引用的状态栏元素的名称。
StatusbarElementRemove (页 1146)	RW	-	-	删除引用其名称的用户自定义状态栏元素。
StatusbarElementRename (页 1147)	RW	-	-	指定使用"StatusbarElementIndex"引用的用户自定义的状态栏元素的新名称。
StatusbarElementRepos (页 1148)	RW	-	-	指定对象状态栏中使用"StatusbarElementIndex"引用的元素位置。
StatusbarElements	-	-	-	-
StatusbarElementText (页 1149)	RW	-	-	指定使用"StatusbarElementIndex"引用的用户自定义的状态栏元素的文本。
StatusbarElementTooltipText (页 1150)	RW	-	-	指定使用"StatusbarElementIndex"引用的用户自定义的状态栏元素的工具提示文本。
StatusbarElementUserDefined (页 1151)	RW	-	-	指定组态工程师是否已添加使用"StatusbarElementIndex"引用的状态栏元素作为新的用户自定义元素。
StatusbarElementVisible (页 1152)	RW	-	-	指定是否在对象中显示使用"StatusbarElementIndex"引用的状态栏元素。
StatusbarElementWidth (页 1153)	RW	-	-	指定使用"StatusbarElementIndex"引用的状态栏元素的宽度（以像素为单位）。
StatusbarFont (页 1154)	RW	-	-	指定状态栏中文本的字体。
StatusbarFontColor (页 1155)	RW	-	-	指定状态栏中文本的颜色。
StatusbarShowTooltips (页 1156)	RW	-	-	指定是否在运行系统中显示状态栏元素的工具提示。

属性	RT Professional	RT Advanced	面板 RT	说明
StatusbarText (页 1157)	RW	-	-	指定状态栏中的默认文本。
StatusbarUseBackColor (页 1158)	RW	-	-	指定是否显示状态栏的背景色。
StatusbarVisible (页 1159)	RW	-	-	指定是否显示控件的状态栏。
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableColor (页 1171)	RW	-	-	指定对象中表格行的背景色。
TableColor2 (页 1172)	RW	-	-	指定对象中表格行的第二种背景色。
TableForeColor (页 1173)	RW	-	-	指定对象表格单元格中采用的文本颜色。
TableForeColor2 (页 1174)	RW	-	-	指定对象表格单元格中采用的第二种文本颜色。
TimeBase (页 1211)	RW	-	-	指定显示时间值的时区。
TitleColor (页 1236)	RW	-	-	指定表格标题的背景色。
TitleCut (页 1237)	RW	-	-	指定列宽过窄时是否缩减标题栏中的字段内容。
TitleDarkShadowColor (页 1238)	RW	-	-	为对象列表中的列和行标题指定 3D 底纹的暗边颜色。
TitleForeColor (页 1239)	RW	-	-	为对象的表格列和行标题指定文本颜色。
TitleGridLineColor (页 1240)	RW	-	-	指定表格标题栏中分隔线的颜色。
TitleLightShadowColor (页 1241)	RW	-	-	为对象的表格列和行标题指定 3D 底纹的亮边颜色。
TitleSort (页 1242)	RW	-	-	指定如何触发按列标题排序。

属性	RT Professional	RT Advanced	面板 RT	说明
TitleStyle (页 1243)	RW	-	-	指定是否使用列标题文本的底纹颜色。
ToolbarAlignment (页 1252)	RW	-	-	指定工具栏的位置。
ToolbarBackColor (页 1253)	RW	-	-	指定工具栏的背景色。
ToolbarButtonActive (页 1254)	RW	-	-	指定是否在运行系统中激活按钮的相应功能。
ToolbarButtonAdd (页 1255)	RW	-	-	在对象的工具栏中创建一个新的用户自定义按钮。
ToolbarButtonAuthorization (页 1256)	RW	-	-	指定所选按键功能的权限。
ToolbarButtonBeginGroup (页 1257)	RW	-	-	在工具栏上，为所选按键功能插入前导分隔符（垂直线）。
ToolbarButtonClick (页 1258)	RW	-	-	单击工具栏按钮。
ToolbarButtonCount (页 1259)	RW	-	-	指定工具栏中的已组态按钮数目。
ToolbarButtonEnabled (页 1260)	RW	-	-	指定是否可以操作用户自定义工具栏按钮。
ToolbarButtonHotKey (页 1261)	RW	-	-	指定所选对象按钮的热键。
ToolbarButtonID (页 1262)	RW	-	-	使用其 ID 引用按钮。
ToolbarButtonIndex (页 1263)	RW	-	-	引用按钮。
ToolbarButtonLocked (页 1264)	RW	-	-	指定是否显示使用“ToolbarButtonIndex”引用的用户自定义按钮的锁定、按下状态。
ToolbarButtonName (页 1265)	RW	-	-	指定使用“ToolbarButtonIndex”引用的用户自定义按钮的名称。

属性	RT Professional	RT Advanced	面板 RT	说明
ToolbarButtonRemove (页 1266)	RW	-	-	移除一个引用了其名称的用户自定义按钮。
ToolbarButtonRename (页 1267)	RW	-	-	指定使用"ToolbarButtonIndex"引用的用户自定义按钮的新名称。
ToolbarButtonRepos (页 1268)	RW	-	-	在对象的工具栏中指定使用"ToolbarButtonIndex"引用的按钮位置。
ToolbarButtons	-	-	-	-
ToolbarButtonTooltipText (页 1270)	RW	-	-	指定工具栏中用户自定义按钮的工具提示文本。
ToolbarButtonUserDefined (页 1271)	RW	-	-	指定项目工程师是否已将此工具栏按钮添加为新的用户自定义按钮。
ToolbarButtonVisible (页 1272)	RW	-	-	指定是否在工具栏中显示使用"ToolbarButtonIndex"引用的按钮。
ToolbarShowTooltips (页 1273)	RW	-	-	指定是否在运行系统中显示按键功能的工具提示。
ToolbarUseBackColor (页 1275)	RW	-	-	指定是否显示工具栏的背景色。
ToolbarUseHotKeys (页 1276)	RW	-	-	指定是否激活工具栏中按钮的热键。
ToolbarVisible (页 1277)	RW	-	-	指定是否显示控件的工具栏。
Top (页 1279)	RW	-	-	指定 Y 坐标的值。
UseSelectedTitleColor (页 1362)	RW	-	-	指定是否将选择颜色用于选定表格单元格的标题。
UseTableColor2 (页 1364)	RW	-	-	指定是否用第二种行颜色来表现表格。
VerticalGridLines (页 1414)	RW	-	-	指定是否显示垂直分隔线。
Visible (页 1418)	RW	-	-	指定是否显示所选对象。
Width (页 1432)	RW	-	-	指定对象的宽度（以像素为单位）。

表格 1-118 方法

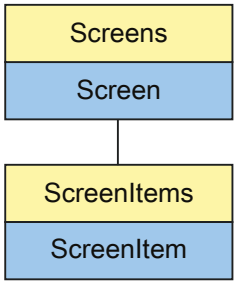
方法	RT Professional	RT Advanced	面板 RT	说明
Activate (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，使用 VBScript 动态激活触发器和属性的指定周期。
CopyRows (页 1479)	√	-	-	执行控件的“复制行”键功能。
CutRows (页 1481)	√	-	-	执行配方视图的“剪切行”键功能。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。
DeleteRows (页 1483)	√	-	-	执行配方视图的“删除行”键功能。
导出 (页 1485)	√	-	-	执行控件的“导出日志”(Export log) 或“导出数据”(Export data) 按键功能。
GetColumn (页 1486)	√	-	-	以“ICCAxUAColumn”类型的形式返回配方视图中具有指定名称或索引的列对象。
GetColumnCollection (页 1487)	√	-	-	以“ICCAxCollection”类型的形式返回配方视图中所有列对象的列表。
GetRow (页 1498)	√	-	-	以“ICCAxDataRow”类型的形式返回基于表格的控件中的行编号所指定的行对象。
GetRowCollection (页 1499)	√	-	-	以“ICCAxDataRowCollection”类型的形式返回基于表格的控件的所有行对象的列表。
GetSelectedRow (页 1507)	√	-	-	以“ICCAxDataRow”类型的形式返回基于表格的控件的所选行对象。
GetSelectedRows (页 1508)	√	-	-	以“ICCAxDataRow”类型的形式返回用于多项选择的基于表格的控件的所选行对象。
GetStatusbarElement (页 1514)	√	-	-	以“ICCAxStatusbarElement”类型的形式返回按名称或索引指定的控件状态栏元素。
GetStatusbarElementCollection (页 1516)	√	-	-	以类型“ICCAxCollection”形式返回控件的所有状态栏元素的列表。
GetToolbarButton (页 1523)	√	-	-	以“ICCAxToolbarButton”类型的形式返回控件工具栏上按名称或索引指定的按钮功能。

方法	RT Professional	RT Advanced	面板 RT	说明
GetToolbarButtonCollection (页 1525)	√	-	-	以"ICCAxCollection"类型的形式返回控件工具栏的所有按钮功能的列表。
MoveToFirst (页 1547)	√	-	-	执行控件的"第一行"(First line) 按钮功能。
MoveToLast (页 1549)	√	-	-	执行控件的"最后一条数据记录"(Last data record) 按钮功能。
MoveToNext (页 1551)	√	-	-	执行控件的"下一条数据记录"(Next data record) 按钮功能。
MoveToPrevious (页 1552)	√	-	-	执行控件的"前一条数据记录"(Previous data record) 按钮功能。
PasteRows (页 1555)	√	-	-	执行配方视图的"插入行"(Insert rows) 按钮功能。
打印 (页 1557)	√	-	-	执行控件的"打印"(Print) 按钮功能。
ReadTags (页 1563)	√	-	-	执行配方视图的"读取变量"(Read tags) 按钮功能。
SelectAll (页 1573)	√	-	-	在基于表格的控件中选择所有行。
SelectRow (页 1575)	√	-	-	在基于表格的控件中选择特定行。
ServerExport (页 1576)	√	-	-	执行配方视图的"导出日志"(Export log) 按钮功能。
ServerImport (页 1576)	√	-	-	执行配方视图的"导入日志"(Import log) 按钮功能。
ShowHelp (页 1580)	√	-	-	执行控件的"帮助"(Help) 按钮功能。
ShowPropertyDialog (页 1585)	√	-	-	执行控件的"组态对话框"(Configuration dialog) 按钮功能。
ShowSelectArchive (页 1586)	√	-	-	执行配方视图的"选择数据连接"(Select data connection) 按钮功能。
ShowSelection (页 1586)	√	-	-	执行配方视图的"选择对话框"(Selection dialog) 按钮功能。
ShowSelectTimeBase (页 1587)	√	-	-	执行配方视图的"时基对话框"(Timebase dialog) 按钮功能。
ShowSort (页 1588)	√	-	-	执行配方视图的"排序对话框"(Sorting dialog) 按钮功能。

方法	RT Professional	RT Advanced	面板 RT	说明
UnselectAll (页 1595)	√	-	-	在基于表格的控件的单元格中移除所有选择内容。
UnselectRow (页 1595)	√	-	-	在基于表格的控件的特定单元格中移除选择内容。
WriteTags (页 1599)	√	-	-	执行配方视图的“写入变量”(Write tags) 按钮功能。

UserView

说明



表示“UserView”对象。UserView 对象是 ScreenItems 列表的元素。

说明

无法通过用户自定义函数实现“Simple UserView”对象的动态化。

VBS 的类型标识符

HMIUserView

缩写	运行系统中的访问权限
R	读取
RW	读和写
-	无访问权

表格 1-119 属性

属性	RT Professional	RT Advanced	Panel RT	说明
AllTagTypesAllowed	-	-	-	-
AnimationIgnore	-	-	-	-
Appearance	-	-	-	-
Authorization (页 626)	-	RW	RW	指定所选对象在运行系统中的操作权限。
BackColor (页 637)	-	RW	RW	指定所选对象的背景色。
BorderBackColor	-	-	-	-
BorderColor	-	-	-	-
BorderWidth	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Columns	-	-	-	-
ColumnsMoveable	-	-	-	-
ColumnTextGroup	-	-	-	-
ColumnTextLogTime	-	-	-	-
ColumnTextPassword	-	-	-	-
ColumnTextUser	-	-	-	-
CornerRadius	-	-	-	-
CountVisibleItems	-	-	-	-

属性	RT Professional	RT Advanced	Panel RT	说明
DeviceStyle	-	-	-	-
EdgeStyle	-	-	-	-
Enabled (页 792)	RW	RW	RW	指定是否可以在运行系统中操作所选对象。
Es2rtButtonPositions	-	-	-	-
FitToSize	-	-	-	-
Flashing	-	-	-	-
Height (页 863)	RW	R	R	指定所选对象的高度。
IsMinPasswordValueSet	-	-	-	-
IsRunningUnderCE	-	-	-	-
Layer (页 913)	RW	RW	RW	在包含对象的画面中指定图层。
Left (页 920)	RW	RW	RW	指定所选对象的 X 坐标的值。
ListAreaHeight	-	-	-	-
ListAreaWidth	-	-	-	-
Location	-	-	-	-
MinPasswordValue	-	-	-	-
Name	-	-	-	-
PaddingBottom	-	-	-	-
PaddingLeft	-	-	-	-
PaddingRight	-	-	-	-
PaddingTop	-	-	-	-
RTPersistence	-	-	-	-
RTPersistenceAuthorization	-	-	-	-
RTPersistenceType	-	-	-	-
SelectionBackColor (页 1084)	-	RW	RW	指定所选单元格的背景色。
SelectionForeColor (页 1086)	-	RW	RW	指定所选单元格的前景色。
ShowColumnHeaders	-	-	-	-

属性	RT Professional	RT Advanced	Panel RT	说明
ShowTableGridlines	-	-	-	-
Size	-	-	-	-
StyleItem	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
TableBackColor (页 1170)	-	RW	RW	指定所选对象的表格单元格的背景色。
TableEvenRowBackColor	-	-	-	-
TableFont	-	-	-	-
TableForeColor (页 1173)	-	RW	RW	指定对象表格单元格中采用的文本颜色。
TableGridLineColor (页 1175)	-	RW	RW	确定指定对象的表中的网格线颜色。
TableHeaderBackColor (页 1176)	-	RW	RW	指定选定对象表格标题的背景色。
TableHeaderBackFillStyle	-	-	-	-
TableHeaderBorderBackColor	-	-	-	-
TableHeaderBorderColor	-	-	-	-
TableHeaderBorderWidth	-	-	-	-
TableHeaderCornerRadius	-	-	-	-
TableHeaderEdgeStyle	-	-	-	-
TableHeaderFirstGradientColor	-	-	-	-
TableHeaderFirstGradientOffset	-	-	-	-

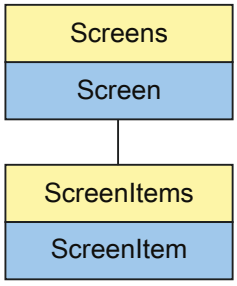
属性	RT Professional	RT Advanced	Panel RT	说明
TableHeaderFont	-	-	-	-
TableHeaderForeColor (页 1179)	-	RW	RW	指定选定对象表格标题的文本颜色。
TableHeaderMiddleGradientColor	-	-	-	-
TableHeaderPaddingBottom	-	-	-	-
TableHeaderPaddingLeft	-	-	-	-
TableHeaderPaddingRight	-	-	-	-
TableHeaderPaddingTop	-	-	-	-
TableHeaderSecondGradientColor	-	-	-	-
TableHeaderSecondGradientOffset	-	-	-	-
Top (页 1279)	R	RW	RW	指定所选对象的 Y 坐标的值。
UseDesignColorSchema	-	-	-	-
UseTableHeaderFirstGradient	-	-	-	-
UseTableHeaderSecondGradient	-	-	-	-
VerticalScrolling	-	-	-	-
ViewType	-	-	-	-
ViewTypeForSaveStream	-	-	-	-
Visible (页 1418)	R	RW	RW	指定是否显示所选对象。
Width (页 1432)	RW	R	R	指定对象的宽度（以像素为单位）。

表格 1-120 方法

方法	RT Professional	RT Advanced	Panel RT	说明
Activate (页 1470)	支持	支持	支持	启用永久区域或根画面。
ActivateDynamic (页 1473)	支持	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	支持	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

WindowSlider

说明



表示“Window slider”对象。WindowSlider 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIWindowSlider

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-121 属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
AskOperationMotive (页 624)	RW	-	-	指定是否记录操作此对象的原因。
Authorization (页 626)	RW	-	-	指定操作权限。
BackColor (页 637)	RW	-	-	指定背景色。
BackColorBottom (页 641)	RW	-	-	指定对象下半部/右侧的颜色。
BackColorTop (页 642)	RW	-	-	指定对象上半部/左侧的颜色。
BackFillStyle (页 643)	RW	-	-	指定填充图案
BackFlashingColorOff (页 645)	RW	-	-	指定闪烁状态“关闭”(Off) 的背景色。
BackFlashingColorOn (页 646)	RW	-	-	指定闪烁状态“打开”(On) 的背景色。
BackFlashingEnabled (页 648)	RW	-	-	指定是否在运行系统中闪烁背景。
BackFlashingRate (页 649)	RW	-	-	指定背景的闪烁频率。
BorderBackColor (页 674)	RW	-	-	指定间断边框线的背景色。

属性	RT Professional	RT Advanced	面板 RT	描述
BorderColor (页 678)	RW	-	-	指定线颜色。
BorderFlashingColorOff (页 683)	RW	-	-	指定闪烁状态“关闭”时的边框线颜色。
BorderFlashingColorOn (页 685)	RW	-	-	指定闪烁状态“开启”时的边框线颜色。
BorderFlashingEnabled	-	-	-	-
BorderFlashingRate (页 689)	RW	-	-	指定边框线的闪烁速率。
BorderStyle (页 695)	RW	-	-	指定边框线的闪烁类型。
BorderWidth (页 697)	RW	-	-	指定线宽。
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
CornerStyle (页 764)	RW	-	-	指定边角的形状。
DrawInsideFrame (页 787)	RW	-	-	指定要显示的线宽大于 1 边框线是在边框内描绘，还是与边框对称。
EdgeStyle (页 788)	RW	-	-	指定线样式。
Enabled (页 792)	RW	-	-	指定是否可以在运行系统中操作所选对象。
FillingDirection (页 825)	RW	-	-	指定填充方向。
FillPatternColor (页 821)	RW	-	-	指定填充样式的颜色。
Flashing	-	-	-	-
Height (页 863)	RW	-	-	指定高度。
HighLimitColor (页 870)	RW	-	-	指定滚动条中顶部或右侧滚动按钮的颜色。
JumpToLimitsAfterMouseClick (页 906)	RW	-	-	指定是否对相关终值设置滑块。

属性	RT Professional	RT Advanced	面板 RT	描述
Layer (页 913)	RW	-	-	在包含对象的画面中指定图层。
Left (页 920)	RW	-	-	指定 X 坐标的值。
LineEndShapeStyle (页 940)	RW	-	-	指定线端的形状。
Location	-	-	-	-
LogOperation (页 948)	RW	-	-	指定是否在操作对象后将报警输出到报警系统。
LowLimitColor (页 950)	RW	-	-	指定滚动条中底部或左侧滚动按钮的颜色。
MarginToBorder (页 952)	RW	-	-	指定 3D 边框的宽度（以像素为单位）。
MaximumValue (页 953)	RW	-	-	指定所选对象的刻度范围的最大值。
MinimumValue (页 979)	RW	-	-	指定所选对象的刻度范围的最小值。
Name	-	-	-	-
OperationSteps (页 1000)	R	-	-	返回单击一次鼠标滚动条滑块移动的步数。
ProcessValue (页 1037)	R	-	-	返回要显示的数值的默认值。
RelativeFillLevel (页 1046)	R	-	-	返回对象的填充百分比。
ShowBadTagState (页 1099)	R	-	-	返回关于出现不良质量代码或变量状态时对象是否呈灰显的信息。
ShowFillLevel (页 1108)	R	-	-	返回关于特定的对象是否已填充的信息。
Size	-	-	-	-
StyleSettings (页 1164)	R	-	-	返回对象的显示样式。
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-

属性	RT Professional	RT Advanced	面板 RT	描述
TextOrientation (页 1189)	R	-	-	返回文本方向。
ThumbBackColor (页 1192)	R	-	-	返回滑块的背景色。
ToolTipText (页 1278)	R	-	-	返回工具提示文本。
Top (页 1279)	R	-	-	返回 Y 坐标的值。
Transparency (页 1284)	R	-	-	返回对象透明度（百分数形式）。
UseDesignColorSchema (页 1353)	R	-	-	返回关于在当前设计的全局颜色方案中定义的颜色是否用于此对象的信息。
UseDesignShadowSettings (页 1355)	R	-	-	返回关于显示的对象是否带有全局阴影的信息。
Visible (页 1418)	R	-	-	返回关于是否显示特定对象的信息。
Width (页 1432)	R	-	-	返回对象的宽度（以像素为单位）。
WindowsStyle (页 1440)	R	-	-	返回关于是否以普通 Windows 样式显示对象的信息。

表格 1-122 方法

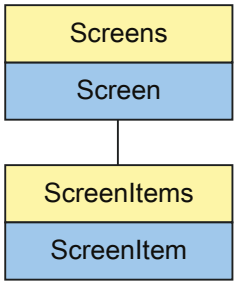
方法	RT Professional	RT Advanced	面板 RT	描述
Activate (页 1470)	√	-	-	启用永久区域或根画面。
ActivateDynamic (页 1473)	√	-	-	在运行期间，动态激活触发器和属性的指定周期。
DeactivateDynamic (页 1481)	√	-	-	在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

参见

BorderFlashingEnabled (页 687)

WLanQualityView

描述



表示“WLAN reception”对象。WLanQualityView 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-123　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-

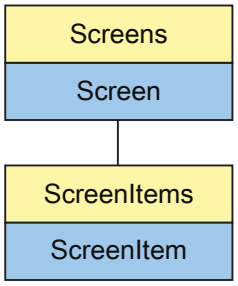
属性	RT Professional	RT Advanced	面板 RT	描述
CanBeGrouped	-	-	-	-
Height	-	R	R	指定高度。
Layer (页 913)	-	-	RW	在包含对象的画面中指定图层。
Left (页 920)	-	-	RW	指定 X 坐标的值。
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (页 1279)	-	-	RW	指定 Y 坐标的值。
Visible (页 1418)	-	-	RW	指定是否显示所选对象。
Width	-	R	R	指定宽度。

表格 1-124 方法

方法	RT Professional	RT Advanced	面板 RT	描述
Activate (页 1470)	-	-	√	启用永久区域或根画面。

ZoneLabelView

描述



表示“Zone name”对象。ZoneLabelView 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

HMIZoneLabelView

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-125　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-

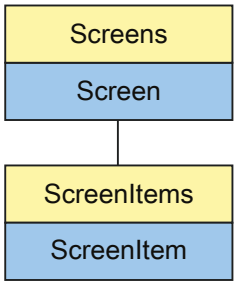
属性	RT Professional	RT Advanced	面板 RT	描述
CanBeGrouped	-	-	-	-
Font	-	-	-	-
Height	-	R	R	指定高度。
Layer (页 913)	-	-	RW	在包含对象的画面中指定图层。
Left (页 920)	-	-	RW	指定 X 坐标的值。
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (页 1279)	-	-	RW	指定 Y 坐标的值。
Visible (页 1418)	-	-	RW	指定是否显示所选对象。
Width	-	R	R	指定宽度。

表格 1-126 方法

方法	RT Professional	RT Advanced	面板 RT	描述
Activate (页 1470)	-	-	√	启用永久区域或根画面。

ZoneQualityView

描述



表示“ZoneSignal”对象。ZoneQualityView 对象是 ScreenItems 列表的元素。

VBS 的类型标识符

缩写	运行系统中的访问权限
R	读
RW	读和写
-	无访问权

表格 1-127　属性

属性	RT Professional	RT Advanced	面板 RT	描述
AllTagTypesAllowed	-	-	-	-
Bounds	-	-	-	-
CanBeGrouped	-	-	-	-
Height	-	R	R	指定高度。

属性	RT Professional	RT Advanced	面板 RT	描述
Layer (页 913)	-	-	RW	在包含对象的画面中指定图层。
Left (页 920)	-	-	RW	指定 X 坐标的值。
Location	-	-	-	-
Name	-	-	-	-
Size	-	-	-	-
TabIndex	-	-	-	-
TabIndexAlpha	-	-	-	-
Top (页 1279)	-	-	RW	指定 Y 坐标的值。
Visible (页 1418)	-	-	RW	指定是否显示所选对象。
Width	-	R	R	指定宽度。

表格 1-128 方法

方法	RT Professional	RT Advanced	面板 RT	描述
Activate (页 1470)	-	-	√	启用永久区域或根画面。

1.5.5 属性

1.5.5.1 属性 A

AboveUpperLimitColor

描述

指定超出上限时显示的颜色。

运行系统中的访问权限

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**AboveUpperLimitColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- IOField

在运行系统中您没有以下格式的访问权限：

- GraphicIOField
- Switch
- SymbolLibrary
- SymbolicIOField

Color

可选项。用于指定超出上限情况颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

IOField (页 382)

GraphicIOField (页 371)

Switch (页 499)

SymbolLibrary (页 511)

SymbolicIOField (页 504)

AcceptOnExit

描述

指定当保留输入域内容时，是否自动对输入域进行确认。

运行系统中的访问权限： 读和写

语法

Object.**AcceptOnExit**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- IOField
- SymbolicIOField

BOOLEAN

可选。若保留输入域内容时自动对其进行确认，则为 TRUE。

参见

IOField (页 382)

SymbolicIOField (页 504)

AcceptOnFull

描述

指定当输入设定值时，是否保留输入域内容并自动进行确认。

运行系统中的访问权限： 读和写

语法

Object.**AcceptOnFull**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- IOField

BOOLEAN

可选。若输入设定值时，保留输入域内容并自动进行确认，则为 TRUE。

参见

IOField (页 382)

AccessPath

描述

返回画面的存储路径。

运行中可进行的访问：读取

语法

Object.**AccessPath**

Object

必需。 “Screen”对象。

示例

在以下示例中，发出“ScreenWindow1”画面的路径：

```
'VBS67
Dim objScreen
Set objScreen = HMIRuntime.Screens("ScreenWindow1")
MsgBox objScreen.AccessPath
```

参见

Screen (页 231)

ActiveProject

描述

返回指定的项目。

运行系统中可进行的访问：读

语法

Object.**ActiveProject**

Object

必选。"HMIRuntime"对象。

参见

HMIRuntime（页 224）

Project (页 230)

ActiveScreen

描述

返回"Screen"类型的对象，该对象显示当前具有焦点的画面。

说明

如果在用户自定义函数中查询"ActiveScreen"属性，则可能由于屏幕保护程序的原因，属性不返回有效的"screen"对象，而返回"Nothing"。将发出一条系统消息。

运行系统中的访问权限：读

语法

Object.ActiveScreen

对象

必需项。"HMIRuntime"类型的对象。

注释

返回哪一个画面取决于根画面或永久性区域是否具有焦点。

如果没有画面具有焦点，则 ActiveScreen 属性将返回 NOTHING。例如，当其它窗口具有焦点时，便是这种情况。使用语句"If not [expression] is nothing"，您可以询问是否要返回一个画面：

```
VBS example ActiveScreen
Dim objActiveScreen
Set objActiveScreen = HmiRuntime.ActiveScreen
If not objActiveScreen is nothing then
'found an active screen
HmiRuntime.Trace("There is an active screen." & vbCrLf)
Else
'found NO active screen
HmiRuntime.Trace("There is NO active screen." & vbCrLf)
End If
```

参见

HMIRuntime（页 224）

ActiveScreen

描述

返回"Screen"类型的对象，该对象显示当前具有焦点的画面。

说明

如果在某个函数中查询"ActiveScreen"属性，则可能由于 ScreenSavers 的原因，该属性不返回有效的"Screen"对象，而是返回"Nothing"，并显示系统报警。

运行系统中的访问权限：读取

语法

Object.ActiveScreen

Object

必需项。"HMIRuntime"类型的对象。

注释

返回哪一个画面取决于根画面或永久性区域是否具有焦点。

如果没有画面具有焦点，则 ActiveScreen 属性将返回 NOTHING。例如，当其它窗口具有焦点时，便是这种情况。使用指令“If not [printout] Is Nothing”可以查询是否会返回画面：

```
'VBS_Example_ActiveScreen
Dim objActiveScreen
Set objActiveScreen = HmiRuntime.ActiveScreen
If Not objActiveScreen Is Nothing Then
'found an active screen
HmiRuntime.Trace("There is an active screen." & vbCrLf)
Else
'found NO active screen
HmiRuntime.Trace("There is NO active screen." & vbCrLf)
End If
```

参见

HMIRuntime（页 224）

ActiveScreenItem

描述

引用当前具有焦距的画面对象。

适用于只有当前选择相应“Screen”对象的画面，并且输入域“Screen”对象的“ActiveScreenItem”属性为有效的“ScreenItem”对象时的情况。 例如，在所有其它情况下，如果选择“Screens”列表中的另一个画面、WinCC 中的一个独立窗口或另一个应用程序，画面中将不会提供该属性，即为该属性分配“Nothing”。

应用

“ActiveScreenItem”对象用于对在运行系统中具有焦距的对象的属性进行寻址。

ActiveScreenItem

说明

引用当前具有焦点的画面对象。

如果相应“Screen”对象的画面处于激活状态，并且具有输入域，那么，仅为“Screen”对象的“ActiveScreenItem”属性分配一个有效的“ScreenItem”对象。 在所有其它情况下，例如，在通

过“Screens”列表的另一个画面，在 WinCC 中选择独立的窗口或者另一个应用程序的情况下，该属性不会应用到这些画面，其值为“Nothing”。

应用

“ActiveScreenItem”对象用来寻址运行时具有焦点的对象属性。

ActualPointIndex

描述

指定当前角点的数量。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**ActualPointIndex**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Polygon
- Polyline
- Tubepolyline

在运行系统中您没有以下格式的访问权限：

- Line

Int32

可选项。用于指定当前角点数的值或常量。

参见

Polygon (页 444)

Polyline (页 448)

TubePolyline (页 560)

Line (页 388)

ActualPointLeft

描述

指定当前角点相对于画面原点的X坐标。画面原点位于对象的左上角。每个角都由现有角数目（“PointCount”）中的索引来标识。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**ActualPointLeft**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Polygon
- Polyline
- Tubepolyline

在运行系统中您没有以下格式的访问权限：

- Line

Int32

可选项。指定当前角点相对于画面原点的 X 坐标的值或常量。

注释

更改该值会对属性“Width”（对象宽度）和“Left”（对象位置的 X 坐标）产生影响。

参见

Polygon (页 444)

Polyline (页 448)

TubePolyline (页 560)

Line (页 388)

ActualPointTop

描述

指定当前角点相对于画面原点的 Y 坐标。画面原点位于对象的左上角。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**ActualPointTop**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Polygon
- Polyline
- Tubepolyline

在运行系统中您没有以下格式的访问权限：

- Line

Int32

可选项。指定当前角点相对于画面原点的 Y 坐标的值或常量。

注释

更改该值会对属性"Height"（对象高度）和"Top"（对象位置的 Y 坐标）产生影响。

参见

Polygon (页 444)

Polyline (页 448)

TubePolyline (页 560)

Line (页 388)

AdaptBorder

说明

指定对象的边框是否根据文本的大小进行动态调整。

运行系统中的访问权限：读和写

语法

Object.**AdaptBorder**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Button
- CheckBox
- IOField
- OptionGroup
- SymbolicIOField

BOOLEAN

可选项。

如果对象边框将根据文本大小动态调整，则选择 TRUE。

如果对象边框不根据文本大小动态调整，则选择 FALSE。

参见

Button (页 296)

CheckBox (页 307)

IOField (页 382)

OptionGroup (页 435)

SymbolicIOField (页 504)

AdaptPicture

描述

在运行系统中无访问权限。

AdaptScreenToWindow

描述

指定画面窗口中显示的画面是否适合运行系统中画面窗口的大小。

运行系统中可进行读写访问

语法

Object.**AdaptScreenToWindow**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Screenwindow

BOOLEAN

可选项。

如果画面适合画面窗口大小，则该值为 TRUE。

如果画面不适合画面窗口大小，则该值为 FALSE。

参见

ScreenWindow (页 479)

AdaptWindowtoScreen

描述

指定画面窗口是否适合其在运行系统中显示的画面。

运行系统中可进行读写访问

语法

Object.**AdaptWindowtoScreen**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Screenwindow

BOOLEAN

可选项。

如果画面窗口大小适合画面，则该值为 TRUE。

如果画面窗口大小不适合画面，则该值为 FALSE。

参见

ScreenWindow (页 479)

Address

说明

指定在 HTML 浏览器中将打开的 Web 地址。

运行系统中的访问权限：读和写

语法

Object.**Address**[=STRING]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- HTMLBrowser

STRING

可选项。包含 Web 地址的值或常量。

参见

HTMLBrowser (页 379)

AddressEnabled

描述

在运行系统中无访问权限。

AdressPreview

描述

在运行系统中无访问权限。

AdvancedButtonPositions

说明

在运行系统中无访问权限。

AdvancedView

说明

在运行系统中无访问权限。

Alarm

描述

在运行系统中无访问权限。

AlarmAreaHeight

说明

在运行系统中无访问权限。

AlarmAreaWidth

说明

在运行系统中无访问权限。

AlarmClasses

描述

在运行系统中无访问权限。

AlarmColor

描述

在运行系统中无访问权限。

AlarmID

描述

返回 Alarm 对象的 AlarmID。 AlarmID 是唯一的，由系统进行分配。

AlarmID (readonly)

参见

Alarms（列表） (页 215)

AlarmLog

描述

在运行系统中无访问权限。

AlarmLogs

说明

返回"AlarmLogs"类型对象。

运行系统中可进行的访问：Read

语法

Object.**AlarmLogs**

Object

要求"Logging"对象。

参见

Logging (页 229)

AlarmLowerLimit

描述

指定触发报警的下限值。

运行系统中的访问权限： 读和写

语法

Object.**AlarmLowerLimit**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

DOUBLE

可选项。用于指定触发报警下限的值或常量。

注释

通过"AlarmLowerLimitRelative"属性定义估算的类型（百分比或绝对值）。

"AlarmLowerLimitEnable"属性定义是否启用对该限值的监视。

参见

Bar (页 285)

AlarmLowerLimitColor

描述

指定"AlarmLowerLimit"极值的滑块颜色。

若滚动条颜色在达到限值后立即更改，则"AlarmLowerLimitEnable"属性值必须为 TRUE。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**AlarmLowerLimitColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar

在运行系统中您没有以下格式的访问权限：

- Slider

Color

可选项。用于指定"AlarmLowerLimit"限值滑块颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Bar (页 285)

Slider (页 485)

AlarmLowerLimitEnabled

描述

指定是否监视“AlarmLowerLimit”限值。

运行系统中的访问权限： 读和写

语法

Object.**AlarmLowerLimitEnabled**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

BOOLEAN

可选。如果监视“AlarmLowerLimit”限值，则为 TRUE。

注释

通过属性“AlarmLowerLimit”、“AlarmLowerLimitColor”和“AlarmLowerLimitRelative”定义以下值：

限值

达到限值后的显示

估算的类型

参见

Bar (页 285)

AlarmLowerLimitRelative

描述

确定触发中断的下限值以百分比还是绝对值进行估算。

运行系统中的访问权限： 读和写

语法

Object.**AlarmLowerLimitRelative**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

BOOLEAN

可选。若触发中断的下限值以百分比进行估算，则为 TRUE。

参见

Bar (页 285)

AlarmSource

描述

在运行系统中无访问权限。

报警文本变量

描述

在运行系统中无访问权限。

AlarmUpperLimit

描述

确定触发中断的上限。

运行系统中的访问权限： 读和写

语法

Object.**AlarmUpperLimit**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

DOUBLE

可选项。用于定义触发中断的上限的值或常量。

注释

通过“AlarmUpperLimitRelative”属性定义估算的类型（百分比或绝对值）。

“AlarmUpperLimitEnable”属性定义是否启用对该限值的监视。

参见

Bar (页 285)

AlarmUpperLimitColor

描述

指定“AlarmUpperLimit”极值的滑块颜色。

若滚动条颜色在达到限值后立即更改，则“AlarmUpperLimitEnable”属性值必须为 TRUE。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**AlarmUpperLimitColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar

在运行系统中您没有以下格式的访问权限：

- Slider

Color

可选项。用于指定"AlarmUpperLimit"限值滑块颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Bar (页 285)

Slider (页 485)

AlarmUpperLimitEnabled

描述

指定是否监视"AlarmUpperLimit"限值。

运行系统中的访问权限： 读和写

语法

Object.**AlarmUpperLimitEnabled**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

BOOLEAN

可选。如果监视“AlarmUpperLimit”限值，则为 TRUE。

注释

通过属性“AlarmUpperLimit”、“AlarmUpperLimitColor”和“AlarmUpperLimitRelative”定义以下值：

- 限值
- 达到限值后的显示
- 估算的类型

参见

Bar (页 285)

AlarmUpperLimitRelative

描述

确定触发中断的上限值以百分比还是绝对值进行估算。

运行系统中的访问权限： 读和写

语法

Object.**AlarmUpperLimitRelative**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

BOOLEAN

可选。若触发中断的上限值以百分比进行估算，则为 TRUE。

参见

Bar (页 285)

AllFilters

描述

在运行系统中无访问权限。

AllFiltersForHitlist

描述

在运行系统中无访问权限。

AllowEdit

描述

在运行系统中无访问权限。

AllowMenu

描述

在运行系统中无访问权限

AllServer

说明

指定是否显示所有可用服务器中的报警。

运行系统中的访问权限：读和写

语法

Object.**AllServer**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- AlarmControl

BOOLEAN

可选项。

如果显示所有可用服务器中的报警，则选择 TRUE。

如果仅显示所选服务器中的报警，则选择 FALSE, 。

参见

AlarmControl (页 255)

AllTagTypesAllowed

说明

在运行系统中无访问权限。

Analog

描述

指定时针是否显示为模拟时钟。

运行系统中的访问权限： 读和写

语法

Object.**Analog**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Clock

BOOLEAN

可选。若时钟显示为模拟时钟，则为 TRUE。

参见

Clock (页 323)

AngleMax

描述

指定“Gauge”对象刻度范围的角度。

运行系统中的访问权限：读和写

语法

Object.**AngleMax**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Gauge

DOUBLE

可选项。用于以度为单位指定角度的值或常量。

注释

起始刻度等级和结束刻度等级在属性“AngleMin”和“AngleMax”中以角度为单位描述。

AngleMin 属性的值必须始终小于 AngleMax 属性的值。 零度角度位于刻度上 3 点的位置。角度的正值按顺时针计数。

参见

Gauge (页 366)

AngleMin

描述

指定“Gauge”对象起始刻度的角度。

运行系统中的访问权限：读和写

语法

Object.**AngleMin**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Gauge

DOUBLE

可选项。用于以度为单位指定角度的值或常量。

注释

起始刻度等级和结束刻度等级在属性“AngleMin”和“AngleMax”中以角度为单位描述。

AngleMin 属性的值必须始终小于 AngleMax 属性的值。

零度角度位于刻度上 3 点的位置。 角度的正值按顺时针计数。

参见

Gauge (页 366)

AnimationIgnore

说明

在运行系统中无访问权限。

Appearance

描述

在运行系统中无访问权限。

ApplyProjectSettings

描述

指定是否从“HMI 报警”编辑器中应用项目设置。

运行系统中可进行读写访问

语法

Object.**ApplyProjectSettings**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

BOOLEAN

可选项。

TRUE 在"HMI 报警"编辑器中组态的报警文本块连同其属性一起应用于报警视图时。报警视图中将显示报警文本块及其属性。

FALSE 属性未应用时。

参见

AlarmControl (页 255)

ApplyProjectSettingsForDesignMode

描述

在运行系统中无访问权限。

ArchiveName

说明

指定日志名称。

运行系统中的访问权限：读和写

语法

Object.**ArchiveName**[=STRING]

Object

必需项。 “ScreenItem”对象，且具有以下格式：

- UserArchiveControl

STRING

可选项。 用于指定日志名称的值或常量。

参见

用户归档控件 (页 566)

ArchiveType

说明

设置日志类型。

运行系统中的访问权限：读和写

语法

Object.**ArchiveType**[=RecipeControlDataSourceType]

Object

必需项。 “ScreenItem”对象，且具有以下格式：

- UserArchiveControl

RecipeControlDataSourceType

可选项。 用于指定日志类型的值或常量。

值	名称	说明
0	未知	指定无日志类型。
1	配方	指定日志类型为配方。
2	配方查询	指定日志类型为配方查询。

参见

用户归档控件 (页 566)

AskOperationMotive

描述

指定是否记录操作此对象的原因。如果在系统运行期间操作对象，操作员需要在对话框中输入原因。

运行系统中可进行读写访问

语法

Object.**AskOperationMotive**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- ComboBox
- IOField
- ListBox
- SymbolicIOField
- WindowSlider

BOOLEAN

可选项。若记录操作此对象的原因，则为 TRUE。

参见

ComboBox (页 326)

IOField (页 382)

Listbox (页 392)

SymbolicIOField (页 504)

WindowSlider (页 586)

AspectRatio

说明

指定大小改变时，是否保持媒体播放器的纵横比。

运行系统中的访问权限：读和写

语法

Object.**AspectRatio**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- MediaPlayer

BOOLEAN

可选项。

如果在大小改变时不保持媒体播放器的纵横比，则选择 TRUE。

如果在大小改变时不保持媒体播放器的纵横比，则选择 FALSE。

参见

MediaPlayer (页 396)

AssignedFilters

说明

在运行系统中无访问权限。

AssignedHitlistFilters

说明

在运行系统中无访问权限。

Assignments

描述

指定包含输出值与实际要输出的输出文本之间分配关系的列表。 分配取决于设置的列表类型。 可通过 ListType 属性定义列表类型。

运行系统中的访问权限：读和写

语法

Object.**Assignments**[=STRING]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- SymbolicIOField

STRING

可选项。指定包含输出值与实际要输出的输出文本之间分配关系的列表。

参见

SymbolicIOField (页 504)

AssociatedS7GraphDBName

说明

在运行系统中无访问权限。

AssociatedS7GraphDBTag

描述

在运行系统中无访问权限。

Authorization

描述

指定操作权限。

运行系统中可进行的访问：读和写

语法

Object.**Authorization**[=HMIRTAuthorization]

Object

必选。"ScreenItem" 对象，且具有以下格式：

- Bar
- Button
- CheckBox*
- Circle *
- CircleSegment *
- CircularArc *
- ComboBox*
- Connector *
- DateTimeField **
- Ellipse *
- EllipseSegment *
- EllipticalArc *
- GraphicIOField
- GraphicView *
- IOField
- Line *
- ListBox *
- MultiLineEdit *
- OptionGroup *
- Polygon *
- Polyline *
- RecipeView **
- Rectangle *
- RoundButton *
- S7GraphOverview *

- Slider
- StatusForce **
- Switch **
- SymbolLibrary **
- SymbolicIOField
- TextField *
- TrendView
- TubeArcObject *
- TubeDoubleTeeObject *
- TubeTeeObject *
- Tubepolyline *
- UserView **
- WindowSlider *

* 在 RT Advanced 中无访问权限

** 在 RT Professional 中无访问权限

在运行系统中您没有以下格式的访问权限：

- AlarmView
- PLCCodeViewer
- PdfView
- ProtectedAreaNameView
- RangeLabelView
- SysDiagControl

HMIRTAuthorization

可选项。用于指定操作权限的值或常量。

AutoCompleteColumns

描述

指定在控件比组态的列宽时是否显示空列。

运行系统中可进行读写访问

语法

Object.**AutoCompleteColumns**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

如果显示空列，则为 TRUE。

如果不显示空列，则为 FALSE。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

AutoCompleteRows

描述

指定在控件比组态的行数长时是否显示空行。

运行系统中可进行读写访问

语法

Object.**AutoCompleteRows**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

如果显示空行，则为 TRUE。

如果不显示空行，则为 FALSE。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

AutoPosition

说明

指定值表格是否自动位于为值表格提供数据的对象下。

运行系统中的访问权限：读和写

语法

Object.**AutoPosition**[=BOOLEAN]

Object

必需项。 具有以下格式的"ScreenItem"类型的对象：

- TrendRulerControl

BOOLEAN

可选项。

如果对象自动位于数据源下，则选择 TRUE。

如果对象位于组态位置，则选择 FALSE。

参见

TrendRulerControl (页 532)

AutoScroll

描述

指定是否使用自动滚动。

运行系统中可进行读写访问

语法

Object.**AutoScroll**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

BOOLEAN

可选项

如果使用自动滚动，则为 TRUE。

参见

AlarmControl (页 255)

AutoSelectionColors

描述

指定是否将系统定义的颜色用作单元格和行的选择颜色。

运行系统中可进行读写访问

语法

Object.**AutoSelectionColors**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

如果使用系统颜色，则为 TRUE。

如果使用自定义颜色，则为 FALSE。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

AutoSelectionRectColor

描述

指定是否使用系统定义的颜色显示选择框架。

运行系统中的访问权限：读和写

语法

Object.**AutoSelectionRectColor**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

如果使用系统颜色，则为 TRUE。

如果使用自定义颜色，则为 FALSE。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

AutoShow

说明

指定是否自动显示值表格。

运行系统中的访问权限：读和写

语法

Object.**AutoShow**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- TrendRulerControl

BOOLEAN

可选项。

如果自动显示值表格，则选择 TRUE。

如果不自动显示值表格，则选择 FALSE。

参见

TrendRulerControl (页 532)

AutoSizing

描述

在运行系统中无访问权限。

AutoStart

说明

在运行系统中无访问权限。

AvailableStatusbarElements

说明

在运行系统中无访问权限。

AvailableToolbarButtons

说明

在运行系统中无访问权限。

AverageLast15Values

描述

指定是否显示最后 15 个值的平均值。

运行系统中可进行读写访问

语法

Object.**AverageLast15Values**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar

BOOLEAN

可选项。

如果显示最后 15 个值的平均值，则为 TRUE。

如果不显示最后 15 个值的平均值，则为 FALSE。

参见

Bar (页 285)

AxisXBunchCount

描述

在运行系统中无访问权限。

AxisXMarkCount

描述

在运行系统中无访问权限。

AxisXNoOfDigits

说明

在运行系统中无访问权限。

AxisXShowBunchValues

描述

在运行系统中无访问权限。

AxisXStyle

说明

在运行系统中无访问权限。

AxisY1BunchCount

描述

在运行系统中无访问权限。

AxisY1MarkCount

描述

在运行系统中无访问权限。

AxisY1ShowBunchValues

描述

在运行系统中无访问权限。

AxisY2BunchCount

描述

在运行系统中无访问权限。

AxisY2MarkCount

描述

在运行系统中无访问权限。

AxisY2ShowBunchValues

描述

在运行系统中无访问权限。

1.5.5.2 属性 B

BackButtonVisible

说明

在运行系统中无访问权限。

BackColor

描述

指定背景色。

运行系统中的访问权限：读和写

语法

Object.**BackColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl *
- AlarmView
- Bar
- Button **
- CheckBox *
- Circle
- CircleSegment *
- CircularArc *
- ComboBox *
- Connector *
- DateTimeField **
- Ellipse
- EllipseSegment *
- EllipticalArc *
- FunctionTrendControl *
- Gauge
- GraphicIOField **
- GraphicView
- IOField
- Line
- ListBox *
- MultiLineEdit *
- OnlineTableControl *
- OnlineTrendControl *
- OptionGroup *
- Polygon
- Polyline

- RecipeView **
- Rectangle
- RoundButton *
- Slider
- StatusForce **
- Switch **
- SymbolLibrary
- SymbolicIOField
- TextField
- TrendRulerControl *
- UserArchiveControl *
- UserView **
- WindowSlider *

* RT Advanced 无访问权限

** RT Professional 无访问权限

在运行系统中您没有以下格式的访问权限：

- TrendView
- TubeArcObject

Color

可选项。用于指定背景色的数值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

如果"BorderStyle"属性的值为"0"，则背景色不可见。

参见

AlarmControl (页 255)

AlarmView (页 274)

Bar (页 285)

Button (页 296)

CheckBox (页 307)

Circle (页 312)

CircleSegment (页 316)

CircularArc (页 320)

ComboBox (页 326)

Connector (页 330)

DateTimeField (页 334)

Ellipse (页 340)

EllipseSegment (页 344)

EllipticalArc (页 348)

FunctionTrendControl (页 351)

Gauge (页 366)

GraphicIOField (页 371)

GraphicView (页 375)

IOField (页 382)

Line (页 388)

Listbox (页 392)

MultiLineEdit (页 399)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

OptionGroup (页 435)

Polygon (页 444)

Polyline (页 448)

RecipeView (页 458)

Rectangle (页 467)

RoundButton (页 471)

Slider (页 485)

StatusForce (页 493)

Switch (页 499)

SymbolicIOField (页 504)

SymbolLibrary (页 511)

TextField (页 527)

TrendRulerControl (页 532)

TrendView (页 546)

TubeArcObject (页 554)

用户归档控件 (页 566)

UserView (页 581)

WindowSlider (页 586)

BackColorBottom

描述

指定对象下半部/右侧的颜色。

运行系统中可进行读写访问

语法

Object.**BackColorBottom**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- WindowSlider

Color

可选项。用于指定 对象下半部/右侧颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 vbRed 和 vbGreen。

参见

WindowSlider (页 586)

BackColorTop

描述

指定对象上半部/左侧的颜色。

运行系统中可进行读写访问

语法

Object.**BackColorTop**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- WindowSlider

Color

可选项。用于指定对象上半部/左侧颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 vbRed 和 vbGreen。

参见

WindowSlider (页 586)

BackFillStyle

描述

指定填充图案。

运行系统中的访问权限：读和写

语法

Object.**BackFillStyle**[=FillStyle]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Bar
- Button*
- CheckBox*
- Circle
- CircleSegment *
- Clock*
- ComboBox
- Ellipse
- EllipseSegment *
- Gauge*
- GraphicView*
- IOField*
- ListBox*
- OptionGroup*
- Polygon
- Rectangle

- RoundButton*
- Slider*
- SymbolLibrary*
- SymbolicIOField*
- TextField*
- WindowSlider*

* RT Advanced 无访问权限

在运行系统中您没有以下格式的访问权限：

- DateTimeField
- GraphicIOField
- Switch

FillStyle

可选项。用于指定填充样式的值或常量。

值	VB 常量	说明
0	hmiFillStyleSolid	使用指定颜色填充对象。
1	hmiFillStyleTransparent	透明填充

参见

Bar (页 285)
Button (页 296)
CheckBox (页 307)
Clock (页 323)
ComboBox (页 326)
DateTimeField (页 334)
Gauge (页 366)
GraphicIOField (页 371)
GraphicView (页 375)

IOField (页 382)

Listbox (页 392)

OptionGroup (页 435)

RoundButton (页 471)

Slider (页 485)

Switch (页 499)

SymbolicIOField (页 504)

SymbolLibrary (页 511)

WindowSlider (页 586)

BackFlashingColorOff

描述

指定闪烁状态“关闭”(Off) 的背景色。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**BackFlashingColorOff**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Bar
- Button
- CheckBox
- GraphicView
- IOField
- OptionGroup
- RoundButton

- SymbolicIOField
- TextField
- WindowSlider

在运行系统中您没有以下格式的访问权限：

- Switch

Color

可选项。用于指定闪烁状态“关闭”背景颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Bar (页 285)

Button (页 296)

CheckBox (页 307)

GraphicView (页 375)

IOField (页 382)

OptionGroup (页 435)

RoundButton (页 471)

Switch (页 499)

SymbolicIOField (页 504)

WindowSlider (页 586)

BackFlashingColorOn

描述

指定闪烁状态“打开"(On) 的背景色。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**BackFlashingColorOn**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox
- GraphicView
- IOField
- OptionGroup
- RoundButton
- SymbolicIOField
- TextField
- WindowSlider

在运行系统中您没有以下格式的访问权限：

- Switch

Color

可选项。用于指定闪烁状态"开启"背景颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Bar (页 285)

Button (页 296)

CheckBox (页 307)

GraphicView (页 375)

IOField (页 382)

OptionGroup (页 435)

RoundButton (页 471)

Switch (页 499)

SymbolicIOField (页 504)

WindowSlider (页 586)

BackFlashingEnabled

描述

指定是否在运行系统中闪烁背景。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**BackFlashingEnabled**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox
- GraphicView
- IOField

- OptionGroup
- RoundButton
- SymbolicIOField
- TextField
- WindowSlider

在运行系统中您没有以下格式的访问权限：

- Switch

BOOLEAN

可选。如果在运行系统中背景闪烁则为 TRUE。

参见

Bar (页 285)

Button (页 296)

CheckBox (页 307)

GraphicView (页 375)

IOField (页 382)

OptionGroup (页 435)

RoundButton (页 471)

Switch (页 499)

SymbolicIOField (页 504)

WindowSlider (页 586)

BackFlashingRate

说明

指定背景的闪烁频率。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**BackFlashingRate**[=FlashingRate]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox
- GraphicView
- IOField
- OptionGroup
- RoundButton
- SymbolicIOField
- TextField
- WindowSlider

在运行系统中您没有以下格式的访问权限：

- Switch

FlashingRate

可选项。用于指定背景的闪烁速率的值或常量。

值	VB 常量	说明
0	hmiFlashingRateSlow	闪烁间隔为 250 ms。
1	hmiFlashingRateMedium	闪烁间隔为 500 ms。
2	hmiFlashingRateFast	闪烁间隔为 1000 ms。

参见

Bar (页 285)

Button (页 296)

CheckBox (页 307)

GraphicView (页 375)

IOField (页 382)

OptionGroup (页 435)

RoundButton (页 471)

Switch (页 499)

SymbolicIOField (页 504)

WindowSlider (页 586)

BackgroundColor

描述

在运行系统中无访问权限。

参见

TrendRulerControl (页 532)

BackPicture

说明

指定背景图形。

运行系统中的访问权限：读和写

语法

Object.**BackPicture**[=HmiObjectHandle]

Object

必需项。 “ScreenItem”对象，且具有以下格式：

- Gauge
- Slider

HmiObjectHandle

可选项。 用于指定背景图形的数值或常量。

参见

Gauge (页 366)

Slider (页 485)

BackStyle

说明

指定背景样式。

运行系统中的访问权限：读和写

语法

Object.**BackStyle**[=Int32]

Object

必需项。 “ScreenItem”对象，且具有以下格式：

- Gauge

Int32

可选项。 用于指定背景样式的数值或常量。

值	名称	说明
0	实线	显示的矩形背景以指定的边框颜色填充。 刻度磁盘填充有指定的背景色。
1	透明边框	表盘的矩形背景是透明的。 刻度磁盘填充有指定的背景色。因此，呈现圆形显示。
2	透明	矩形背景和刻度是透明的。

参见

Gauge (页 366)

BarBackColor

描述

确定所选对象的栏的背景色。

运行系统中可进行读写访问

语法

Object.**BarBackColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Slider

Color

可选项。用于指定棒图背景颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Bar (页 285)

Slider (页 485)

BarBackFillStyle

描述

指定滑块的填充样式。

运行系统中可进行读写访问

语法

Object.**BarBackFillStyle**[=FillStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar

FillStyle

可选项。用于指定填充样式的值或常量。

值	VB 常量	说明
65536	hmiFillStyleTransparent	透明填充图案
0	hmiFillStyleSolid	实心填充图案
131075	hmiFillStyleBackwardDiagonal	沿右上角方向对角线条纹填充图案
131076	hmiFillStyleCross	格子填充图案
131077	hmiFillStyleDiagonalCross	对角线格子填充图案
131074	hmiFillStyleForwardDiagonal	沿左上角方向对角线条纹填充图案
131072	hmiFillStyleHorizontal	水平条纹填充图案
131073	hmiFillStyleVertical	垂直条纹填充图案
196608 到 196644	hmiFillStylePattern1 - hmiFillStylePattern37	预定义填充图案

参见

Bar (页 285)

BarBackFlashingColorOff

描述

指定颜色 ###

运行系统中的访问权限：读和写

语法

Object.**BarBackFlashingColorOff**[=Color]

Object

必选项。具有以下格式的"ScreenItem"类型的对象：

- Bar

Color

可选项。用于指定 ### 颜色的值或常量。

BarBackFlashingColorOn

描述

指定颜色 ###

运行系统中的访问权限：读和写

语法

Object.**BarBackFlashingColorOn**[=Color]

Object

必选项。具有以下格式的"ScreenItem"类型的对象：

- Bar

Color

可选项。用于指定 ### 颜色的值或常量。

BarBackFlashingEnabled

描述

指定是否 ###

运行系统中的访问权限：读和写

语法

Object.**BarBackFlashingEnabled**[=BOOLEAN]

Object

必选项。具有以下格式的"ScreenItem"类型的对象：

- Bar

BOOLEAN

可选项。如果 ### 则为 TRUE

BarBackFlashingRate

描述

设置闪烁频率 ###

运行系统中的访问权限：读和写

语法

Object.**BarBackFlashingRate**[=FlashingRate]

Object

必选项。具有以下格式的"ScreenItem"类型的对象：

- Bar

FlashingRate

可选项。用于指定闪烁频率 ### 的值或常量

BarColor

描述

指定工具栏的颜色。

运行系统中可进行读写访问

语法

Object.**BarColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Slider

Color

可选项。用于指定滑块颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

范围从 "MinimumValue (页 979)"延伸至滑块位置。

参见

Slider (页 485)

BarEdgeStyle

描述

指定 ###

运行系统中的访问权限：

- RT Advanced：读取
- RT Professional：读和写

语法

Object.**BarEdgeStyle**[=LineStyle]

Object

必选项。具有以下格式的"ScreenItem"类型的对象：

- Bar

LineStyle

可选项。值或常量，其中 ###

BarOrientation

说明

指定棒图对齐方式。

运行系统中的访问权限

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**BarOrientation**[=BarOrientation]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar

在运行系统中您没有以下格式的访问权限：

- Slider

BarOrientation

可选项。用于指定棒图方向的值或常量。

值	名称	说明
0	顶部	垂直对齐棒图。最大刻度值位于棒图顶部。
1	底部	垂直对齐棒图。最大刻度值位于棒图底部。
2	左对齐	水平对齐棒图。最大刻度值位于左侧。
3	右对齐	水平对齐棒图。最大刻度值位于右侧。

参见

Bar (页 285)

Slider (页 485)

BaseScreenName

说明

通过设置新画面名称，读取当前根画面的名称或触发根画面更改。

运行系统中可进行的访问： 读和写

语法

Object.**BaseScreenName**[= STRING]

Object

要求"HMIRuntime"对象。

STRING

包含画面名称的选项 A 值或常量。

注释

还可以使用属性确定当前显示的画面。

参见

HMIRuntime（页 224）

BelowLowerLimitColor

描述

为 "低于下限"(Low limit violated) 事件指定颜色。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**BelowLowerLimitColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- IOField

在运行系统中您没有以下格式的访问权限：

- GraphicIOField
- Switch
- SymbolLibrary
- SymbolicIOField

Color

可选项。用于指定颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

BitNumber

描述

指定必须更改状态才能触发值更改的位。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**BitNumber**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- SymbolicIOField

在运行系统中您没有以下格式的访问权限：

- Button
- GraphicIOField

Int32

可选择指定必须更改状态才能触发值更改的位。

注释

使用的变量必须是类型 BYTE、WORD 或 DWORD。

参见

SymbolicIOField (页 504)

Button (页 296)

GraphicIOField (页 371)

BlinkColor

描述

指定对象在运行系统中闪烁的颜色。

运行系统中可进行读写访问

语法

Object.**BlinkColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- SymbolLibrary

Color

可选项。用于指定闪烁颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

SymbolLibrary (页 511)

BlinkMode

描述

指定所选对象的闪烁图形的类型。

运行系统中可进行读写访问

语法

Object.**BlinkMode**[=SymbolLibraryBlinkMode]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- SymbolLibrary

SymbolLibraryBlinkMode

可选项。用于指定所选对象闪烁画面类型的值或常量。

值	VB 常量	说明
0	hmiSymbolLibraryFlashingNone	闪烁关闭。
1	hmiSymbolLibraryFlashingInvisible	闪烁画面可见。
2	hmiSymbolLibraryFlashingShaded	闪烁画面呈现彩色阴影表面。表面颜色对应属性"BlinkColor"的设置。
3	hmiSymbolLibraryFlashingSolid	闪烁画面呈现彩色无阴影表面。表面颜色对应属性"BlinkColor"的设置。

参见

SymbolLibrary (页 511)

BlinkSpeed

描述

设置闪烁速度。

快速 - 250：闪烁间隔为 250 ms。中速 - 500：闪烁间隔为 500 ms。

缓慢 - 1000：闪烁间隔为 1000 ms。默认值为中速 - 500。

运行系统中可进行读写访问

语法

Object.**BlinkSpeed**[=FlashingRate]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- SymbolLibrary

FlashingRate

可选项。用于指定闪烁速率的值或常量。

值	VB 常量	说明
0	hmiFlashingRateSlow	闪烁间隔为 250 ms。
1	hmiFlashingRateMedium	闪烁间隔为 500 ms。
2	hmiFlashingRateFast	闪烁间隔为 1000 ms。

参见

SymbolLibrary (页 511)

BlockAlignment

说明

指定引用块的列标题内的文本对齐方式。

运行系统中可进行读写访问

语法

Object.**BlockAlignment**[=HorizontalAlignment]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- TrendRulerControl

HorizontalAlignment

可选项。确定水平对齐方式的值或常量。

值	VB 常量	说明
0	hmiAlignmentLeft	列标题内文本左对齐。
1	hmiAlignmentCentered	列标题内文本居中。
2	hmiAlignmentRight	列标题内文本右对齐。

参见

TrendRulerControl (页 532)

BlockAutoPrecision

说明

指定是否自动调整当前块中显示的小数位数。

运行系统中的访问权限：读和写

语法

Object.**BlockAutoPrecisions**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下格式：

- TrendRulerControl

BOOLEAN

可选项。

TRUE，实现自动调整显示的小数位数。

FALSE，实现不自动调整显示的小数位数。

参见

TrendRulerControl (页 532)

BlockCaption

说明

指定当前块的标题。

运行系统中的访问权限：读和写

语法

Object.**BlockCaption**[=STRING]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- TrendRulerControl

STRING

可选项。 用于指定块标题的值或常量。

参见

TrendRulerControl (页 532)

BlockCount

说明

指定块数。

运行系统中的访问权限：读和写

语法

Object.**BlockCount**[=Int32]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- TrendRulerControl

Int32

可选项。 用于指定块数的值或常量。

参见

TrendRulerControl (页 532)

BlockDateFormat

说明

指定当前块中日期信息的格式。

运行系统中的访问权限：读和写

语法

Object.**BlockDateFormat**[=STRING]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- TrendRulerControl

STRING

可选项。 用于指定日期信息格式的值或常量。

参见

TrendRulerControl (页 532)

BlockExponentialFormat

说明

指定是否用指数记数法显示当前块中的值。

运行系统中的访问权限：读和写

语法

Object.**BlockExponentialFormat**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- TrendRulerControl

BOOLEAN

可选项。

TRUE，用指数计数法显示数值。

FALSE，用十进制计数法显示数值。

参见

TrendRulerControl (页 532)

BlockHideText

说明

在运行系统中无访问权限。

BlockHideTitleText

说明

指定是否将块标题显示为文本。

运行系统中的访问权限：读和写

语法

Object.**BlockHideTitleText**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- TrendRulerControl

BOOLEAN

可选项。

TRUE，将块标题显示为文本。

FALSE，不将块标题显示为文本。

参见

TrendRulerControl (页 532)

BlockId

说明

使用块 ID 引用块。要访问块的属性，需要设置“BlockId”。

运行系统中可进行读写访问

语法

Object.**BlockId**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- TrendRulerControl

Int32

可选项。用于指定当前块 ID 的值或常量。

参见

TrendRulerControl (页 532)

BlockIndex

说明

引用块。要访问块的属性，需要设置“BlockIndex”。

介于 0 至 (BlockIndex - 1) 之间的值为“BlockCount”的有效值。“BlockCount”属性可指定已组态块的数目。

运行系统中可进行读写访问

语法

Object.**BlockIndex**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- TrendRulerControl

Int32

可选项。用于指定当前块编号的值或常量。

参见

TrendRulerControl (页 532)

BlockLength

说明

指定当前块的字符数。

运行系统中的访问权限：读和写

语法

Object.**BlockLength**[=Int32]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- TrendRulerControl

Int32

可选项。 用于指定字符数的值或常量。

参见

TrendRulerControl (页 532)

BlockName

说明

指定当前块的名称。

运行系统中的访问权限：读和写

语法

Object.**BlockName**[=STRING]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- TrendRulerControl

STRING

可选项。 用于指定块名称的值或常量。

参见

TrendRulerControl (页 532)

BlockPrecision

说明

指定当前块中的小数位数。

运行系统中的访问权限：读和写

语法

Object.**BlockPrecisions**[=Int16]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- TrendRulerControl

Int16

可选项。 用于指定小数位数的值或常量。

参见

TrendRulerControl (页 532)

Blocks

描述

在运行系统中无访问权限。

BlockShowDate

说明

指定当前块中显示的日期。

运行系统中的访问权限：读和写

语法

Object.**BlockShowDate**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- TrendRulerControl

BOOLEAN

可选项。

TRUE，在当前块中显示日期。

FALSE，不在当前块中显示日期。

参见

TrendRulerControl (页 532)

BlockShowIcon

说明

指定是否将当前块的内容显示为图标。

运行系统中的访问权限：读和写

语法

Object.**BlockShowIcon**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- TrendRulerControl

BOOLEAN

可选项。

TRUE，将当前块的内容显示为图标。

FALSE，不将当前块的内容显示为图标。

参见

TrendRulerControl (页 532)

BlockShowTitleIcon

说明

指定是否将当前块的标题显示为图标。

运行系统中的访问权限：读和写

语法

Object.**BlockShowTitleIcon**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- TrendRulerControl

BOOLEAN

可选项。

TRUE，将当前块的标题显示为图标。

FALSE，不将当前块的标题显示为图标。

参见

TrendRulerControl (页 532)

BlockTimeFormat

说明

指定当前块中时间信息的格式。

运行系统中的访问权限：读和写

语法

Object.**BlockTimeFormat**[=STRING]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- TrendRulerControl

STRING

用于指定时间信息格式的值或常量。

参见

TrendRulerControl (页 532)

BlockUseSourceFormat

说明

指定在当前块中是否采用连接控件格式。

运行系统中的访问权限：读和写

语法

Object.**BlockUseSourceFormat**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- TrendRulerControl

BOOLEAN

可选项。

TRUE，在当前块中采用连接控件格式。

FALSE，在当前块中采用组态格式。

参见

TrendRulerControl (页 532)

BorderBackColor

描述

指定间断边框线的背景色。

运行系统中的访问权限

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**BorderBackColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- ComboBox
- Ellipse
- EllipseSegment
- GraphicIOField
- GraphicView
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- Pölygon
- Rectangle
- RoundButton
- TextField
- WindowSlider

在运行系统中您没有以下格式的访问权限：

- AlarmView
- Clock
- DateTimeField
- Gauge
- RecipeView
- Slider
- StatusForce
- Switch
- SymbolicIOField
- SysDiagControl
- TrendView
- UserView

Color

可选项.指定间断边框线的背景色值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmView (页 274)

Bar (页 285)

Button (页 296)

CheckBox (页 307)

Clock (页 323)

ComboBox (页 326)

DateTimeField (页 334)

GraphicIOField (页 371)

GraphicView (页 375)

Gauge (页 366)

IOField (页 382)

Listbox (页 392)

MultiLineEdit (页 399)

OptionGroup (页 435)

RecipeView (页 458)

RoundButton (页 471)

Slider (页 485)

StatusForce (页 493)

Switch (页 499)

SymbolicIOField (页 504)

SysDiagControl (页 514)

TextField (页 527)

TrendView (页 546)

UserView (页 581)

WindowSlider (页 586)

BorderBrightColor3D

描述

指定 3D 显示中以下边框部分的边框颜色：

- 外侧边框部分（上、下）
- 内侧边框部分（上、右）

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**BorderBrightColor3D**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Button
- RoundButton
- Slider*

* RT Advanced 读写访问权限

在运行系统中您没有以下格式的访问权限：

- Switch

Color

可选项。用于指定边框颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Button (页 296)

RoundButton (页 471)

Slider (页 485)

Switch (页 499)

BorderColor

描述

指定线颜色。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**BorderColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox
- Circle*
- CircleSegment
- ComboBox
- DateTimeField**
- Ellipse*
- EllipseSegment
- FunctionTrendControl
- GraphicIOField
- GraphicView
- IOField
- ListBox
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- OptionGroup
- Polygon*
- Rectangle*
- RoundButton

- SymbolicIOField*
- TextField***
- TrendRulerControl
- UserArchiveControl
- WindowSlider

* RT Advanced 读写访问权限

** RT Advanced 读写访问权限，RT Professional 无访问权限

*** RT Advanced 无访问权限

在运行系统中您没有以下格式的访问权限：

- AlarmControl
- AlarmView
- Clock
- Gauge
- RecipeView
- Slider
- StatusForce
- Switch
- SysDiagControl
- TrendView
- UserView

Color

可选项。指定线颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

AlarmView (页 274)

Bar (页 285)

Button (页 296)

CheckBox (页 307)

Circle (页 312)

CircleSegment (页 316)

Clock (页 323)

ComboBox (页 326)

DateTimeField (页 334)

Ellipse (页 340)

EllipseSegment (页 344)

FunctionTrendControl (页 351)

Gauge (页 366)

GraphicIOField (页 371)

GraphicView (页 375)

IOField (页 382)

Listbox (页 392)

MultiLineEdit (页 399)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

OptionGroup (页 435)

Polygon (页 444)

RecipeView (页 458)

Rectangle (页 467)

RoundButton (页 471)

Slider (页 485)

StatusForce (页 493)

Switch (页 499)

SymbolicIOField (页 504)

SysDiagControl (页 514)

TextField (页 527)

TrendRulerControl (页 532)

TrendView (页 546)

用户归档控件 (页 566)

UserView (页 581)

WindowSlider (页 586)

BorderEnabled

描述

返回关于窗口在运行系统中是否带边框显示的信息。

运行系统中的访问权限：读和写

语法

Object.**BorderEnabled**[=BOOLEAN]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- ApplicationWindow
- Screenwindow

BOOLEAN

可选项。如果窗口在运行系统中带边框显示，则为 TRUE。

参见

ApplicationWindow (页 282)

ScreenWindow (页 479)

BorderEndStyle

描述

指定所选对象线端的类型。

运行系统中的访问权限：读和写

语法

Object.**BorderEndStyle**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Connector
- Line
- Polyline

Int32

可选项。用于指定线端类型的值或常量。

参见

Connector (页 330)

Line (页 388)

Polyline (页 448)

BorderFlashingColorOff

描述

指定闪烁状态“关闭”时的边框线颜色。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**BorderFlashingColorOff**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- Ellipse
- EllipseSegment
- GraphicIOField
- GraphicView
- IOField
- OptionGroup
- Polyline
- Polygon
- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- WindowSlider

在运行系统中您没有以下格式的访问权限：

- Switch

Color

可选项。用于指定闪烁状态“关闭”时边框线颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Bar (页 285)

Button (页 296)

CheckBox (页 307)

GraphicIOField (页 371)

IOField (页 382)

OptionGroup (页 435)

RoundButton (页 471)

Switch (页 499)

SymbolicIOField (页 504)

TextField (页 527)

WindowSlider (页 586)

BorderFlashingColorOn

描述

指定闪烁状态"开启"时的边框线颜色。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**BorderFlashingColorOn**[=Color]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- Ellipse
- EllipseSegment
- GraphicIOField
- GraphicView
- IOField
- OptionGroup
- Polyline
- Polygon
- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- WindowSlider

在运行系统中您没有以下格式的访问权限：

- Switch

Color

可选项。用于指定闪烁状态“开启”时边框线颜色的值或常量。

注释

可以使用“RGB”函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Bar (页 285)

Button (页 296)

CheckBox (页 307)

GraphicIOField (页 371)

IOField (页 382)

OptionGroup (页 435)

RoundButton (页 471)

Switch (页 499)

SymbolicIOField (页 504)

WindowSlider (页 586)

BorderFlashingEnabled

说明

指定是否在运行系统中闪烁对象的限值。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**BorderFlashingEnabled**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox

- Circle
- CircleSegment
- Ellipse
- EllipseSegment
- GraphicIOField
- GraphicView
- IOField
- OptionGroup
- Polyline
- Polygon
- Rectangle
- RoundButton
- SymbolicIOField
- TextField

在运行系统中您没有以下格式的访问权限：

- Switch
- WindowSlider

BOOLEAN

可选项。

如果对象限值可以在运行系统中闪烁则为 TRUE。

如果对象限值不能在运行系统中闪烁则为 FALSE。

参见

Bar (页 285)

Button (页 296)

CheckBox (页 307)

GraphicIOField (页 371)

IOField (页 382)

OptionGroup (页 435)

RoundButton (页 471)

Switch (页 499)

SymbolicIOField (页 504)

WindowSlider (页 586)

BorderFlashingRate

说明

指定边界线的闪烁频率。

运行系统中可进行的访问：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**BorderFlashingRate**[=FlashingRate]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- Ellipse
- EllipseSegment
- GraphicView
- IOField
- OptionGroup
- Polyline
- Polygon

- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- WindowSlider

在运行系统中您没有以下格式的访问权限：

- GraphicIOField
- Switch

FlashingRate

可选项。用于指定边框线的闪烁速率的值或常量。

值	VB 常量	说明
0	hmiFlashingRateSlow	闪烁间隔为 250 ms。
1	hmiFlashingRateMedium	闪烁间隔为 500 ms。
2	hmiFlashingRateFast	闪烁间隔为 1000 ms。

参见

Bar (页 285)

Button (页 296)

CheckBox (页 307)

GraphicIOField (页 371)

IOField (页 382)

OptionGroup (页 435)

RoundButton (页 471)

Switch (页 499)

SymbolicIOField (页 504)

WindowSlider (页 586)

BorderInnerStyle3D

说明

指定对象边框的内侧部分的外观。

运行系统中可进行读写访问

语法

Object.**BorderInnerStyle3D**[=3DStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Gauge
- Slider

3DStyle

可选项。用于定义对象边框内侧部分显示方式的值或常量。

值	VB 常量	说明
0	hmiBorder3DStyleNone	对象边框没有内侧部分。
1	hmiBorder3DStyleRecessed	对象边框以浮雕形式显示。
2	hmiBorder3DStyleRaised	对象边框凸起显示。
3	hmiBorder3DStyleGray	对象边框统一显示为灰色。
4	hmiBorder3DStyleColored	对象边界颜色统一。边框颜色对应于背景色。

参见

Gauge (页 366)

Slider (页 485)

BorderInnerWidth3D

描述

以 3D 形式表现所选择的对象时，指定内边框的宽度。

运行系统中可进行读写访问

语法

Object.**BorderInnerWidth3D**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Slider

Int32

可选项。以像素为单位指定内边框宽度的值或常量。

参见

Slider (页 485)

BorderOuterStyle3D

说明

指定对象边框的外侧部分的外观。

运行系统中的访问权限

- RT Advanced: 无访问权限
- RT Professional: 读写访问权限

语法

Object.**BorderOuterStyle3D**[=3DStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Gauge

在运行系统中您没有以下格式的访问权限：

- Slider

3DStyle

可选项。用于定义对象边框外侧部分显示方式的值或常量。

值	VB 常量	说明
0	hmiBorder3DStyleNone	对象边框没有内侧部分。
1	hmiBorder3DStyleRecessed	对象边框以浮雕形式显示。
2	hmiBorder3DStyleRaised	对象边框凸起显示。
3	hmiBorder3DStyleGray	对象边框统一显示为灰色。
4	hmiBorder3DStyleColored	对象边界颜色统一。边框颜色对应于背景色。

参见

Gauge (页 366)

Slider (页 485)

BorderOuterWidth3D

描述

以 3D 形式表现所选择的对象时，指定外边框的宽度。

运行系统中可进行读写访问

语法

Object.**BorderOuterWidth3D**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Slider

Int32

可选项。用于指定以像素为单位的外边框宽度的值或常量。

参见

Slider (页 485)

BorderShadeColor3D

描述

指定 3D 显示中以下边框部分的边框颜色：

- 内侧边框部分（上、下）
- 外侧边框部分（下、右）

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**BorderShadeColor3D**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Button
- Slider*

* RT Advanced 读写访问权限

在运行系统中您没有以下格式的访问权限：

- RoundButton
- Switch

Color

可选项。用于指定底纹颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Button (页 296)

RoundButton (页 471)

Slider (页 485)

Switch (页 499)

BorderStyle

描述

指定边框线的闪烁类型。

运行系统中的访问权限：读和写

语法

Object.**BorderStyle**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- ComboBox
- Connector

- Ellipse
- EllipseSegment
- EllipticalArc
- GraphicIOField
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit
- Polygon
- Polyline
- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- WindowSlider

在运行系统中您没有以下格式的访问权限：

- OptionGroup

Int32

可选项。用于指定边框线类型的值或常量。

值	说明
0	实线
1	短划线
2	虚线
3	点划线
4	双点划线

参见

TextField (页 527)

Bar (页 285)

Button (页 296)

CheckBox (页 307)

ComboBox (页 326)

GraphicIOField (页 371)

IOField (页 382)

Listbox (页 392)

MultiLineEdit (页 399)

OptionGroup (页 435)

RoundButton (页 471)

SymbolicIOField (页 504)

WindowSlider (页 586)

BorderWidth

描述

指定线宽。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**BorderWidth**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Ala rmControl
- Bar

- Button
- Circle *
- CircleSegment
- CheckBox
- ComboBox
- Ellipse *
- EllipseSegment
- FunctionTrendControl
- Gauge
- GraphicIOField
- GraphicView
- IOField
- ListBox
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- Polygon *
- Rectangle *
- OptionGroup
- RoundButton
- Slider *
- SymbolicIOField *
- TextField *
- TrendRulerControl
- UserArchiveControl
- WindowSlider

* RT Advanced 读写访问权限

在运行系统中您没有以下格式的访问权限：

- AlarmView
- Clock
- DateTimeField
- RecipeView
- StatusForce
- Switch
- SysDiagControl
- TrendView
- UserView

Int32

可选项。用于指定以像素为单位的线宽的值或常量。

参见

AlarmControl (页 255)

AlarmView (页 274)

Bar (页 285)

Button (页 296)

CheckBox (页 307)

Clock (页 323)

ComboBox (页 326)

DateTimeField (页 334)

FunctionTrendControl (页 351)

Gauge (页 366)

GraphicIOField (页 371)

IOField (页 382)

Listbox (页 392)

MultiLineEdit (页 399)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

OptionGroup (页 435)

RecipeView (页 458)

RoundButton (页 471)

Slider (页 485)

StatusForce (页 493)

Switch (页 499)

SymbolicIOField (页 504)

SysDiagControl (页 514)

TrendRulerControl (页 532)

TrendView (页 546)

用户归档控件 (页 566)

UserView（页 581)

WindowSlider (页 586)

BorderWidth3D

描述

以 3D 形式表现所选择的对象时，指定内边框的宽度。

运行系统中的访问权限

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**BorderWidth3D**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Button
- Gauge
- RoundButton

在运行系统中您没有以下格式的访问权限：

- Bar
- IOField
- Switch

Int32

可选项。以像素为单位确定 3D 显示内边框宽度的值或常量。

参见

Bar (页 285)

Button (页 296)

Gauge (页 366)

IOField (页 382)

RoundButton (页 471)

Switch (页 499)

BottomMargin

描述

在运行系统中无访问权限。

Bounds

说明

在运行系统中无访问权限。

BrowserTypeUsed

说明

在运行系统中无访问权限。

BufferViewColumnOrder

说明

在运行系统中无访问权限。

BufferViewInternalRowOrder

说明

在运行系统中无访问权限。

BusyText

描述

在运行系统中无访问权限。

ButtonBackColor

描述

在运行系统中无访问权限。

ButtonBackFillStyle

描述

在运行系统中无访问权限。

ButtonBarElements

描述

在运行系统中无访问权限。

ButtonBarHeight

说明

在运行系统中无访问权限。

ButtonBarStyle

描述

在运行系统中无访问权限。

ButtonBorderBackColor

描述

在运行系统中无访问权限。

ButtonBorderColor

描述

在运行系统中无访问权限。

ButtonBorderWidth

描述

在运行系统中无访问权限。

ButtonCornerRadius

描述

在运行系统中无访问权限。

ButtonEdgeStyle

描述

在运行系统中无访问权限。

ButtonFirstGradientColor

描述

在运行系统中无访问权限。

ButtonFirstGradientOffset

描述

在运行系统中无访问权限。

ButtonMiddleGradientColor

描述

在运行系统中无访问权限。

ButtonPositions

说明

在运行系统中无访问权限。

ButtonSecondGradientColor

描述

在运行系统中无访问权限。

ButtonSecondGradientOffset

描述

在运行系统中无访问权限。

BV_ColumnWidth_Date

描述

在运行系统中无访问权限。

BV_ColumnWidth_Event

描述

在运行系统中无访问权限。

BV_ColumnWidth_EventSeverity

描述

在运行系统中无访问权限。

BV_ColumnWidth_EventState

描述

在运行系统中无访问权限。

BV_ColumnWidth_Number

描述

在运行系统中无访问权限。

BV_ColumnWidth_Time

描述

在运行系统中无访问权限。

BV_ItemText_Date

描述

在运行系统中无访问权限。

BV_ItemText_Event

描述

在运行系统中无访问权限。

BV_ItemText_EventSeverity

描述

在运行系统中无访问权限。

BV_ItemText_EventState

描述

在运行系统中无访问权限。

BV_ItemText_Number

描述

在运行系统中无访问权限。

BV_ItemText_Time

描述

在运行系统中无访问权限。

BV_ShowItem_Date

描述

在运行系统中无访问权限。

BV_ShowItem_Event

描述

在运行系统中无访问权限。

BV_ShowItem_EventSeverity

描述

在运行系统中无访问权限。

BV_ShowItem_EventState

描述

在运行系统中无访问权限。

BV_ShowItem_Number

描述

在运行系统中无访问权限。

BV_ShowItem_Time

描述

在运行系统中无访问权限。

1.5.5.3 属性 C

CameraUrl

描述

在运行系统中无访问权限。

CanBeGrouped

描述

在运行系统中无访问权限。

Caption

描述

指定标题中将要显示的文本。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**Caption**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTrendControl
- Slider *
- TrendRulerControl
- UserArchiveControl

* RT Advanced 读写访问权限

在运行系统中您没有以下格式的访问权限：

- OnlineTableControl

STRING

可选项。包含要在标题行中显示的文本的值或常量。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

Slider (页 485)

TrendRulerControl (页 532)

用户归档控件 (页 566)

CaptionBackColor

描述

指定标题栏的背景色。

运行系统中可进行读写访问

语法

Object.**CaptionBackColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- SymbolicIOField

Color

可选项。用于指定标题栏背景色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

SymbolicIOField (页 504)

CaptionColor

描述

指定将要在标题栏中显示文本的颜色。

运行系统中可进行读写访问

语法

Object.**CaptionColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Gauge
- Switch
- SymbolicIOField

Color

可选项。用于指定文本颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Gauge (页 366)

Switch (页 499)

SymbolicIOField (页 504)

CaptionFont

说明

设置字体。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**CaptionFont**[=Font]

Object

必需项。 “ScreenItem”对象，且具有以下格式：

- Gauge

在运行系统中您没有以下特性的访问权限：

- Switch

Font

可选项。 用于指定字体的值或常量。

参见

Gauge (页 366)

Switch (页 499)

CaptionText

说明

指定标题中将要显示的文本。

运行系统中可进行读写访问

语法

Object.**CaptionText**[= STRING]

Object

必选。具有以下格式的“ScreenItem”类型的对象：

- Gauge
- ScreenWindow
- Switch

STRING

可选项包含要在标题行中显示的文本的值或常量。

参见

Gauge (页 366)

ScreenWindow (页 479)

Switch (页 499)

CaptionTop

描述

指定仪器标签距对象上端的距离。仪器标签只能与刻度盘垂直方向的直径对齐。属性值参考指定对象的高度。该高度用于确定指定对象的上端和文字的下端。

运行系统中可进行读写访问

语法

Object.**CaptionTop**[=DOUBLE]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Gauge

DOUBLE

可选项。指定仪器标签距对象上端距离的值或常量。

值	说明
0	文字的下端与对象的上端对齐。文本位于所选对象范围之外时，文本将不可见。
1	文字的下端与所选对象的下端对齐。

参见

Gauge (页 366)

CellCut

描述

指定单元格过窄时是否缩略单元格内容。

运行系统中可进行读写访问

语法

Object.**CellCut**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl

在运行系统中您没有以下格式的访问权限：

- UserArchiveControl

BOOLEAN

可选项。

如果缩略内容，则为 TRUE。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

CellSpaceBottom

描述

指定表格单元格的下边距。

运行系统中的访问权限：读和写

语法

Object.**CellSpaceBottom**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于定义表格单元格中采用的下边距的值或常量。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

CellSpaceLeft

描述

指定表格单元格中采用的左缩进。

运行系统中的访问权限：读和写

语法

Object.**CellSpaceLeft**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于定义表格单元格中左缩进的值或参数。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

CellSpaceRight

描述

指定表格单元格中采用的右缩进。

运行系统中的访问权限：读和写

语法

Object.**CellSpaceRight**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于定义表格单元格中采用的右缩进的值或常量。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

CellSpaceTop

描述

定义表格单元格的上边距。

运行系统中的访问权限：读和写

语法

Object.**CellSpaceTop**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于定义表格单元格中采用的上边距的值或常量。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

CenterColor

描述

指定中心点的颜色。

运行系统中可进行读写访问

语法

Object.**CenterColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Gauge

Color

可选项。用于指定中心颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Gauge (页 366)

CenterSize

描述

指定刻度中心点的直径。

运行系统中的访问权限： 读和写

语法

Object.**CenterSize**[=SINGLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Gauge

SINGLE

可选项。用于指定圆盘刻度中心点直径的值或常量。

值范围为 0.03 到 1。

1: 直径对应于几何属性“Width”或“Height”的较小值。

参见

Gauge (页 366)

ChangeMouseCursor

描述

指定光标置于图标上方时，它在运行系统中的外观如何变化。

运行系统中可进行读写访问

语法

Object.**ChangeMouseCursor**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- SymbolLibrary

BOOLEAN

可选项。

如果将鼠标光标置于图标上方时显示为箭头，则为 TRUE。

如果鼠标光标显示为带绿色闪烁符号的 3D 箭头，则为 FALSE。这表明在运行系统中可以操作相应的对象。

参见

SymbolLibrary (页 511)

CheckMarkAlignment

说明

指定各域是否右对齐。

运行系统中的访问权限：读和写

语法

Object.**CheckMarkAlignment**[=CheckMarkAlignment]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- CheckBox
- OptionGroup

CheckMarkAlignment

可选项。用于指定字段是否为右对齐的值或常量。

值	说明
0	按左对齐排列域。
1	字段为右对齐。

参见

OptionGroup (页 435)

CheckBox (页 307)

CheckMarkCount

描述

指定字段数量。

运行系统中的访问权限：读和写

语法

Object.**CheckMarkCount**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- CheckBox
- OptionGroup

Int32

可选项。用于指定字段数量的值或常量。值范围为 0 到 31。

参见

OptionGroup (页 435)

CheckBox (页 307)

ClearOnError

描述

指定该对象的无效输入是否自动删除。

运行系统中的访问权限： 读和写

语法

Object.**ClearOnError**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- IOField

BOOLEAN

可选。若该对象的无效输入自动删除，则为 TRUE。

参见

IOField (页 382)

ClearOnFocus

描述

指定是否只要激活 I/O 域便删除域条目。

运行系统中的访问权限： 读和写

语法

Object.**ClearOnFocus**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- IOField

BOOLEAN

可选。如果只要 I/O 域激活就立即删除域条目，则为 TRUE。

参见

IOField (页 382)

Closeable

说明

指定是否可在运行系统中关闭对象。

运行系统中的访问权限：读和写

语法

Object.**Closeable**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl

- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选。如果对象可在运行系统中关闭则为 TRUE。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

Color

描述

指定线颜色。

运行系统中的访问权限：读和写

语法

Object.**Color**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- CircularArc
- Connector
- EllipticalArc
- Line
- Polyline

- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline

Color

可选项。指定线颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

CircularArc (页 320)

Connector (页 330)

EllipticalArc (页 348)

Line (页 388)

Polyline (页 448)

TubeArcObject (页 554)

TubeDoubleTeeObject (页 557)

TubePolyline (页 560)

TubeTeeObject (页 563)

ColorChangeHysteresis

描述

指定作为显示值百分数的滞后值。

"ColorChangeHysteresisEnable"属性的值必须为 TRUE 以便计算滞后值。

运行系统中的访问权限：读和写

语法

Object.**ColorChangeHysteresis**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

DOUBLE

可选项。用于指定作为显示值百分数的滞后值的值或常量。

参见

Bar (页 285)

ColorChangeHysteresisEnabled

描述

决定对象是否滞后显示。

运行系统中的访问权限：读和写

语法

Object.**ColorChangeHysteresisEnabled**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

BOOLEAN

可选。若对象滞后显示，则为 TRUE。

参见

Bar (页 285)

ColumnAdd

说明

创建一个新列。新创建的列会自动使用“ColumnIndex”而引用。

运行系统中可进行读写访问

语法

Object.**ColumnAdd**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- TrendRulerControl

STRING

必需项。用于指定新列名称的值或常量。

参见

ColumnIndex (页 734)

TrendRulerControl (页 532)

ColumnAlias

说明

返回使用“ColumnIndex”引用的列的显示名称。

运行系统中可进行读写访问

语法

Object.**ColumnAlias**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- UserArchiveControl

STRING

可选项。用于指定列显示名称的值或常量。

参见

用户归档控件 (页 566)

ColumnAlignment

描述

指定使用"ColumnIndex"引用的列内容的对齐方式。

运行系统中可进行读写访问

语法

Object.**ColumnAlignment**[=HorizontalAlignment]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- UserArchiveControl

HorizontalAlignment

可选项。确定水平对齐方式的值或常量。

值	名称	说明
0	左对齐	文本左对齐。
1	居中	文本居中。
2	右对齐	文本右对齐。

参见

用户归档控件 (页 566)

ColumnAutoPrecisions

说明

指定是否自动设置使用“ColumnIndex”引用的列的小数位数。

运行系统中可进行读写访问

语法

Object.**ColumnAutoPrecisions**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- UserArchiveControl

BOOLEAN

可选项。

如果自动确定小数位数，则为 TRUE。

如果使用“ColumnPrecisions”值，则为 FALSE。

参见

用户归档控件 (页 566)

ColumnCaption

说明

指定当前列的标题。

运行系统中的访问权限：读和写

语法

Object.**ColumnCaption**[=STRING]

Object

必需项。 “ScreenItem”对象，且具有以下格式：

- UserArchiveControl

STRING

可选项。 用于指定当前列的标题的值或常量。

参见

用户归档控件 (页 566)

ColumnCount

说明

指定所组态列的数目。

运行系统中可进行读写访问

语法

Object.**ColumnCount**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于指定已组态的列数的值或常量。

参见

TrendRulerControl (页 532)

用户归档控件 (页 566)

ColumnDateFormat

描述

指定日期格式。

运行系统中可进行读写访问

语法

Object.**ColumnDateFormat**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- UserArchiveControl

STRING

可选项。指定日期格式的值或常量。

参见

用户归档控件 (页 566)

ColumnDMVarName

说明

指定变量的名称。

运行系统中的访问权限：读和写

语法

Object.**ColumnDMVarName**[=STRING]

Object

必需项。"ScreenItem"对象，且具有以下格式：

- UserArchiveControl

STRING

可选项。用于指定变量名称的值或常量。

参见

用户归档控件 (页 566)

ColumnExponentialFormat

说明

指定是否以指数计数法显示使用"ColumnIndex"引用的列的值。

运行系统中可进行读写访问

语法

Object.**ColumnExponentialFormat**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- UserArchiveControl

BOOLEAN

可选项。

如果以指数记数法显示值，则为 TRUE。

如果不以指数记数法显示值，则为 FALSE。

参见

ColumnIndex (页 734)

用户归档控件 (页 566)

ColumnFlagNotNull

说明

指定分配给使用"ColumnIndex"引用的列的用户归档字段是否必须有一个值。

运行系统中可进行读写访问

语法

Object.**ColumnFlagNotNull**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- UserArchiveControl

BOOLEAN

可选项。

TRUE，需要值。

FALSE，不需要值。

参见

用户归档控件 (页 566)

ColumnFlagUnique

说明

指定分配给使用"ColumnIndex"引用的列的用户归档字段是否必须有唯一值。该列中的值必须各不相同。

运行系统中可进行读写访问

语法

Object.**ColumnFlagUnique**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- UserArchiveControl

BOOLEAN

可选项。

TRUE，该值必须唯一。

FALSE，该值不必唯一。

参见

用户归档控件 (页 566)

ColumnHideText

说明

指定是否隐藏使用“ColumnIndex”引用的列的文本。

运行系统中可进行读写访问

语法

Object.**ColumnHideText**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- UserArchiveControl

BOOLEAN

可选项。

TRUE，以隐藏当前列文本。

FALSE，以显示当前列文本。

参见

用户归档控件 (页 566)

ColumnHideTitleText

说明

指定是否隐藏使用“ColumnIndex”引用的列的标题。

运行系统中可进行读写访问

语法

Object.**ColumnHideTitleText**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- UserArchiveControl

BOOLEAN

可选项。

如果隐藏引用的列的标题，则为 TRUE。

如果显示引用列的标题，则为 FALSE。

参见

用户归档控件 (页 566)

ColumnIndex

说明

引用一个列。要访问列的属性，需要设置""ColumnIndex""。

介于 0 至 (ColumnIndex - 1) 之间的值为"ColumnCount"的有效值。"ColumnCount"属性会指定已组态列的数目。

运行系统中可进行读写访问

语法

Object.**ColumnIndex**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- TrendRulerControl
- UserArchiveControl

Int32

可选项。指定要编辑索引的列的值或常量。

参见

ColumnCount (页 729)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ColumnLeadingZeros

说明

指定使用"ColumnIndex"引用的列的值所显示的前导零数量。

运行系统中可进行读写访问

语法

Object.**ColumnLeadingZeros**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- UserArchiveControl

Int32

可选项。用于指定前导零数量的值或常量。最大数量为 11。

参见

用户归档控件 (页 566)

ColumnLength

说明

指定使用"ColumnIndex"引用的列中所显示的字符数量。

运行系统中可进行读写访问

语法

Object.**ColumnLength**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- UserArchiveControl

Int32

可选项。用于指定所显示字符数的值或常量。

参见

用户归档控件 (页 566)

ColumnMaxValue

说明

返回使用"ColumnIndex"引用的列的用户归档中所定义的最大值。

运行系统中可进行读写访问

语法

Object.**ColumnMaxValue**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- UserArchiveControl

STRING

可选项。用于返回用户日志指定列的最大值的值或常量。

参见

用户归档控件 (页 566)

ColumnMinValue

说明

返回使用"ColumnIndex"引用的列的用户归档中所定义的最小值。

运行系统中可进行读写访问

语法

Object.**ColumnMinValue**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- UserArchiveControl

STRING

可选项。用于返回用户日志指定列的最小值的值或常量。

参见

用户归档控件 (页 566)

ColumnName

说明

指定使用"ColumnIndex"引用的列的名称。

运行系统中可进行读写访问

语法

Object.**ColumnName**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- TrendRulerControl
- UserArchiveControl

STRING

可选项。可返回引用的列名称的值或常量。

参见

ColumnIndex (页 734)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ColumnOrder

说明

在运行系统中无访问权限。

ColumnPosition

说明

指定使用"ColumnIndex"引用的列的位置。

运行系统中可进行读写访问

语法

Object.**ColumnPosition**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- UserArchiveControl

Int32

可选项。用于指定列位置的值或常量。

参见

用户归档控件 (页 566)

ColumnPrecisions

说明

指定使用"ColumnIndex"引用的列中的小数位数。

运行系统中可进行读写访问

语法

Object.**ColumnPrecisions**[=Int16]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- UserArchiveControl

Int16

可选项。用于指定列中小数位数的值或常量。

参见

用户归档控件 (页 566)

ColumnReadAccess

说明

返回使用"ColumnIndex"引用的列的用户归档中所定义的读访问权限。

其编号与在"用户管理器"编辑器中分配给该权限的编号相对应。

运行系统中可进行读写访问

语法

Object.**ColumnReadAccess**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- UserArchiveControl

Int32

可选项。用于将用户日志中指定的读访问权限返回到该列的值或常量。

参见

用户归档控件 (页 566)

ColumnReadonly

说明

指定是否只可读取使用“ColumnIndex”引用的列的值。

运行系统中可进行读写访问

语法

Object.**ColumnReadonly**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- UserArchiveControl

BOOLEAN

可选项。

TRUE，当前列的值是只读值。

FALSE，当前列的值不是只读值。

参见

用户归档控件 (页 566)

ColumnRemove

说明

使用引用的列名称可移除该列。

运行系统中可进行读写访问

语法

Object.**ColumnRemove**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- TrendRulerControl

STRING

可选项。可返回要移除的引用的列名称的值或常量。

参见

TrendRulerControl (页 532)

ColumnRepos

说明

对于多个列，指定使用"ColumnIndex"引用的列的位置。

如果已使用"ColumnRepos"更改了列位置，"ColumnRepos"的值则会分配给"ColumnIndex"。

运行系统中可进行读写访问

语法

Object.**ColumnRepos**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于指定引用的列位置的值或常量。值范围为 0 到 (ColumnCount - 1)。超出该范围的值无效。

0：引用的列放置在左侧。

参见

ColumnIndex (页 734)

ColumnCount (页 729)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ColumnResize

描述

指定是否可以更改列宽。

运行系统中可进行读写访问

语法

Object.**ColumnResize**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

如果能更改列宽，则为 TRUE。

如果不能更改列宽，则为 FALSE。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

Columns

描述

在运行系统中无访问权限。

ColumnScrollbar

描述

指定水平滚动条显示的时间。

运行系统中可进行读写访问

语法

Object.**ColumnScrollbar**[=ScrollbarVisibility]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

ScrollbarVisibility

可选项。用于指定水平滚动条显示时间的值或常量。

值	说明
0	水平滚动条不显示。
1	如果显示区太小不足以显示所有列，则会显示水平滚动条。
2	始终显示水平滚动条。

参见

用户归档控件 (页 566)

TrendRulerControl (页 532)

OnlineTableControl (页 402)

AlarmControl (页 255)

ColumnSettings

描述

在运行系统中无访问权限。

ColumnSettingsBufferView

描述

在运行系统中无访问权限。

ColumnShowDate

说明

指定是否在使用"ColumnIndex"引用的列中显示日期。

运行系统中可进行读写访问

语法

Object.**ColumnShowDate**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- UserArchiveControl

BOOLEAN

可选项。

TRUE，在列中显示日期。

FALSE，不在列中显示日期。

参见

用户归档控件 (页 566)

ColumnShowIcon

说明

指定是否在使用"ColumnIndex"引用的列中显示符号。

运行系统中可进行读写访问

语法

Object.**ColumnShowIcon**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- UserArchiveControl

BOOLEAN

可选项。

TRUE，在列中显示图标。

FALSE，不在列中显示图标。

参见

用户归档控件 (页 566)

ColumnShowTitleIcon

说明

指定是否在使用"ColumnIndex"引用的列中显示符号。

运行系统中可进行读写访问

语法

Object.**ColumnShowTitleIcon**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- UserArchiveControl

BOOLEAN

可选项。

TRUE，在列中显示图标。

FALSE，不在列中显示图标。

参见

用户归档控件 (页 566)

ColumnsMoveable

描述

在运行系统中无访问权限。

ColumnSort

说明

指定使用"ColumnIndex"引用的列的排序类型。

运行系统中可进行读写访问

语法

Object.**ColumnSort**[=SortMode]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- TrendRulerControl
- UserArchiveControl

SortMode

可选项。用于指定排序模式的值或常量。

值	说明
0	无排序
1	从最小值到最大值的升序排列
2	从最大值到最小值的降序排列

参见

TrendRulerControl (页 532)

用户归档控件 (页 566)

ColumnSortIndex

说明

指定使用"ColumnIndex"引用的列的排序顺序，其中列会以一个接一个的方式排序。

运行系统中可进行读写访问

语法

Object.**ColumnSortIndex**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于指定列的排序索引的值或常量。

值	说明
0	将“ColumnSort (页 746)”属性中所定义的排序方向设置为“不排序”。
1	标题中显示编号 1。将先按此列排序。
2 到 4	所选编号显示在标题中。并按编号对应顺序对列进行排序。

参见

ColumnIndex (页 734)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ColumnStartValue

说明

返回使用"ColumnIndex"引用的列的用户日志中所定义的起始值。

运行系统中可进行读写访问

语法

Object.**ColumnStartValue**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- UserArchiveControl

STRING

可选项。用于返回用户日志指定列的起始值的值或常量。

参见

用户归档控件 (页 566)

ColumnStringLength

说明

返回使用"ColumnIndex"引用的列的用户归档中所定义的字符串长度。

运行系统中可进行读写访问

语法

Object.**ColumnStringLength**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- UserArchiveControl

Int32

可选项。用于返回用户日志中指定列的字符串长度的值或常量。

参见

用户归档控件 (页 566)

ColumnTextAckGroup

描述

在运行系统中无访问权限。

ColumnTextAlarmState

描述

在运行系统中无访问权限。

ColumnTextAlarmText

描述

在运行系统中无访问权限。

ColumnTextBit

说明

在运行系统中无访问权限。

ColumnTextClassName

描述

在运行系统中无访问权限。

ColumnTextConnection

说明

在运行系统中无访问权限。

ColumnTextDataType

说明

在运行系统中无访问权限。

ColumnTextDate

描述

在运行系统中无访问权限。

ColumnTextDateTime

说明

在运行系统中无访问权限。

ColumnTextDbNumber

说明

在运行系统中无访问权限。

ColumnTextDevice

描述

在运行系统中无访问权限。

ColumnTextDiagnosable

描述

在运行系统中无访问权限。

ColumnTextFormat

说明

在运行系统中无访问权限。

ColumnTextGroup

说明

在运行系统中无访问权限。

ColumnTextLogTime

说明

在运行系统中无访问权限。

ColumnTextNumber

描述

在运行系统中无访问权限。

ColumnTextOffset

说明

在运行系统中无访问权限。

ColumnTextPassword

说明

在运行系统中无访问权限。

ColumnTextRead

说明

在运行系统中无访问权限。

ColumnTextTagConnection

说明

在运行系统中无访问权限。

ColumnTextTime

描述

在运行系统中无访问权限。

ColumnTextTrend

说明

在运行系统中无访问权限。

ColumnTextType

说明

在运行系统中无访问权限。

ColumnTextUser

说明

在运行系统中无访问权限。

ColumnTextValue

说明

在运行系统中无访问权限。

ColumnTextWrite

说明

在运行系统中无访问权限。

ColumnTextXValue

说明

在运行系统中无访问权限。

ColumnTimeFormat

说明

指定使用“ColumnIndex”引用的列的时间格式。

运行系统中可进行读写访问

语法

Object.**ColumnTimeFormat**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- UserArchiveControl

STRING

可选项。用于指定列的时间格式的值或常量。

值	说明
自动	自动设置时间格式。
HH:mm:ss.ms	时:分:秒，例如，15:35:44.240。
hh:mm:ss tt	时:分:秒 AM/PM，例如 03:35:44 PM。
hh:mm:ss.ms tt	时:分:秒.毫秒 AM/PM，例如 03:35:44.240 PM。

参见

用户归档控件 (页 566)

ColumnTitleAlignment

描述

指定使用"ColumnIndex"引用的列标题的对齐方式。

运行系统中可进行读写访问

语法

Object.**ColumnTitleAlignment**[=GridColumnHeaderHorizontalAlignment]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl

- TrendRulerControl
- UserArchiveControl

GridColumnHeaderHorizontalAlignment

可选项。用于指定列标题对齐方式的值或常量。

值	名称	说明
0	左侧	列标题左对齐。
1	居中	列标题居中。
2	右侧	列标题右对齐
3	按照表格内容	列标题按照列内容的“ColumnAlignment (页 727)”属性中定义的方式对齐。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ColumnTitles

描述

指定是否显示列标题。

运行系统中的访问权限：读和写

语法

Object.**ColumnTitles**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- OnlineTableControl

- TrendRulerControl
- UserArchive Control

BOOLEAN

可选项。

如果显示列标题 则为 TRUE。

如果未显示列标题，则为 FALSE。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ColumnType

说明

返回使用“ColumnIndex”引用的列的用户归档中所定义的数据类型。

运行系统中可进行读写访问

语法

Object.**ColumnType**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- UserArchiveControl

Int32

可选项。返回用户日志中指定列的数据类型的值或常量。

参见

用户归档控件 (页 566)

ColumnVisible

说明

指定是否在对象中显示使用“ColumnIndex”引用的列。

运行系统中可进行读写访问

语法

Object.**ColumnVisible**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，显示引用的列。

FALSE，不显示引用的列。

参见

ColumnIndex (页 734)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ColumnWidth

说明

在运行系统中无访问权限。

ColumnWriteAccess

说明

返回使用"ColumnIndex"引用的列的用户归档中所定义的写访问权限。

其编号与在“用户管理器”编辑器中分配给该权限的编号相对应。

运行系统中可进行读写访问

语法

Object.**ColumnWriteAccess**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- UserArchiveControl

Int32

可选项。用于将用户日志中指定的写访问权限返回到该列的值或常量。

参见

用户归档控件 (页 566)

ComboboxFont

描述

在运行系统中无访问权限。

CompatibilityMode

描述

在运行系统中无访问权限。

ComplexViewToolbar

说明

在运行系统中无访问权限。

ComplexViewToolbarBounds

说明

在运行系统中无访问权限。

ComponentInfoText

说明

在运行系统中无访问权限。

ComputerName

说明

返回触发了报警对象的计算机的名称。

ComputerName（只读）

ConfiguredAlarmClasses

说明

在运行系统中无访问权限。

ConnectionType

描述

指定连接器类型。可以选择两种连接类型之一。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**ConnectionType**[=ConnectorConnectionType]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Connector

在运行系统中您没有以下格式的访问权限：

- SmartClientView

ConnectorConnectionType

可选项。用于指定连接器类型的值或常量。

(0)：自动：两个对象由水平和垂直部分构成的折线连接。

(1)：简单：两个对象由两个连接点之间的直线连接。字段为右对齐。

参见

Connector (页 330)

SmartClientView (页 490)

ConnectOnStart

描述

在运行系统中无访问权限。

ConnectTrendWindows

描述

指定是否连接已组态的趋势视图。前提是您已组态多个趋势视图。

所连接的趋势视图具有下列属性：

- 公共 X 轴
- 滚动条
- 标尺

运行系统中可进行读写访问

语法

Object.**ConnectTrendWindows**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

可选项。

TRUE，已连接所有已组态的趋势视图。

FALSE，单独显示各个趋势视图。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

Context

说明

读取或设置报警对象服务器前缀。

ContinousChange

描述

指定在运行系统中放开鼠标按钮时或只要滚动条位置改变时，是否传递“ProcessValue”属性的值。

运行系统中的访问权限：读和写

语法

Object.**ContinousChange**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Slider

BOOLEAN

可选。如果在运行系统中放开鼠标按钮时或只要滚动条位置改变时就传递“ProcessValue”属性的值，则为 TRUE。

参见

Slider (页 485)

ControlDesignMode

说明

指定控制设计。

运行系统中可进行读写访问

语法

Object.**ControlDesignMode**[=RTControlModes]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

RTControlModes

可选项。用于指定控件设计的值或常量。

值	名称	说明
	项目设置	此设计对应于 WinCC 项目管理器中的项目设置。
0	简单	WinCC 经典设计
1	标准	WinCC V7 最新设计

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

CornerRadius

描述

在运行系统中无访问权限。

CornerStyle

说明

指定边角的形状。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**CornerStyle**[=CornerStye | LineJoinStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- ComboBox
- Connector
- Ellipse
- EllipseSegment
- EllipticalArc
- GraphicIOField
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit

- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton
- SymbolicIOField
- TextField
- Tubepolyline
- WindowSlider

在运行系统中您没有以下格式的访问权限：

- Switch
- TubeArcObject

CornerStyle | LineJoinStyle

可选项。用于指定角形状的值或常量。

值	说明
0	实线
1	短划线
2	虚线
3	点划线
4	双点划线

Count

说明

返回指定列表元素的数量。

运行系统中可进行的访问：Read

语法

Object.Count

Object

要求“Collection”对象。

CountDivisions

描述

指定根据主要刻度标记长度可将栏分隔的段数量。

运行系统中的访问权限：读和写

语法

Object.**CountDivisions**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

Int32

可选项。用于根据主要刻度标记长度指定棒图分隔的段数量的值或常量。

0-100: 对象最多可分为 100 段。

= 0: 最佳的段数量自动确定。

参见

Bar (页 285)

CountOfLinesPerAlarms

描述

在运行系统中无访问权限。

CountOfVisibleAlarms

描述

在运行系统中无访问权限。

CountSubDivisions

描述

指定两个主要标记之间的刻度标记数。

运行系统中可进行读写访问

语法

Object.**CountSubDivisions**[= Int32]

Object

必需项。具有以下格式的"ScreenItem"对象：

- Bar

Int32

可选项。指定刻度段数量的值或常量。

参见

Bar (页 285)

CountVisibleItems

描述

指定选择列表将包含的行数。如果组态文本的数量大于该值，则选择列表将显示垂直滚动条。

运行系统中的访问权限：

- RT Advanced：读取
- RT Professional：读和写

语法

Object.**CountVisibleItems**[=Int32]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- ComboBox
- ListBox
- SymbolicIOField

在运行系统中您没有以下格式的访问权限：

- StatusForce
- TrendView
- UserView

Int32

可选项。用于指定选择列表将包含的行数的值或常量。

参见

ComboBox (页 326)

Listbox (页 392)

SymbolicIOField (页 504)

StatusForce (页 493)

TrendView (页 546)

UserView (页 581)

CurrentContext

描述

根据函数的使用方式返回字符串。

如果函数包含在画面窗口的画面中，CurrentContext 将返回提供此画面的符号服务器名称。
示例： "WinCCProject_MyComputer::"

如果函数包含在主画面中，则返回空字符串。

运行系统中可进行的访问：读

语法

Object.**CurrentContext**

Object
必选。“HMIRuntime”对象。

参见

HMIRuntime（页 224）

CursorControl

描述

指定鼠标光标离开某一域后是否按 TAB 顺序跳至下一域。

运行系统中的访问权限：读和写

语法

Object.**CursorControl**[=BOOLEAN]

Object
必需项。"ScreenItem" 对象，且具有以下格式：

- IOField
- SymbolicIOField

BOOLEAN
可选。如果鼠标光标在离开某一字段后按 TAB-顺序跳至下一字段，则为 TRUE,。

注释

为此，“CursorMode”属性必须设置为 TRUE。

参见

IOField (页 382)

SymbolicIOField (页 504)

Curves

描述

在运行系统中无访问权限。

1.5.5.4 属性 D

DangerRangeColor

描述

指定“Gauge”对象危险刻度范围的颜色。

“DangerRangeVisible”属性的值必须为 TRUE，以显示警告范围。

运行系统中可进行读写访问

语法

Object.**DangerRangeColor**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Gauge

Color

可选项。用于指定警告范围颜色的值或常量。

注释

可以使用“RGB”函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Gauge (页 366)

DangerRangeStart

描述

指定"Gauge"对象危险范围的起始刻度值。

"DangerRangeColor"属性的值必须为 TRUE，以显示警告范围。

运行系统中可进行读写访问

语法

Object.**DangerRangeStart**[=SINGLE]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Gauge

SINGLE

可选项。用于指定警告范围起始刻度值的值或常量。

注释

范围从"DangerRangeStart"开始至刻度终止。

参见

Gauge (页 366)

DangerRangeVisible

描述

指定是否显示"Gauge"对象的警告刻度范围。

运行系统中的访问权限：读和写

语法

Object.**DangerRangeVisible**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Gauge

BOOLEAN

可选。如果要显示警告刻度范围，则为 TRUE。

注释

指定“DangerRangeColor”属性的警告范围的颜色。

指定“DangerRangeStart”属性的警告范围的起始值。

参见

Gauge (页 366)

DataFormat

说明

返回显示格式。

运行系统中的访问权限：读和写

语法

Object.**DataFormat**[=IOFieldDataFormat]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- IOField

IOFieldDataFormat

可选项。用于返回显示格式的值或常量。

值	名称	说明
0	二进制	内容以“二进制”数据格式显示。
1	十进制	内容以“十进制”数据格式显示。
2	字符串	表示字符串。
3	十六进制	内容以“十六进制”数据格式显示。

参见

IOField (页 382)

DataLogs

说明

返回“DataLogs”类型对象。

运行系统中可进行的访问：Read

语法

Object.**DataLogs**

Object

要求“Logging”对象。

参见

Logging (页 229)

DataProviderGuid

说明

返回控件的 GUID。

运行系统中可进行的访问：Read

语法

Object.DataProviderGuid

对象

必选项。“UserArchiveControl”类型的对象。

示例

```
Dim objRecipeview
Set objRecipeview= HMIRuntime.Screens("Screen_1").ScreenItems("Recipeview_1")
SmartTags("Tag1") = objRecipeView.DataProviderGuid
```

参见

用户归档控件 (页 566)

DataRecordNameCaption

描述

在运行系统中无访问权限。

DataRecordNrCaption

描述

在运行系统中无访问权限。

DataSet

说明

返回“DataSet”类型对象。

运行系统中可进行的访问：Read

语法

Object.**DataSet**

对象

要求"Screen"对象。

参见

HMIRuntime（页 224）

Screen (页 231)

DataSource

说明

在运行系统中无访问权限。

DefaultFilterEom

说明

在运行系统中无访问权限。

DefaultHitListFilterEom

说明

在运行系统中无访问权限。

DefaultMsgFilterSQL

说明

指定用作报警过滤器默认值的 SQL 语句。

运行系统中的访问权限：读和写

语法

Object.**DefaultMsgFilterSQL**[=STRING]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- AlarmControl

STRING

可选项。 用于指定用作报警过滤器默认值的 SQL 语句的值或常量。

参见

AlarmControl (页 255)

DefaultSort

说明

指定排序类型。

运行系统中的访问权限：读和写

语法

Object.**DefaultSort**[=SortMode]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- AlarmControl

SortMode

可选项。 用于指定排序类型的值或常量。

值	说明
0	无排序
1	报警从最底行开始更新。
2	报警从最顶行开始更新。

参见

AlarmControl (页 255)

DefaultSort2

说明

指定排序类型。

运行系统中的访问权限：读和写

语法

Object.**DefaultSort2**[=SortMode]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- AlarmControl

SortMode

可选项。 用于指定排序类型的值或常量。

值	说明
0	无排序
1	报警从最底行开始更新。
2	报警从最顶行开始更新。

参见

AlarmControl (页 255)

DefaultSort2Column

说明

指定表条目的初始排序的列。

如果该值已分配，可根据以下列顺序对报警排序：

- 日期/时间/编号

如果以值的形式组态“报警文本”列，例如，根据以下列顺序对报警排序：

- 报警文本/日期/时间/编号

运行系统中可进行读写访问

语法

Object.**DefaultSort2Column**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl

STRING

可选项。用于指定表条目初始排序的列的名称的值或常量。

参见

AlarmControl (页 255)

DeviceStyle

说明

在运行系统中无访问权限。

DiagnosticsContext

说明

指定诊断环境。

运行系统中的访问权限：读和写

语法

Object.**DiagnosticsContext**[=STRING]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- AlarmControl

STRING

可选项。 用于指定诊断环境的值或常量。

参见

AlarmControl (页 255)

DiagramAreaHeight

说明

在运行系统中无访问权限。

DiagramAreaLeft

说明

在运行系统中无访问权限。

DiagramAreaTop

说明

在运行系统中无访问权限。

DiagramAreaWidth

说明

在运行系统中无访问权限。

DialColor

描述

确定所选对象的对话框的颜色。

运行系统中可进行读写访问

语法

Object.**DialColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Clock
- Gauge

Color

可选项。用于指定对话框颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Clock (页 323)

Gauge (页 366)

DialFillStyle

描述

指定背景类型。

运行系统中可进行读写访问

语法

Object.**DialFillStyle**[=GaugeFillStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Gauge

GaugeFillStyle

可选项。用于指定背景类型的值或常量。

值	VB 常量	说明
0	hmiBackStyleSolid	显示的矩形背景以指定的边框颜色填充。刻度装置填充有指定的背景色。
1	hmiBackStyleFrameTransparent	矩形背景是透明的。刻度装置填充有指定的背景色。因此，呈现圆形显示。
2	hmiBackStyleTransparent	矩形背景和刻度是透明的。

参见

Gauge (页 366)

DialPicture

说明

指定表盘表面的图形。

运行系统中的访问权限：读和写

语法

Object.**DialPicture**[=HmiObjectHandle]

Object

必需项。 "ScreenItem"对象，且具有以下特性：

- Gauge

HmiObjectHandle

可选项。 用于指定表盘表面图形的值或常量。

参见

Gauge (页 366)

DialSize

描述

指定与几何属性"Width"或"Height"的较小值相关的刻度装置的直径。

运行系统中的访问权限：读和写

语法

Object.**DialSize**[=SINGLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Gauge

SINGLE

可选项。用于指定与几何属性"Width"和"Height"的较小值相关的标尺直径的值或常量。

参见

Gauge (页 366)

Display3D

说明

在运行系统中无访问权限。

DisplayButton2Plc

描述

在运行系统中无访问权限。

DisplayButtonComparison

描述

在运行系统中无访问权限。

DisplayButtonDelete

描述

在运行系统中无访问权限。

DisplayButtonFromPlc

描述

在运行系统中无访问权限。

DisplayButtonHelp

描述

在运行系统中无访问权限。

DisplayButtonNew

描述

在运行系统中无访问权限。

DisplayButtonSave

描述

在运行系统中无访问权限。

DisplayButtonSaveAs

描述

在运行系统中无访问权限。

DisplayCentury

说明

在运行系统中无访问权限。

DisplayComboBox

描述

在运行系统中无访问权限。

DisplayGridLines

描述

在运行系统中无访问权限。

DisplayLabeling

描述

在运行系统中无访问权限。

DisplayNumbers

描述

在运行系统中无访问权限。

DisplayOptions

描述

指定是否显示视图被抑制的报警。

运行系统中可进行读写访问

语法

Object.**DisplayOptions**[=AlarmDisplayOptions]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

AlarmDisplayOptions

可选项。用于指定报警是隐藏还是显示的值或常量。

值	说明
0	所有消息
1	仅显示的消息
2	仅隐藏的消息

参见

AlarmControl (页 255)

DisplaySize

说明

在运行系统中无访问权限。

DisplayStatusBar

描述

在运行系统中无访问权限。

DisplaySystemTime

说明

在运行系统中无访问权限。

DisplayTable

描述

在运行系统中无访问权限。

DoubleClickAction

描述

指定双击消息行时将在运行系统中执行的动作。

运行系统中可进行读写访问

语法

Object.**DoubleClickAction**[=AlarmControlActions]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

AlarmControlActions

可选项。指定双击报警行时将在运行系统中执行的动作。

值	描述	说明
0	无	无动作。
1	报警循环	调用“报警回路”功能。
2	打开注释对话框	调用“注释对话框”按钮功能。
3	打开信息文本对话框	调用“信息文本对话框”按钮功能。
4	列相关	该操作由双击的列确定。

参见

AlarmControl (页 255)

DrawInsideFrame

描述

指定要显示的线宽大于 1 边框线是在边框内描绘，还是与边框对称。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**DrawInsideFrame**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Ellipse
- EllipseSegment
- EllipticalArc
- GraphicIOField
- OptionGroup
- Rectangle
- RoundButton
- SymbolicIOField

- TextField
- WindowSlider

在运行系统中您没有以下格式的访问权限：

- Switch
- TubeArcObject

BOOLEAN

可选。如果显示的线宽大于 1 的边框线在边框内，则为 TRUE。

Drive

描述

在运行系统中无访问权限。

1.5.5.5 属性 E-F

EdgeStyle

描述

指定线样式。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**EdgeStyle**[=LineStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox

- Circle *
- CircleSegment
- ComboBox
- Ellipse *
- EllipseSegment
- GraphicIOField
- GraphicView
- IOField *
- ListBox
- MultiLineEdit
- OptionGroup
- Polygon *
- Rectangle *
- RoundButton
- SymbolicIOField
- TextField *
- WindowSlider

* RT Advanced 读写访问权限

在运行系统中您没有以下格式的访问权限：

- AlarmView
- Clock
- DateTimeField
- Gauge
- RecipeView
- Slider
- StatusForce
- Switch
- SysDiagControl

- TrendView
- UserView

LineStyle

可选项。用于指定线样式的值或常量。值范围为 -1 到 4。

对象“Ellipse”、“Circle”、“Rectangle”和“Polygon”支持线样式：

值	VB 常量	说明
0	hmiLineStyleSolid	实线
1	hmiLineStyleDash	短划线
2	hmiLineStyleDot	虚线
3	hmiLineStyleDashDot	点划线
4	hmiLineStyleDashDotDot	双点划线

对象“TextField”“和“IOField”仅支持线样式：

值	VB 常量	说明
-1	hmiLineStyleNone	无线
0	hmiLineStyleSolid	实线

参见

Bar (页 285)

Button (页 296)

CheckBox (页 307)

Circle (页 312)

CircleSegment (页 316)

ComboBox (页 326)

Ellipse (页 340)

EllipseSegment (页 344)

GraphicIOField (页 371)

GraphicView (页 375)

IOField (页 382)

Listbox (页 392)

MultiLineEdit (页 399)

OptionGroup (页 435)

Polygon (页 444)

Rectangle (页 467)

RoundButton (页 471)

SymbolicIOField (页 504)

TextField (页 527)

WindowSlider (页 586)

AlarmView (页 274)

DateTimeField (页 334)

Gauge (页 366)

RecipeView (页 458)

Slider (页 485)

StatusForce (页 493)

Switch (页 499)

SysDiagControl (页 514)

TrendView (页 546)

UserView (页 581)

EditOnFocus

描述

指定若使用 <Tab> 键选择输入域后是否可以立即输入数据。

运行系统中的访问权限： 读和写

语法

Object.**EditOnFocus**[=BOOLEAN]

Object

必需项。 "ScreenItem" 对象，且具有以下格式：

- IOField
- SymbolicIOField

BOOLEAN

可选。若使用 <Tab> 键选择输入域后可以立即输入数据，则为 TRUE。

参见

IOField (页 382)

SymbolicIOField (页 504)

Enabled

描述

指定是否可以在运行系统中操作所选对象。

运行系统中的访问权限：读和写

语法

Object.**Enabled**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl *
- AlarmView **
- Bar
- Button
- ChannelDiagnose *
- CheckBox *
- Circle
- CircleSegment *
- CircularArc *

- Clock
- ComboBox *
- Connector *
- DateTimeField **
- DiscSpaceView *
- Ellipse
- EllipseSegment *
- EllipticalArc *
- FunctionTrendControl **
- Gauge
- GraphicIOField
- GraphicView
- HTMLBrowser
- IOField
- Line
- ListBox *
- MediaPlayer
- MultiLineEdit *
- OptionGroup *
- PLCCodeViewer **
- PdfView **
- Polygon
- Polyline
- ProDiagOverview ***
- RecipeView **
- Rectangle
- RoundButton *
- S7GraphOverview **
- Slider

- SmartClientView **
- StatusForce **
- Switch **
- SymbolLibrary
- SymbolicIOField
- SysDiagControl ****
- TextField
- TrendRulerControl *
- TrendView **
- TubeArcObject *
- TubeDoubleTeeObject *
- TubeTeeObject *
- Tubepolyline *
- UserView
- WindowSlider *

* RT Advanced 无访问权限

** RT Professional 无访问权限

*** RT Advanced 读访问权限

**** 只读访问权限

在运行系统中您没有以下格式的访问权限：

- OnlineTableControl
- OnlineTrendControl
- UserArchiveControl

BOOLEAN

可选。如果可在运行系统中操作指定对象，则为 T RUE。

参见

AlarmControl (页 255)

AlarmView (页 274)

Bar (页 285)

Button (页 296)

ChannelDiagnose (页 305)

CheckBox (页 307)

Circle (页 312)

CircleSegment (页 316)

CircularArc (页 320)

Clock (页 323)

ComboBox (页 326)

DateTimeField (页 334)

DiskSpaceView (页 337)

Ellipse (页 340)

EllipseSegment (页 344)

EllipticalArc (页 348)

FunctionTrendControl (页 351)

Gauge (页 366)

GraphicIOField (页 371)

GraphicView (页 375)

HTMLBrowser (页 379)

IOField (页 382)

Line (页 388)

Listbox (页 392)

MediaPlayer (页 396)

MultiLineEdit (页 399)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

EnableDelete

说明

指定在运行系统中是否可以删除数据。

运行系统中的访问权限：读和写

语法

Object.**EnableDelete**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- UserArchiveControl

BOOLEAN

可选项。

TRUE，在运行系统中可以删除数据。

FALSE，在运行系统中不可删除数据。

参见

用户归档控件 (页 566)

EnableEdit

描述

指定是否可以在运行系统中编辑显示的数据。

运行系统中可进行读写访问

语法

Object.**EnableEdit**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，可在运行系统中更改数据。

FALSE，不可在运行系统中更改数据。

参见

OnlineTableControl (页 402)

用户归档控件 (页 566)

EnableInsert

说明

指定在运行系统中是否可以插入数据。

运行系统中的访问权限：读和写

语法

Object.**EnableInsert**[=BOOLEAN]

Object

必需项。 "ScreenItem"对象，且具有以下特性：

- UserArchiveControl

BOOLEAN

可选项。

TRUE，在运行系统中可以插入数据。

FALSE，在运行系统中不可插入数据。

参见

用户归档控件 (页 566)

EnableNavigateButtons

说明

在运行系统中无访问权限。

EnableNavigateKeys

描述

在运行系统中无访问权限。

EncryptCommunication

说明

在运行系统中无访问权限。

EndAngle

描述

指定端点偏离零位置 (0°) 的角度。

运行系统中可进行读写访问

语法

Object.**EndAngle**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- CircleSegment
- CircularArc

- EllipseSegment
- EllipticalArc
- TubeArcObject

Int32

可选项。用于指定所选对象端点偏离零位置 (0°) 角度的值或常量。

参见

CircleSegment (页 316)

CircularArc (页 320)

EllipseSegment (页 344)

EllipticalArc (页 348)

TubeArcObject (页 554)

EndLeft

描述

在运行系统中无访问权限。

EndStyle

描述

指定如何显示线端。

运行系统中的访问权限：读和写

语法

Object.**EndStyle**[=LineEndStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Connector
- Line
- Polyline

LineEndStyle

可选项。用于指定线端形状的值或常量。

值	VB 常量	说明
0	hmiLineEndStyleNone	没有线端符号。
65536*	hmiLineEndStyleArrow	线端为空心箭头。
131072	hmiLineEndStyleFilledArrow	线端为实心箭头。
196608*	hmiLineEndStyleFilledArrowReversed	线端为反向箭头。
262144*	hmiLineEndStyleLine	线端为垂直线。
327680*	hmiLineEndStyleCircle	线端为圆。
393216*	hmiLineEndStyleFilledCircle	线端为实心圆。

* 仅限 RT Professional

参见

Line (页 388)

Polyline (页 448)

Connector (页 330)

EndTop

描述

在运行系统中无访问权限。

EnterButtonVisible

说明

在运行系统中无访问权限。

EntryNameCaption

描述

在运行系统中无访问权限。

EntryNameColumnWidth

说明

在运行系统中无访问权限。

EntryValueColFirst

说明

在运行系统中无访问权限。

EntryValueColumnWidth

说明

在运行系统中无访问权限。

EntryValueFieldLength

说明

在运行系统中无访问权限。

EntryValuePos

说明

在运行系统中无访问权限。

ErrorColor

说明

指定 GRAPH 概览中错误的颜色。

运行系统中的访问权限：读和写

语法

Object.ErrorColor[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- S7GraphOverview

Color

可选项。用于指定错误颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

如果"BorderStyle"属性的值为"0"，则背景色不可见。

参见

S7GraphOverview (页 476)

ErrorDescription

说明

返回下列错误描述（英文）之一：

输出	说明
" "	正常
"Operation Failed"	执行错误
"Variable not found"	变量错误
"Server down"	服务器不可用。
"An error occured for one or several tags"	多个变量错误（一个或多个变量错误）

为了获得错误描述，首先执行 Read 方法。

说明

若通过 TagSet 对象访问时发生错误，则判断 TagSet 对象各个变量的 ErrorDescription 属性。

运行系统中可进行的访问：Read

语法

Object.**ErrorDescription**

Object

要求“Tag”对象。

示例

下列示例显示的是“Tag1”变量的错误描述：

```
'VBS72
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objtag.Read
MsgBox objTag.ErrorDescription
```

下列示例显示的是向 TagSet 列表添加两个变量，将 ErrorDescription 属性作为 Trace 输出：

```
'VBS179
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
HMIRuntime.Trace "ErrorDescription: " & group.ErrorDescription & vbNewLine
```

按如下步骤可以访问列表中变量的 ErrorDescription 属性：

```
HMIRuntime.Trace "ErrorDescription: " & group("Motor1").ErrorDescription & vbNewLine
```

参见

变量 (页 244)

TagSet（列表） (页 248)

Errorflag

说明

指定是否在设备/详细信息视图中显示错误描述。

运行系统中可进行读写访问

语法

Object.**Errorflag**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- S7GraphOverview

BOOLEAN

可选项。

TRUE，在设备/详细视图中显示错误描述。

FALSE，在设备/详细视图中不显示任何错误描述。

参见

S7GraphOverview (页 476)

ES2RT_ButtonPositions

说明

在运行系统中无访问权限。

ES2RT_ColumnOrder

说明

在运行系统中无访问权限。

ES2RT_ColumnWidth

说明

在运行系统中无访问权限。

ES2RT_EntryNameColumnWidth

说明

在运行系统中无访问权限。

ES2RT_EntryValueColumnWidth

说明

在运行系统中无访问权限。

ES2RT_ListAreaHeight

说明

在运行系统中无访问权限。

ES2RT_ListAreaWidth

说明

在运行系统中无访问权限。

ES2RT_MessageAreaHeight

说明

在运行系统中无访问权限。

ES2RT_MessageAreaWidth

说明

在运行系统中无访问权限。

ES2RT_StoreAsCheckBack

说明

在运行系统中无访问权限。

Es2rtButtonPositions

说明

在运行系统中无访问权限。

Es2rtTableBounds

说明

在运行系统中无访问权限。

EscButtonVisible

说明

在运行系统中无访问权限。

EsPreviewType

说明

在运行系统中无访问权限。

EvenRowBackColor

描述

在运行系统中无访问权限。

ExportDelimiter

描述

在运行系统中无访问权限。

ExportDirectoryChangeable

描述

指定是否可以在运行系统中更改数据导出目录。

运行系统中的访问权限：读和写

语法

Object.**ExportDirectoryChangeable**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

如果数据导出目录可在运行系统中进行更改，则为 TRUE。

如果数据导出目录不可在运行系统中进行更改，则为 FALSE。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ExportDirectoryname

描述

定义所导出运行系统数据写入的目录。

运行系统中的访问权限：读和写

语法

Object.**ExportDirectoryname**[=STRING]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

可选项。用于指定目录的值或常量。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ExportFileExtension

描述

指定导出文件的文件扩展名。目前仅支持“csv”文件扩展名。

运行系统中的访问权限：读和写

语法

Object.**ExportFileExtension**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

可选项。指定导出文件的文件扩展名。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ExportFilename

描述

指定导出的运行系统数据的目标文件名称。

运行系统中的访问权限：读和写

语法

Object.**ExportFilename**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

可选项。指定导出的运行系统数据的目标文件名称。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ExportFilenameChangeable

描述

指定是否可以在运行系统中更改导出文件名。

运行系统中的访问权限：读和写

语法

Object.**ExportFilenameChangeable**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

如果导出文件可以在运行系统中进行重命名，则为 TRUE。

如果导出文件不能在运行系统中进行重命名，则为 FALSE。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ExportFormat

说明

在运行系统中无访问权限。

ExportFormatGuid

描述

指定 ID 编号和导出数据源的分配。

运行系统中的访问权限：读和写

语法

Object.**ExportFormatGuid**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

可选项。用于指定 ID 编号和导出数据源分配的值或常量。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ExportFormatName

描述

定义导出文件格式。 当前只能导出"csv"格式的文件。

运行系统中的访问权限：读和写

语法

Object.**ExportFormatName**[=STRING]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

可选项。用于定义导出文件格式的值或常量。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ExportParameters

描述

通过属性对话框指定所选格式的参数。

运行系统中可进行读写访问

语法

Object.**ExportParameters**

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

ExportParameters

可选项。用于指定 "属性"(Properties) 对话框中所选格式参数的值或常量。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ExportSelection

说明

指定导出控件的哪些运行系统数据。

运行系统中的访问权限：读和写

语法

Object.**ExportSelection**[=ExportRange]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

ExportRange

可选项。用于指定可导出控件的运行系统数据的值或常量。

值	说明	解释
0	全部	导出控件的所有运行系统数据。
1	选择	导出选定的控件运行系统数据。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ExportShowDialog

描述

启用在运行期间导出对话框的显示。

运行系统中的访问权限：读和写

语法

Object.**ExportShowDialog**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选。如果在运行系统中显示对话框，则为 TRUE。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ExtendedZoomingEnable

说明

指定操作员是否可以使用鼠标滚轮在运行系统中放大或缩小画面。

运行中可进行的访问：读和写

语法

Object.**ExtendedZoomingEnable**[=BOOLEAN]

Object

必需。 “Screen”对象。

BOOLEAN

可选项。操作员可在运行系统中放大和缩小画面时为 TRUE。

示例

以下示例显示的是对于 NewPDL1 画面如何启用扩展缩放：

```
'VBS155
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
objScreen.ExtendedZoomingEnable = 1
```

参见

Screen (页 231)

ExtraHeightOffset

说明

在运行系统中无访问权限。

FieldLength

描述

在运行系统中无访问权限。

FileName

描述

指定要加载的文件的名称。

运行系统中的访问权限：

- RT Advanced: 无访问权限
- RT Professional: 读写访问权限

语法

Object.**FileName**[=STRING]

Object

必需项。"ScreenItem"对象，且具有以下特性：

- MediaPlayer

在运行系统中您没有以下格式的访问权限：

- PdfView

STRING

可选项。用于指定要加载的文件名称的值或常量。

参见

MediaPlayer (页 396)

FillColorMode

描述

指定所选对象的前景类型。

运行系统中可进行读写访问

语法

Object.**FillColorMode**[=SymbolLibraryColorMode]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- SymbolLibrary

SymbolLibraryColorMode

可选项。用于指定对象背景类型的值或常量。

值	VB 常量	说明
0	hmiSymbolLibraryAppearanceOriginal	表面呈灰色。
1	hmiSymbolLibraryAppearanceShaded	显示呈现阴影。
2	hmiSymbolLibraryAppearanceSolid	显示为实心。
3	hmiSymbolLibraryAppearanceTransparent	灰色显示。

参见

SymbolLibrary (页 511)

FillPattern

说明

在运行系统中无访问权限。

FillPatternColor

描述

指定填充图案的颜色。

运行系统中的访问权限

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**FillPatternColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- Ellipse
- EllipseSegment
- GraphicView
- IOField
- OptionGroup
- Polygon
- Rectangle
- RoundButton
- SymbolicIOField*
- TextField
- WindowSlider

*：只读访问权限

在运行系统中您没有以下格式的访问权限：

- AlarmControl
- ComboBox
- DateTimeField
- ListBox
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

可选项。用于指定填充图案的颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Bar (页 285)

Button (页 296)

CheckBox (页 307)

Circle (页 312)

CircleSegment (页 316)

ComboBox (页 326)

Ellipse (页 340)

EllipseSegment (页 344)

GraphicView (页 375)

IOField (页 382)

Listbox (页 392)

OptionGroup (页 435)

Polygon (页 444)

Rectangle (页 467)

RoundButton (页 471)

SymbolicIOField (页 504)

TextField (页 527)

WindowSlider (页 586)

AlarmControl (页 255)

DateTimeField (页 334)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

FillStyle

描述

指定是否以虚线显示背景颜色。

运行系统中可进行读写访问

语法

Object.**FillStyle**[=LineFillStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Line
- Polyline

LineFillStyle

可选项。用于指定填充样式的值或常量。

值	VB 常量	说明
0	hmiLineFillStyleTransparent	透明背景
1	hmiLineFillStyleSolid	使用指定颜色填充对象背景。

参见

Line (页 388)

Polyline (页 448)

FillingDirection

说明

指定填充方向。

运行系统中的访问权限：读和写

语法

Object.**FillingDirection**[=FillDirection]

Object

必选项。具有以下格式的"ScreenItem"类型的对象：

- Button
- CheckBox
- Circle
- CircleSegment
- Ellipse
- EllipseSegment
- GraphicView
- OptionGroup
- Polygon
- Rectangle
- RoundButton
- TextField
- WindowSlider

FillDirection

可选项。用于指定填充方向的值或常量。

Filter

描述

在运行系统中无访问权限。

FilterSQL

说明

针对过滤条件设置 SQL 语句。

运行系统中的访问权限：读和写

语法

Object.**FilterSQL**[=STRING]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- UserArchiveControl

STRING

可选项。 用于指定过滤条件对应的 SQL 语句的值或常量。

参见

用户归档控件 (页 566)

FilterTag

描述

在运行系统中无访问权限。

FilterText

描述

在运行系统中无访问权限。

FirstConnectedObject

描述

在运行系统中无访问权限。

FirstConnectedObjectIndex

说明

指定连接器上部点的索引值。

运行系统中的访问权限：读和写

语法

Object.**FirstConnectedObjectIndex**[=Int]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Connector

Int

可选项。用于指定连接器上部点的索引号的值或常量。

参见

Connector (页 330)

FirstConnectedObjectName

说明

指定连接于连接器上部点的对象的名称。

运行系统中的访问权限：读和写

语法

Object.**FirstConnectedObjectName**[=String]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Connector

String

可选项。用于指定连接于连接器上部点的对象名称的值或常量。

参见

Connector (页 330)

FirstGradientColor

说明

在运行系统中无访问权限。

FirstGradientOffset

说明

在运行系统中无访问权限。

FitToLargest

描述

在运行系统中无访问权限。

FitToSize

描述

在运行系统中无访问权限。

FitToSizeLowerRows

说明

在运行系统中无访问权限。

FitToSizeUpperRows

说明

在运行系统中无访问权限。

FixedAspectRatio

描述

指定当符号大小改变时是否应保持还是改变纵横比。

运行系统中的访问权限：读和写

语法

Object.**FixedAspectRatio**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- SymbolLibrary

BOOLEAN

可选。若符号大小改变时应保持纵横比，则为 TRUE。

参见

SymbolLibrary (页 511)

Flashing

描述

指定所选对象是否在运行系统中闪烁。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**Flashing**[=FlashingType]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Clock
- Gauge
- Slider

在运行系统中您没有以下格式的访问权限：

- AlarmView
- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Connector
- DateTimeField
- Ellipse
- EllipseSegment
- EllipticalArc
- GraphicIOField

- GraphicView
- HTMLBrowser
- IOField
- Line
- OptionGroup
- Polygon
- Polyline
- RecipeView
- Rectangle
- RoundButton
- ScreenWindow
- SmartClientView
- StatusForce
- Switch
- SymbolicIOField
- SymbolLibrary
- SysDiagControl
- TextField
- TrendView
- TubeArcObject
- UserView
- WindowSlider

FlashingType

可选项。用于指定对象是否在运行系统中闪烁的值或常量。

名称	说明
无	对象在运行系统中不闪烁。
标准	该对象在运行系统中闪烁。

FlashingColorOff

描述

指定闪烁状态“关闭”时的边框线颜色。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**FlashingColorOff**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Connector
- Ellipse
- EllipseSegment
- EllipticalArc
- IOField
- Line
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton

- SymbolicIOField*
- TextField

*：只读访问权限

在运行系统中您没有以下格式的访问权限：

- Bar
- Switch
- TubeArcObject

Color

可选项。用于指定闪烁状态“关闭”时边框线颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Button (页 296)

Circle (页 312)

CircleSegment (页 316)

CircularArc (页 320)

Ellipse (页 340)

EllipseSegment (页 344)

EllipticalArc (页 348)

IOField (页 382)

Line (页 388)

OptionGroup (页 435)

Polygon (页 444)

Polyline (页 448)

Rectangle (页 467)

RoundButton (页 471)

SymbolicIOField (页 504)

TextField (页 527)

Bar (页 285)

CheckBox (页 307)

Switch (页 499)

TubeArcObject (页 554)

FlashingColorOn

描述

指定闪烁状态“开启”时的边框线颜色。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**FlashingColorOn**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Connector
- Ellipse

- EllipseSegment
- EllipticalArc
- GraphicIOField
- IOField
- Line
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton
- SymbolicIOField*
- TextField

*：只读访问权限

在运行系统中您没有以下格式的访问权限：

- Bar
- Switch
- TubeArcObject

Color

可选项。用于指定闪烁状态“开启”时边框线颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Button (页 296)

Circle (页 312)

CircleSegment (页 316)

CircularArc (页 320)

Ellipse (页 340)

EllipseSegment (页 344)

EllipticalArc (页 348)

IOField (页 382)

Line (页 388)

OptionGroup (页 435)

Polygon (页 444)

Polyline (页 448)

Rectangle (页 467)

RoundButton (页 471)

SymbolicIOField (页 504)

TextField (页 527)

Bar (页 285)

CheckBox (页 307)

Switch (页 499)

TubeArcObject (页 554)

FlashingEnabled

说明

指定是否为运行系统中的对象激活闪烁。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**FlashingEnabled**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Connector
- Ellipse
- EllipseSegment
- EllipticalArc
- GraphicIOField
- IOField
- Line
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton
- SymbolicIOField*
- TextField

*：只读访问权限

在运行系统中您没有以下格式的访问权限：

- Bar
- Switch
- TubeArcObject

BOO LEAN

可选项。

如果为运行系统中的对象激活闪烁，则为 TRUE。

如果没有为运行系统中的对象激活闪烁，则为 FALSE。

参见

Button (页 296)

Circle (页 312)

CircleSegment (页 316)

CircularArc (页 320)

Ellipse (页 340)

EllipseSegment (页 344)

EllipticalArc (页 348)

IOField (页 382)

Line (页 388)

OptionGroup (页 435)

Polygon (页 444)

Polyline (页 448)

Rectangle (页 467)

RoundButton (页 471)

SymbolicIOField (页 504)

TextField (页 527)

Bar (页 285)

CheckBox (页 307)

Switch (页 499)

TubeArcObject (页 554)

FlashingOnLimitViolation

描述

在运行系统中无访问权限。

FlashingRate

说明

指定边界线的闪烁频率。

运行系统中可进行的访问：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**FlashingRate**[=FlashingRate]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Connector
- Ellipse
- EllipseSegment
- EllipticalArc
- GraphicIOField
- IOField
- Line

- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton
- SymbolicIOField*
- TextField

* 只读访问权限

在运行系统中您没有以下格式的访问权限：

- Bar
- Switch
- TubeArcObject

FlashingRate

可选项。用于指定边框线的闪烁速率的值或常量。

值	VB 常量	说明
0	hmiFlashingRateSlow	闪烁间隔为 1000 ms。
1	hmiFlashingRateMedium	闪烁间隔为 500 ms。
2	hmiFlashingRateFast	闪烁间隔为 250 ms。

参见

Button (页 296)

Circle (页 312)

CircleSegment (页 316)

CircularArc (页 320)

Ellipse (页 340)

EllipseSegment (页 344)

EllipticalArc (页 348)

IOField (页 382)

Line (页 388)

OptionGroup (页 435)

Polygon (页 444)

Polyline (页 448)

Rectangle (页 467)

RoundButton (页 471)

SymbolicIOField (页 504)

TextField (页 527)

Bar (页 285)

CheckBox (页 307)

Switch (页 499)

TubeArcObject (页 554)

FlashTransparentColor

说明

指定设置为“透明”的闪烁图形其位图对象的颜色。

运行系统中的访问权限：读和写

语法

Object.**FlashTransparentColor**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- GraphicIOField

Color

可选项。用于指定被设为“透明”的闪烁图形的位图对象颜色的值或常量。

注释

可以使用“RGB”函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

如果"BorderStyle"属性的值为"0"，则背景色不可见。

参见

GraphicIOField (页 371)

Flip

描述

指定是否沿符号的垂直和/或水平中心轴翻转符号。

运行系统中可进行读写访问

语法

Object.**Flip**[=SymbolLibraryFlip]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- SymbolLibrary

SymbolLibraryFlip

可选项。指定是否沿符号的垂直和/或水平中心轴翻转符号的值或常量。

值	VB 常量	说明
0	hmiSymbolLibraryFlipNone	符号未翻转。
1	hmiSymbolLibraryFlipHorizontal	符号水平翻转。
2	hmiSymbolLibraryFlipVertical	符号垂直翻转。
3	hmiSymbolLibraryFlipBoth	符号水平与垂直翻转。

参见

SymbolLibrary (页 511)

FocusColor

描述

当对象获得焦点时，指定焦点边框的颜色。

运行系统中的访问权限：

- RT Advanced：读和写
- RT Professional：无访问权

语法

Object.**FocusColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmView
- Button
- GraphicIOField
- RecipeView
- Slider *
- StatusForce
- Switch
- TrendView

* RT Professional 读写访问权限

在运行系统中您没有以下格式的访问权限：

- SymbolicIOField

Color

可选项。用于指定焦点边框颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmView (页 274)

RecipeView (页 458)

Slider (页 485)

StatusForce (页 493)

Switch (页 499)

TrendView (页 546)

Button (页 296)

GraphicIOField (页 371)

SymbolicIOField (页 504)

FocusWidth

描述

当对象获得焦点时，指定边框宽度。

运行系统中的访问权限：

- RT Advanced：读和写
- RT Professional：无访问权

语法

Object.**FocusWidth**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmView
- Button
- GraphicIOField
- RecipeView
- Slider *
- StatusForce
- Switch
- TrendView

* RT Professional 读写访问权限

在运行系统中您没有以下格式的访问权限：

- SymbolicIOField

Int32

可选项。用于指定以像素为单位的边框宽度的值或常量。值范围为 1 到 10。

参见

AlarmView (页 274)

RecipeView (页 458)

Slider (页 485)

StatusForce (页 493)

Switch (页 499)

TrendView (页 546)

Button (页 296)

GraphicIOField (页 371)

SymbolicIOField (页 504)

Font

说明

指定字体。

字体对象具有下列子属性

- Size（字体大小）
- Bold（是/否）
- Name（字体名称）
- Italic（是/否）
- Underline（是/否有下划线）
- StrikeThrough（是/否）

如果正好分配了两种字体属性，将仅采用默认属性"Name"。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**Font**.[**Size|Bold|Name|Italic|Underline|StrikeThrough**][=Value]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- Bar
- Clock
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- Slider
- TrendRulerControl
- UserArchiveControl

在运行系统中您没有以下格式的访问权限：

- Button
- CheckBox
- ComboBox
- DateTimeField
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- ProtectedAreaNameView
- RangeLabelView
- RecipeView
- RoundButton
- SmartClientView
- Switch
- SymbolicIOField
- TextField
- TrendView
- ZoneLabelView

Value

可选项。用于指定所选子属性的值或常量。

示例

```
'VBS74
Dim objControl1
Dim objControl2
Set objControl1 = ScreenItems("Control1")
Set objControl2 = ScreenItems("Control2")
objControl2.Font.Name = objControl1.Font.Name ' take over only the type of font
```

参见

AlarmControl (页 255)

Clock (页 323)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

Slider (页 485)

TrendRulerControl (页 532)

用户归档控件 (页 566)

Bar (页 285)

Button (页 296)

CheckBox (页 307)

ComboBox (页 326)

DateTimeField (页 334)

IOField (页 382)

Listbox (页 392)

MultiLineEdit (页 399)

OptionGroup (页 435)

ProtectedAreaNameView (页 453)

RangeLabelView (页 455)

RecipeView (页 458)

RoundButton (页 471)

SmartClientView (页 490)

Switch (页 499)

SymbolicIOField (页 504)

TextField (页 527)

TrendView (页 546)

ZoneLabelView (页 593)

FontBold

描述

指定文本是否以粗体显示。

运行系统中的访问权限：读和写

语法

Object.**FontBold**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox
- ComboBox
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- RoundButton
- SymbolicIOField*
- TextField

*：只读访问权限。

BOOLEAN

可选。如果文本以粗体进行显示，则为 TRUE。

FontItalic

描述

指定文本是否以斜体显示。

运行系统中的访问权限：读和写

语法

Object.**FontItalic**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Button
- CheckBox
- ComboBox
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- RoundButton
- SymbolicIOField*
- TextField

*：只读访问权限

BOOLEAN

可选。如果文本以斜体进行显示，则为 TRUE。

参见

TextField (页 527)

Button (页 296)

CheckBox (页 307)

ComboBox (页 326)

IOField (页 382)

Listbox (页 392)

MultiLineEdit (页 399)

OptionGroup (页 435)

RoundButton (页 471)

SymbolicIOField (页 504)

FontName

描述

指定字体。

运行系统中的访问权限：读和写

语法

Object.**FontName**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox
- ComboBox
- IOField
- ListBox
- OptionGroup
- RoundButton
- SymbolicIOField*
- TextField

*：只读访问权限

在运行系统中您没有以下格式的访问权限：

- MultiLineEdit

STRING

可选项。用于指定字体的值或常量。

参见

TextField (页 527)

Bar (页 285)

Button (页 296)

CheckBox (页 307)

ComboBox (页 326)

IOField (页 382)

Listbox (页 392)

MultiLineEdit (页 399)

OptionGroup (页 435)

RoundButton (页 471)

SymbolicIOField (页 504)

FontSize

描述

指定文本字体大小。

运行系统中的访问权限：读和写

语法

Object.**FontSize**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox
- ComboBox

- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- RoundButton
- SymbolicIOField*
- TextField

*：只读访问权限。

Int32

可选项。用于指定文本字体大小的值或常量。

FontUnderline

描述

指定文本是否应以下划线标出。

运行系统中的访问权限：读和写

语法

Object.**FontUnderline**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Button
- CheckBox
- ComboBox
- IOField
- ListBox
- MultiLineEdit
- OptionGroup
- RoundButton

- SymbolicIOField*
- TextField

*：只读访问权限

BOOLEAN

可选。如果文本以下划线进行显示，则为 TRUE。

ForeColor

描述

指定文本字体颜色。

运行系统中可进行读写访问

语法

Object.**ForeColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox **
- ComboBox **
- DateTimeField *
- IOField
- ListBox **
- MultiLineEdit **
- OptionGroup **
- RecipeView *
- RoundButton ***
- Slider
- Switch *

- SymbolLibrary
- SymbolicIOField ****
- TextField

* RT Professional 无访问权限

** RT Advanced 无访问权限

*** RT Advanced 无访问权限，RT Professional 只读访问权限

**** RT Professional 只读访问权限

在运行系统中您没有以下格式的访问权限：

- AlarmView

Color

可选项。用于指定文本字体颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Bar (页 285)

Button (页 296)

CheckBox (页 307)

ComboBox (页 326)

DateTimeField (页 334)

IOField (页 382)

Listbox (页 392)

MultiLineEdit (页 399)

OptionGroup (页 435)

RecipeView (页 458)

RoundButton (页 471)

Slider (页 485)

Switch (页 499)

SymbolicIOField (页 504)

SymbolLibrary (页 511)

TextField (页 527)

AlarmView (页 274)

ForeColorTransparency

描述

在运行系统中无访问权限。

Format

说明

在运行系统中无访问权限。

FormatPattern

描述

指定输出值的格式。

运行系统中的访问权限：读和写

语法

Object.**FormatPattern**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- IOField

STRING

可选项。用于指定输出值格式的值或常量。

参见

IOField (页 382)

FrameColor

描述

指定量表的背景色。

另外将“背景画面”(background graphic) 设为“无”(none)。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**FrameColor**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Gauge

Color

可选项。用于指定量表背景色的值或常量。

备注

可以使用“RGB”函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

Free

描述

在运行系统中无访问权限。

FreePercent

描述

以百分比形式返回可用磁盘空间的测量值。该值可以在运行系统中查询。该值无法预先定义。

运行系统中的访问权限：读取

语法

Object.**FreePercent**[=Int32]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- DiscSpaceView

Int32

可选项。用于返回百分比形式可用磁盘空间测量值的值或常量。

参见

DiskSpaceView (页 337)

1.5.5.6 属性 G-H

Gradation

描述

指定“Gauge”对象两个主要标记长度之间的差值。

运行系统中的访问权限：读和写

语法

Object.**Gradation**[=SINGLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Gauge

SINGLE

可选项。用于指定差值的值或常量。

参见

Gauge (页 366)

GraphDirection

描述

指定当前值在趋势视图的哪个边框上显示。

运行系统中可进行读写访问

语法

Object.**GraphDirection**[=GraphDirection]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- FunctionTrendControl
- OnlineTrendControl

GraphDirection

可选项。指定当前值在趋势视图的哪个边框上显示的值或常量。

值	说明
0	正值的图示方向为向右和向上。
-1	正值的图示方向为向左和向上。

值	说明
-2	正值的图示方向为向右和向上。
-3	正值的图示方向为向右和向下。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

GridlineAxis

说明

在运行系统中无访问权限。

GridLineColor

描述

指定网格线颜色。

运行系统中的访问权限

- RT Advanced：无访问权
- RT Professional：读和写

读和写

语法

Object.**GridLineColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- AlarmView*
- OnlineTableControl

- TrendRulerControl
- UserArchiveControl

* RT Advanced 读写访问权限，RT Professional 无访问权限

在运行系统中您没有以下格式的访问权限：

- TrendView

Color

可选项。用于指定网格线颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

AlarmView (页 274)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

TrendView (页 546)

GridlineEnabled

说明

在运行系统中无访问权限。

GridlineFillColor

描述

在运行系统中无访问权限。

GridlineStyle

描述

在运行系统中无访问权限。

GridLineWidth

描述

以像素为单位定义行/列分隔线的线条粗细。

运行系统中的访问权限：读和写

语法

Object.**GridLineWidth**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于指定网格线宽的值或常量。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

GSCRuntimeAllowed

说明

在运行系统中无访问权限。

HeaderFont

描述

在运行系统中无访问权限。

Height

描述

指定高度。

运行系统中的访问权限：读和写

在“Runtime Advanced”和“Panel Runtime”中，用户具备以下格式的只读权限：

- AlarmView
- Bar
- BatteryView
- Button
- CameraControl
- Clock
- DateTimeField
- FunctionTrendControl
- Gauge
- GraphicIOField
- GraphicView

- HTMLBrowser
- IOField
- Line
- MediaPlayer
- PDFView
- PLCCodeViewer
- Polygon
- Polyline
- ProDiagOverview
- ProtectedAreaNameView
- RangeLabelView
- RangeQualityView
- RecipeView
- S7GraphOverview
- Slider
- SmartClientView
- StatusForce
- Switch
- SymbolicIOField
- SymbolLibrary
- SysDiagControl
- TextField
- TrendView
- UserView
- WLanQualityView
- ZoneLabelView
- ZoneQualityView

语法

Object.**Height**[=Int32]

Object

必需项。"ScreenItem"类型的对象。该属性是 ScreenItem 对象的标准属性，因此适用于所有格式。

Int32

可选项。用于指定以像素为单位的高度的值或常量。

示例

以下示例将图像"NewPDL1"(名称以"Circle"开始）的所有对象的高度减半：

```
'VBS75
Dim objScreen
Dim objCircle
Dim lngIndex
Dim strName
lngIndex = 1
Set objScreen = HMIRuntime.Screens("NewPDL1")
For lngIndex = 1 To objScreen.ScreenItems.Count
'
'Searching all circles
strName = objScreen.ScreenItems.Item(lngIndex).ObjectName
If "Circle" = Left(strName, 6) Then
'
'to halve the height of the circles
Set objCircle = objScreen.ScreenItems(strName)
objCircle.Height = objCircle.Height / 2
End If
Next
```

参见

Line (页 388)

Polyline (页 448)

Ellipse (页 340)

Circle (页 312)

EllipseSegment (页 344)

CircleSegment (页 316)

EllipticalArc (页 348)

CircularArc (页 320)

Rectangle (页 467)

Polygon (页 444)

TextField (页 527)

IOField (页 382)

SymbolicIOField (页 504)

Button (页 296)

Switch (页 499)

GraphicView (页 375)

GraphicIOField (页 371)

Bar (页 285)

Clock (页 323)

Gauge (页 366)

Slider (页 485)

SymbolLibrary (页 511)

OnlineTrendControl (页 418)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

AlarmControl (页 255)

HTMLBrowser (页 379)

CheckBox (页 307)

OptionGroup (页 435)

WindowSlider (页 586)

Connector (页 330)

ScreenWindow (页 479)

DiskSpaceView (页 337)

ChannelDiagnose (页 305)

ScriptDiagnostics (页 482)

ProtectedAreaNameView (页 453)

UserView (页 581)

TubeTeeObject (页 563)

TubePolyline (页 560)

TubeDoubleTeeObject (页 557)

TubeArcObject (页 554)

MultiLineEdit (页 399)

MediaPlayer (页 396)

Listbox (页 392)

DateTimeField (页 334)

用户归档控件 (页 566)

TrendRulerControl (页 532)

AlarmView (页 274)

BatteryView (页 294)

RangeLabelView (页 455)

ZoneQualityView (页 595)

ZoneLabelView (页 593)

WLanQualityView (页 591)

TrendView (页 546)

SysDiagControl (页 514)

StatusForce (页 493)

SmartClientView (页 490)

RecipeView (页 458)

RangeQualityView (页 456)

ApplicationWindow (页 282)

ComboBox (页 326)

PLCCodeViewer (页 442)

RoundButton (页 471)

S7GraphOverview (页 476)

HelpText

描述

返回显示在运行系统中的工具提示，作为指定对象的操作帮助。

运行系统中可进行的访问：读和写

语法

Object.**HelpText**[=STRING]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- Button
- DateTimeField
- GraphicIOField
- IOField
- Switch
- SymbolicIOField
- TrendView

STRING

可选。用于指定工具提示内容的值或常量，工具提示显示在运行系统中作为指定对象的操作员帮助。

参见

Button (页 296)

DateTimeField (页 334)

GraphicIOField (页 371)

IOField (页 382)

Switch (页 499)

SymbolicIOField (页 504)

TrendView (页 546)

HiddenInput

描述

指定输入期间是否显示输入值或每个字符的 *。

运行系统中的访问权限：读和写

语法

Object.**HiddenInput**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- IOField

BOOLEAN

可选。若输入期间不显示输入值，则为 TRUE。 每个字符均显示 *。

参见

IOField (页 382)

HighlightColor

说明

指定图形概览中的高亮颜色。

运行系统中可进行读写访问

语法

Object.**HighlightColor**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- S7GraphOverview

Color

可选项。用于指定突出显示颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 vbRed 和 vbGreen。

参见

S7GraphOverview (页 476)

HighLimitColor

描述

指定滚动条中顶部或右侧滚动按钮的颜色。

运行系统中可进行读写访问

语法

Object.**HighLimitColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- WindowSlider

Color

可选项。用于指定滚动条中顶部或右侧滚动按钮颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 vbRed 和 vbGreen。

参见

WindowSlider (页 586)

Hitlist

描述

在运行系统中无访问权限。

HitlistColumnAdd

描述

在报警统计中创建新的报警文本块。新创建的报警文本块会自动使用"HitlistColumnIndex"而引用。

运行系统中可进行读写访问

语法

Object.**HitlistColumnAdd**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

STRING

必需项。用于使用"HitlistColumnName"指定报警统计中新报警文本块名称的值或常数。

参见

HitlistColumnIndex (页 872)

AlarmControl (页 255)

HitlistColumnCount

描述

指定报警统计中已组态的报警文本块的数目。

运行系统中可进行读写访问

语法

Object.**HitlistColumnCount**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

Int32

可选项。用于指定报警统计中组态的报警文本块数量的值或常量。

参见

AlarmControl (页 255)

HitlistColumnIndex

描述

引用为报警统计选择的报警文本块。要访问报警文本块的属性，需要设置 ""HitlistColumnIndex""。

介于 0 至 (HitlistColumnIndex - 1) 之间的值为"HitlistColumnCount"的有效值。"HitlistColumnCount"属性可指定组态的报警文本块数量。

运行系统中可进行读写访问

语法

Object.**HitlistColumnIndex**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

Int32

可选项。用于通过索引指定要编辑的报警文本块的值或常量。

参见

HitlistColumnCount (页 872)

AlarmControl (页 255)

HitlistColumnName

说明

指定使用"HitlistColumnIndex"引用的报警统计报警文本块的名称。不能编辑此名称。

运行系统中可进行读写访问

语法

Object.**HitlistColumnName**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

STRING

可选项。用于指定报警统计中引用的报警文本块名称的值或常数。

参见

HitlistColumnIndex (页 872)

AlarmControl (页 255)

HitlistColumnRemove

描述

使用引用的报警统计报警文本块的名称可移除该报警统计报警文本块。

运行系统中可进行读写访问

语法

Object.**HitlistColumnRemove**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

STRING

可选项。用于指定待移除的已引用的报警统计报警文本块名称的值或常量。

参见

AlarmControl (页 255)

HitlistColumnRepos

描述

指定在报警统计中使用"HitlistColumnIndex"引用的报警文本块位置。

如果已使用"HitlistColumnRepos"更改了报警文本块位置，"HitlistColumnRepos"的值则会分配给"HitlistColumnIndex"。

运行系统中可进行读写访问

语法

Object.**HitlistColumnRepos**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

Int32

可选项。用于指定报警统计中引用的报警文本块位置的值或常量。值范围为 0 到 (HitlistColumnCount - 1)。超出该范围的值无效。

0：引用的报警文本块放置在左侧。

参见

HitlistColumnIndex (页 872)

HitlistColumnCount (页 872)

AlarmControl (页 255)

HitlistColumnSort

说明

指定统计列表中"HitlistColumnIndex"引用的报警文本块的排序顺序。

运行系统中可进行读写访问

语法

Object.**HitlistColumnSort**[=SortMode]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

SortMode

可选项。用于指定可导出控件的运行系统数据的值或常量。

值	说明
0	无排序
1	从最小值到最大值升序排列
2	从最大值到最小值降序排列

参见

AlarmControl (页 255)

HitlistColumnSortIndex

描述

指定使用"HitlistColumnIndex"引用的报警统计报警文本块的排序顺序。

如果将该值设置为"0"，将从"HitlistColumnSort"中移除排序标准。

运行系统中可进行读写访问

语法

Object.**HitlistColumnSortIndex**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

Int32

可选项。指定引用的报警统计报警文本块的排序顺序。

参见

AlarmControl (页 255)

HitlistColumnVisible

描述

指定是否在报警统计中显示使用"HitlistColumnIndex"引用的报警文本块。

运行系统中可进行读写访问

语法

Object.**HitlistColumnVisible**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

BOOLEAN

可选项。

TRUE，显示引用的报警统计报警文本块。

FALSE，不显示引用的报警统计报警文本块。

参见

HitlistColumnIndex (页 872)

AlarmControl (页 255)

HitlistDefaultSort

描述

指定报警统计表格列的默认排序顺序。

运行系统中可进行读写访问

语法

Object.**HitlistDefaultSort**[=SortMode]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

SortMode

可选项。用于指定报警统计表格列的默认排序顺序的值或常量。

值	说明
0	该列表按频率升序排列。
1	该列表按频率降序排列。

参见

AlarmControl (页 255)

HitlistFilter

说明

在运行系统中无访问权限。

HitlistMaxSourceItems

描述

指定可通过报警日志用于创建报警统计的最大数据记录数。该值可为介于“0 - 10000”的任意值。

运行系统中可进行读写访问

语法

Object.**HitlistMaxSourceItems**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl

Int32

可选项。用于指定可通过报警日志来创建报警统计的最大数据记录数的值或常量。

参见

AlarmControl (页 255)

HitlistMaxSourceItemsWarn

描述

指定是否在报警视图中数据记录数达到最大值时发出警告。

运行系统中可进行读写访问

语法

Object.**HitlistMaxSourceItemsWarn**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

BOOLEAN

"TRUE"，报警视图中数据记录数达到最大值时发出警告。

"FALSE"，报警视图中数据记录数达到最大值时不发出警告。

参见

AlarmControl (页 255)

HitListRelTime

描述

指定是否限制报警统计的计算时间段。

运行系统中可进行读写访问

语法

Object.**HitListRelTime** [=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

BOOLEAN

TRUE，对统计使用指定的时间范围。

FALSE，不使用指定时间范围。

参见

AlarmControl (页 255)

HitListRelTimeFactor

说明

指定时间类型“HitlistRelTimeFactorType (页 880)”旁边的时间因数，以确定创建报警统计的时间段。

运行系统中的访问权限：读和写

语法

Object.**HitListRelTimeFactor**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl

Int32

可选项。用于指定时间因数的值或常量。

参见

AlarmControl (页 255)

HitlistRelTimeFactorType

描述

指定用于定义为报警统计显示的时间范围和时间因素“HitlistRelTimeFactor”的时间单位。

运行系统中可进行读写访问

语法

Object.**HitListRelTimeFactorType**[=AlarmControlTimeUnit)

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

AlarmControlTimeUnit

可选项。用于指定时间单元的值或常量。

值	名称
0	分钟
1	小时
2	日
3	周
4	月

参见

AlarmControl (页 255)

HomeButtonVisible

说明

在运行系统中无访问权限。

HorizontalAlignment

描述

指定文本水平对齐。

运行系统中可进行读写访问

语法

Object.**HorizontalAlignment**[=HorizontalAlignment]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Button
- CheckBox *
- ComboBox *
- DateTimeField **
- IOField
- ListBox *
- MultiLineEdit *
- OptionGroup *
- RoundButton ***
- Switch **
- SymbolicIOField ****
- TextField

* RT Advanced 无访问权限

** RT Professional 无访问权限

*** RT Advanced 无访问权限，RT Professional 只读访问权限

**** RT Professional 只读访问权限

HorizontalAlignment

可选项。用于指定文本水平对齐的值或常量。

值	VB 常量	说明
0	hmiAlignmentLeft	文本左对齐。
1	hmiAlignmentCentered	文本在对象中将水平居中对齐。
2	hmiAlignmentRight	文本在对象内为右对齐。

参见

Button (页 296)

CheckBox (页 307)

ComboBox (页 326)

DateTimeField (页 334)

IOField (页 382)

Listbox (页 392)

MultiLineEdit (页 399)

OptionGroup (页 435)

RoundButton (页 471)

Switch (页 499)

SymbolicIOField (页 504)

TextField (页 527)

HorizontalGridLines

描述

指定是否显示水平分隔线。

运行系统中可进行读写访问

语法

Object.**HorizontalGridLines**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，显示水平分隔线。

FALSE，不显示水平分隔线。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

HorizontalPictureAlignment

说明

在运行系统中无访问权限。

HorizontalScrollBarPosition

说明

指定水平滚动条中的滑块位置。通过水平移动滚动条滑块可显示画面。

如果要将画面显示为剪切状，其中滚动条位于画面的左上边缘，请使用“LeftOffset (页 926)”和“TopOffset (页 1282)”属性来指定该剪切状画面。

运行系统中可进行读写访问

语法

Object.**HorizontalScrollBarPosition**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- ScreenWindow

Int32

可选项。用于指定滚动条中滑块的位置的值或常量。

参见

ScreenWindow (页 479)

HorizontalScrollingEnabled

说明

在运行系统中无访问权限。

Hotkey

说明

在运行系统中无访问权限。

HourNeedleHeight

描述

指定时针的长度。

运行系统中可进行读写访问

语法

Object.**HourNeedleHeight**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Clock

Int32

可选项。用于指定时针长度的值或常量。

指定时针的长度，作为与钟盘半径相关的百分值。

参见

Clock (页 323)

HourNeedleWidth

描述

指定时针的宽度。

运行系统中可进行读写访问

语法

Object.**HourNeedleWidth**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Clock

Int32

可选项。用于指定时针宽度的值或常量。

指定时针的宽度，作为与两倍钟盘长度相关的百分值。

参见

Clock (页 323)

1.5.5.7 属性 I-J

IconSpace

描述

定义表格单元格中图标和文本的间距。 显示图标和文本时，该值将激活。

运行系统中的访问权限：读和写

语法

Object.**IconSpace**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于指定间距的值。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

Index

描述

指定所选文本域的索引。

运行系统中的访问权限：读和写

语法

Object.**Index**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- CheckBox
- ComboBox
- ListBox
- OptionGroup

Int32

可选项。用于指定所选文本域索引的值或常量。

参见

CheckBox (页 307)

ComboBox (页 326)

Listbox (页 392)

OptionGroup (页 435)

IndipendentWindow

说明

在运行系统中无访问权限。

InfoArea_BackgroundColor

说明

在运行系统中无访问权限。

InfoArea_ColumnsMovable

说明

在运行系统中无访问权限。

InfoArea_DefaultTextColor

说明

在运行系统中无访问权限。

InfoArea_ErrorTextBackgroundColor

说明

在运行系统中无访问权限。

InfoArea_ErrorTextColor

说明

在运行系统中无访问权限。

InfoArea_FocusFrameColor

说明

在运行系统中无访问权限。

InfoArea_FocusFrameWidth

说明

在运行系统中无访问权限。

InfoArea_Font

说明

在运行系统中无访问权限。

InfoArea_RootNodeText

说明

在运行系统中无访问权限。

InfoArea_SelectionBackgroundColor

说明

在运行系统中无访问权限。

InfoArea_SelectionForegroundColor

说明

在运行系统中无访问权限。

InfoArea_ShowGridLines

说明

在运行系统中无访问权限。

InfoArea_TableHeaderBackgroundColor

说明

在运行系统中无访问权限。

InfoArea_TableHeaderTextColor

说明

在运行系统中无访问权限。

InnerBackColorOff

描述

若对象状态为 OFF，则指定“Switch”对象滑块下方的颜色。

运行系统中可进行读写访问

语法

Object.**InnerBackColorOff**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Switch

Color

可选项。用于指定 OFF 状态颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Switch (页 499)

InnerBackColorOn

描述

若对象状态为 ON，则指定"Switch"对象滑块下方的颜色。

运行系统中可进行读写访问

语法

Object.**InnerBackColorOn**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Switch

Color

可选项。用于指定 ON 状态颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Switch (页 499)

InnerDialColor

说明

在运行系统中无访问权限。

InnerDialInnerDistance

说明

在运行系统中无访问权限。

InnerDialOuterDistance

说明

在运行系统中无访问权限。

InnerHeight

描述

指定 ###

运行系统中的访问权限

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**InnerHeight**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Button
- IOField
- RoundButton*
- SymbolicIOField*

*：只读访问权限

Int32

可选项。值或常量，其中 ###

InnerWidth

说明

指定 ###

运行系统中的访问权限：读和写

语法

Object.**InnerWidth**[=Int32]

Object

必选项。具有以下格式的"ScreenItem"类型的对象：

- Button
- RoundButton*

*：只读访问权限

Int32

可选项。值或常量，其中 ###

InputAddressText

说明

在运行系统中无访问权限。

InputValue

说明

指定输入值。设置属性时对象中不显示该值。

要在对象中显示该输入的值，请将系统函数"SetPropertyByProperty"组态至事件"对象已更改"：

相关参数	值
属性名称	过程值
属性的名称	输入值

此系统功能只有在输出值未组态变量连接，但用户仍想通过某种方式（例如通过脚本）查询指定值时才能发挥作用。

运行系统中可进行读写访问

语法

Object.**InputValue**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- IOField
- SymbolicIOField*

*：只读访问权限

Int32

可选项。用于指定输入值的值或常量。

InspectorViewInternalColumnOrder

说明

在运行系统中无访问权限。

InspectorViewRowOrder

说明

在运行系统中无访问权限。

实例

说明

返回报警对象的实例。

IntegerDigits

描述

确定整数位个数（0 ～ 20）。

运行系统中的访问权限：读和写

语法

Object.**IntegerDigits**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

Int32

可选项。用于指定整数位个数（0 至 20）的值或常量。

参见

Bar (页 285)

Interval

描述

指定更新显示的测量值的时间间隔。输入数值，以分钟为单位。

运行系统中的访问权限：读取

语法

Object.**Interval**[=Int32]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- DiscSpaceView

Int32

可选项。用于指定更新测量值的时间间隔的值或常量。

参见

DiskSpaceView (页 337)

IsActive

说明

在运行系统中无访问权限。

IsImageMiddleAligned

描述

在运行系统中无访问权限。

IsMinPasswordValueSet

说明

在运行系统中无访问权限。

IsRunningUnderCE

说明

在运行系统中无访问权限。

IsVerticalScrollBarEnabled

说明

在运行系统中无访问权限。

ItemBorderStyle

描述

指定选择列表中分隔线的线类型。

运行系统中可进行读访问

语法

Object.**ItemBorderStyle**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- SymbolicIOField

Int32

可选项。用于指定选择列表中分隔线类型的值或常量。

值	VB 常量	说明
-1	hmiLineStyleNone	选择列表无分隔线。
0	hmiLineStyleSolid	选择列表的分隔线为实心样式。
1	hmiLineStyleDash	选择列表的分隔线为虚线。
2	hmiLineStyleDot	选择列表的分隔线为点线。
3	hmiLineStyleDashDot	选择列表分隔线为点划线。
4	hmiLineStyleDashDotDot	选择列表分隔线为双点划线。

参见

SymbolicIOField (页 504)

ItemText_AKZ

说明

在运行系统中无访问权限。

ItemText_Descriptor

说明

在运行系统中无访问权限。

ItemText_ErrorText

说明

在运行系统中无访问权限。

ItemText_HardwareRevision

说明

在运行系统中无访问权限。

ItemText_IMDataVersion

说明

在运行系统中无访问权限。

ItemText_InstallationDate

说明

在运行系统中无访问权限。

ItemText_LADDR

说明

在运行系统中无访问权限。

ItemText_ManufacturerID

说明

在运行系统中无访问权限。

ItemText_Name

说明

在运行系统中无访问权限。

ItemText_OKZ

说明

在运行系统中无访问权限。

ItemText_OperationState

说明

在运行系统中无访问权限。

ItemText_OrderID

说明

在运行系统中无访问权限。

ItemText_ProfileID

说明

在运行系统中无访问权限。

ItemText_Rack

说明

在运行系统中无访问权限。

ItemText_RevisionCounter

说明

在运行系统中无访问权限。

ItemText_SerialNumber

说明

在运行系统中无访问权限。

ItemText_Slot

说明

在运行系统中无访问权限。

ItemText_SoftwareRevision

说明

在运行系统中无访问权限。

ItemText_SpecificProfileData

说明

在运行系统中无访问权限。

ItemText_State

说明

在运行系统中无访问权限。

ItemText_Station

说明

在运行系统中无访问权限。

ItemText_SubAddress

说明

在运行系统中无访问权限。

ItemText_SubSlot

说明

在运行系统中无访问权限。

ItemText_SubSystem

说明

在运行系统中无访问权限。

ItemText_Type

说明

在运行系统中无访问权限。

IV_ShowItem_AKZ

说明

在运行系统中无访问权限。

IV_ShowItem_Descriptor

说明

在运行系统中无访问权限。

IV_ShowItem_ErrorText

说明

在运行系统中无访问权限。

IV_ShowItem_HardwareRevision

说明

在运行系统中无访问权限。

IV_ShowItem_IMDataVersion

说明

在运行系统中无访问权限。

IV_ShowItem_InstallationDate

说明

在运行系统中无访问权限。

IV_ShowItem_LADDR

说明

在运行系统中无访问权限。

IV_ShowItem_ManufacturerID

说明

在运行系统中无访问权限。

IV_ShowItem_Name

说明

在运行系统中无访问权限。

IV_ShowItem_OKZ

说明

在运行系统中无访问权限。

IV_ShowItem_OperationState

说明

在运行系统中无访问权限。

IV_ShowItem_OrderID

说明

在运行系统中无访问权限。

IV_ShowItem_ProfileID

说明

在运行系统中无访问权限。

IV_ShowItem_Rack

说明

在运行系统中无访问权限。

IV_ShowItem_RevisionCounter

说明

在运行系统中无访问权限。

IV_ShowItem_SerialNumber

说明

在运行系统中无访问权限。

IV_ShowItem_Slot

说明

在运行系统中无访问权限。

IV_ShowItem_SoftwareRevision

说明

在运行系统中无访问权限。

IV_ShowItem_SpecificProfileData

说明

在运行系统中无访问权限。

IV_ShowItem_State

说明

在运行系统中无访问权限。

IV_ShowItem_Station

说明

在运行系统中无访问权限。

IV_ShowItem_SubAddress

说明

在运行系统中无访问权限。

IV_ShowItem_SubSlot

说明

在运行系统中无访问权限。

IV_ShowItem_SubSystem

说明

在运行系统中无访问权限。

IV_ShowItem_Type

说明

在运行系统中无访问权限。

JumpToLimitsAfterMouseClick

描述

指定是否对相关终值设置滑块。终值为最小或最大值。要为终值设置滑块，单击当前滑块设置范围以外的区域。

运行系统中的访问权限：读和写

语法

Object.**JumpToLimitsAfterMouseClick**[=BOOLEAN]

Object

必选项。具有以下格式的"ScreenItem"类型的对象：

- WindowSlider

BOOLEAN

可选项。如果将滑块设为相关终值，则为 TRUE。

参见

WindowSlider (页 586)

1.5.5.8 属性 K-L

KeyboardOnline

说明

在运行系统中无访问权限。

LabelColor

描述

指定刻度标签的颜色。

运行系统中可进行读写访问

语法

Object.**LabelColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Slider

Color

可选项。指定刻度标签颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Slider (页 485)

Language

说明

确定当前运行系统的语言。

运行系统中可进行的访问： 读和写

语法

Object.**Language**[= LONG]

Object

要求"HMIRuntime"对象。

LONG

指定国标代码的选项 A 值或常量。

注释

指定 VBS 运行系统语言的国标代码，例如：德语代码 1031，英语 2057 等 请参考"当前局部 ID (LCID) 图"下方的 VBScript 基本内容，了解各种语言的国标代码。

参见

HMIRuntime（页 224）

LargeTickLabelingStep

说明

返回对刻度部分加标签所需的间隔。

运行系统中的访问权限：读和写

语法

Object.**LargeTickLabelingStep**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar

Int32

可选项。用于返回对刻度部分加标签所需的间隔的值或常量。

参见

Bar (页 285)

LargeTicksBold

描述

指定主要刻度标记长度是否以粗体显示。

运行系统中的访问权限：读和写

语法

Object.**LargeTicksBold**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

BOOLEAN

可选。若以粗体显示主要刻度标记长度，则为 TRUE,。

参见

Bar (页 285)

LargeTicksSize

描述

指定主要刻度标记长度的长度。

运行系统中的访问权限：读和写

语法

Object.**LargeTicksSize**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

Int32

可选项。用于指定主要刻度标记长度之长度的值或常量。

参见

Bar (页 285)

LastConnectedObject

描述

在运行系统中无访问权限。

LastConnectedObjectIndex

说明

指定最后一个连接对象的连接点的索引值。

运行系统中的访问权限：读和写

语法

Object.**LastConnectedObjectIndex**[=Int]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Connector

Int

可选项。用于指定最后一个连接对象的连接点的索引号的值或常量。

参见

Connector (页 330)

LastConnectedObjectName

说明

指定连接于连接器下部点的对象的名称。

运行系统中的访问权限：读和写

语法

Object.**LastConnectedObjectName**[=String]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Connector

String

可选项。用于指定连接于连接器下部点的对象名称的值或常量。

参见

Connector (页 330)

LastError

说明

返回有关上一次操作结果的错误代码，例如有关变量读写流程的信息：

十六进制符号代码	说明
0x00000000	OK
0x80040001	执行错误
0x80040002	变量错误
0x80040003	服务器不可用。
0x80040004	多个变量错误：一个或多个变量错误

为了获得错误描述，首先执行 Read 方法。

说明

若通过 TagSet 对象访问时发生错误，则判断 TagSet 对象各个变量的 LastError 属性。

要获取所提供的值的特性描述，使用“QualityCode”属性。 要获取错误描述，请使用“ErrorDescription”属性。

运行中可进行的访问：读取

语法

Object.**LastError**

Object

必需。 “Tag”对象。

示例

下列示例显示的是“Tag1”变量的错误代码：

```
'VBS77
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
MsgBox objTag.LastError
```

下列示例显示的是向 TagSet 列表添加两个变量，将 LastError 属性作为 Trace 输出：

```
'VBS178
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
HMIRuntime.Trace "LastError: " & group.LastError & vbNewLine
```

按如下步骤可以访问列表中变量的 LastError 属性：

```
HMIRuntime.Trace "LastError: " & group("Motor1").LastError & vbNewLine
```

参见

变量 (页 244)

TagSet（列表） (页 248)

Layer

说明

返回画面中对象所在的层。总共有 32 层，其中层“0”为底层，层“31”为顶层。

组态对象最初位于层背景中。

运行系统中的访问权限：读和写

语法

Object.**Layer**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl *
- AlarmView **
- ApplicationWindow *
- Bar

- BatteryView ***
- Button
- CameraControl ***
- ChannelDiagnose *
- CheckBox *
- Circle
- CircleSegment *
- CircularArc *
- Clock
- ComboBox *
- Connector *
- DateTimeField **
- DiscSpaceView *
- Ellipse
- EllipseSegment *
- EllipticalArc *
- FunctionTrendControl **
- Gauge
- GraphicIOField
- GraphicView
- HTMLBrowser
- IOField
- Line
- ListBox *
- MediaPlayer
- MultiLineEdit *
- OptionGroup *
- PLCCodeViewer **
- Polygon

- Polyline
- ProDiagOverview ****
- ProtectedAreaNameView ***
- RangeLabelView ***
- RangeQualityView ***
- RecipeView **
- Rectangle
- RoundButton *****
- S7GraphOverview
- ScreenWindow *
- Slider
- SmartClientView **
- StatusForce **
- Switch **
- SymbolLibrary
- SymbolicIOField ******
- SysDiagControl *******
- TextField
- TrendRulerControl *
- TrendView **
- TubeArcObject *
- TubeDoubleTeeObject *
- TubeTeeObject *
- Tubepolyline *
- UserView
- WLanQualityView ***
- WindowSlider *
- ZoneLabelView ***
- ZoneQualityView ***

* RT Advanced 无访问权限

** RT Professional 无访问权限

*** 仅限 Panel RT 读写访问权限

**** RT Advanced 读访问权限，RT Professional 读写访问权限

***** RT Advanced 无访问权限，RT Professional 读访问权限

****** RT Advanced 读写访问权限，RT Professional 读访问权限

******* 只读访问权限

在运行系统中您没有以下格式的访问权限：

- OnlineTableControl
- OnlineTrendControl
- UserArchiveControl

Int32

可选项。用于返回画面中对象所在的层的值或常量。

注释

说明

"Layer"属性指定对象所在的层。图层"0"输出为层"0"。
访问时，VBS 中的层从 1 开始计数。因此，根据"layers(2)"对层 1 进行寻址。

示例

以下示例显示的是画面"NewPDL1"中所有对象的名称和层：

```
'VBS78
Dim objScreen
Dim objScrItem
Dim lngAnswer
Dim lngIndex
Dim strName
lngIndex = 1
Set objScreen = HMIRuntime.Screens("NewPDL1")
For lngIndex = 1 To objScreen.ScreenItems.Count
strName = objScreen.ScreenItems.Item(lngIndex).ObjectName
Set objScrItem = objScreen.ScreenItems(strName)
lngAnswer = MsgBox(strName & " is in layer  " & objScrItem.Layer,vbOKCancel)
If vbCancel = lngAnswer Then Exit For
Next
```

参见

AlarmControl (页 255)

AlarmView (页 274)

ApplicationWindow (页 282)

Bar (页 285)

BatteryView (页 294)

Button (页 296)

CameraControl (页 303)

ChannelDiagnose (页 305)

CheckBox (页 307)

Circle (页 312)

CircleSegment (页 316)

CircularArc (页 320)

Clock (页 323)

ComboBox (页 326)

Connector (页 330)

DateTimeField (页 334)

SmartClientView (页 490)

StatusForce (页 493)

Switch (页 499)

SymbolicIOField (页 504)

SymbolLibrary (页 511)

SysDiagControl (页 514)

TextField (页 527)

TrendRulerControl (页 532)

TrendView (页 546)

TubeArcObject (页 554)

TubeDoubleTeeObject (页 557)

TubePolyline (页 560)

TubeTeeObject (页 563)

用户归档控件 (页 566)

UserView (页 581)

WindowSlider (页 586)

WLanQualityView (页 591)

ZoneLabelView (页 593)

ZoneQualityView (页 595)

LayerDeclutteringEnable

说明

指出是否根据设置的最小与最大缩放值显示或隐藏画面的层。

运行中可进行的访问：读取

语法

Object.**LayerDeclutteringEnable**

Object

必需。 “Screen”对象。

示例：

该示例显示的是将画面“NewPDL1”的“LayerDecluttering”属性作为 Trace 输出。

```
'VBS156
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
HMIRuntime.Trace "Enable: " & objScreen.LayerDeclutteringEnable & vbNewLine
```

参见

Screen (页 231)

Layers

说明

返回“Layers”类型对象。

运行系统中可进行的访问：Read

语法

Object.**Layers**

Object

要求“Screen”对象。

参见

Screen (页 231)

Left

描述

指定 X 坐标的值。

运行系统中的访问权限：读和写

语法

Object.**Left**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl *
- AlarmView **
- ApplicationWindow *
- Bar
- BatteryView ***
- Button
- ChannelDiagnose *
- CheckBox *
- Circle
- CircleSegment *
- CircularArc *
- Clock
- ComboBox *
- Connector *
- DateTimeField **
- DiscSpaceView *
- Ellipse
- EllipseSegment *
- EllipticalArc *
- FunctionTrendControl
- Gauge
- GraphicIOField
- GraphicView

- HTMLBrowser
- IOField
- Line
- ListBox *
- MediaPlayer
- MultiLineEdit *
- OnlineTableControl *
- OnlineTrendControl *
- OptionGroup *
- PLCCodeViewer **
- Polygon
- Polyline
- ProDiagOverview ***
- ProtectedAreaNameView ***
- RangeLabelView ***
- RangeQualityView ***
- RecipeView**
- Rectangle
- RoundButton *****
- S7GraphOverview **
- ScreenWindow *
- Slider
- SmartClientView **
- StatusForce **
- Switch **
- SymbolLibrary
- SymbolicIOField ******
- SysDiagControl *******
- TextField

- TrendRulerControl *
- TrendView **
- TubeArcObject *
- TubeDoubleTeeObject *
- TubeTeeObject *
- Tubepolyline *
- UserArchiveControl *
- UserView
- WLanQualityView ***
- WindowSlider *
- ZoneLabelView ***
- ZoneQualityView ***

* RT Advanced 无访问权限

** RT Professional 无访问权限

*** 仅限 Panel RT 读写访问权限

**** RT Advanced 读访问权限，RT Professional 读写访问权限

***** RT Advanced 无访问权限，RT Professional 读访问权限

****** RT Advanced 读写访问权限，RT Professional 读访问权限

Int32

可选项。其中包含以像素为单位的 X 坐标值的值或常量（从画面左上方测量）。

注释

X 坐标指包围对象的矩形左上方的值。在运行系统中还监视画面的极值。如果指定的坐标值超出显示大小，则用户自定义函数会中断，并显示一条错误消息。

参见

Bar (页 285)

BorderWidth (页 697)

CameraControl (页 303)

WindowSlider (页 586)

WLanQualityView (页 591)

ZoneLabelView (页 593)

ZoneQualityView (页 595)

LeftMargin

说明

在运行系统中无访问权限。

LeftOffset

说明

指定大于画面窗口的画面显示的零点水平位移。 此位移参照画面窗口的左侧边缘。

显示的画面为剪切画面。 画面滚动条位于画面左边缘和上边缘。

如果希望使用画面滚动条的水平和垂直偏移在画面窗口中显示画面，应对偏移使用“HorizontalScrollBarPosition (页 884)”和 “VerticalScrollbarPosition (页 1415)”属性。

运行系统中的访问权限： 读和写

语法

Object.**LeftOffset**[=Int32]

Object

必需项。 具有以下格式的“ScreenItem”类型的对象：

- Screenwindow

Int32

可选项。 值或常量，用于指定画面显示零点自画面窗口左侧边缘的水平位移。

参见

ScreenWindow (页 479)

Limit4LowerLimit

描述

指定 "Reserve4" 的下限。

“Limit4LowerLimitEnabled”属性必须设为 TRUE 以监视“Reserve4””限值。

运行系统中的访问权限：读和写

语法

Object.**Limit4LowerLimit**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

DOUBLE

可选项。用于指定“Reserve4”下限的值或常量。

注释

“Limit4LowerLimitRelative”属性指定对象是以百分比还是绝对值形式进行评估。

参见

Bar (页 285)

Limit4LowerLimitColor

描述

指定“Reserve4”下限的颜色。

若滚动条颜色在达到限值后立即更改，则“Limit4LowerLimitEnabled”属性值必须为 TRUE。

运行系统中可进行读写访问

语法

Object.**Limit4LowerLimitColor**[=Color]

Object

必需项。具有以下格式的""ScreenItem"类型的对象：

- Bar

Color

可选项。用于指定 "Reserve4" 下限的颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Bar (页 285)

Limit4LowerLimitEnabled

描述

指定是否要监视下限 "Reserve4"。

运行系统中的访问权限：读和写

语法

Object.**Limit4LowerLimitEnabled**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

BOOLEAN

可选。若要监视下限 "Reserve4"，则为 TRUE。

注释

通过属性""Limit4LowerLimit"、"Limit4LowerLimitColor"和"Limit4LowerLimitRelative""指定以下值：

- 限值
- 达到限值后的显示
- 估算的类型

参见

Bar (页 285)

Limit4LowerLimitRelative

描述

指定"Reserve4"下限是以百分比还是绝对值进行估算。

运行系统中可进行读写访问

语法

Object.**Limit4LowerLimitRelative**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar

BOOLEAN

可选项。

TRUE，将"Reserve4"下限以百分比形式估算。

FALSE，将"Reserve4"下限以绝对值形式估算。

参见

Bar (页 285)

Limit4UpperLimit

描述

指定“Reserve4”的下限。

“Limit4UpperLimitEnabled””属性必须设为 TRUE 以监视“"Reserve4”限值。

运行系统中的访问权限：读和写

语法

Object.**Limit4UpperLimit**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

DOUBLE

可选项。用于指定“Reserve4”上限的值或常量。

注释

“"Limit4UpperLimitRelative”属性指定对象是以百分比还是绝对值形式进行评估。

参见

Bar (页 285)

Limit4UpperLimitColor

描述

指定 "Reserve4" 上限的颜色。

若滚动条颜色在达到限值后立即更改，则“Limit4UpperLimitEnabled””属性值必须为 TRUE。

运行系统中可进行读写访问

语法

Object.**Limit4UpperLimitColor**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Bar

Color

可选项。用于指定“Reserve4”上限颜色的值或常量。

注释

可以使用“RGB”函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Bar (页 285)

Limit4UpperLimitEnabled

描述

指定是否要监视“Reserve4”上限。

运行系统中的访问权限：读和写

语法

Object.**Limit4UpperLimitEnabled**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

BOOLEAN

可选。若要监视 "Reserve4" 上限，则为 TRUE。

注释

通过属性""Limit4UpperLimit"、"Limit4UpperLimitColor"和"Limit4UpperLimitRelative"定义以下值：

- 限值
- 达到限值后的显示
- 估算的类型

参见

Bar (页 285)

Limit4UpperLimitRelative

描述

指定"Reserve4"上限是以百分比还是绝对值进行估算。

运行系统中可进行读写访问

语法

Object.**Limit4UpperLimitRelative**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar

BOOLEAN

可选项。

TRUE，将"Reserve4"下限以百分比形式估算。

FALSE，将"Reserve4"下限以绝对值形式估算。

参见

Bar (页 285)

Limit5LowerLimit

描述

指定 "Reserve5" 的下限。

“Limit5LowerLimitEnabled”属性必须设为 TRUE 以监视“"Reserve5”限值。

运行系统中的访问权限： 读和写

语法

Object.**Limit5LowerLimit**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

DOUBLE

可选项。用于指定 "Reserve5" 下限的值或常量。

注释

“Limit5LowerLimitRelative”属性指定对象是以百分比还是绝对值形式进行评估。

参见

Bar (页 285)

Limit5LowerLimitColor

描述

指定 "Reserve5" 下限的颜色。

若滚动条颜色在达到限值后立即更改，则“Limit5LowerLimitEnabled”属性值必须为 TRUE。

运行系统中可进行读写访问

语法

Object.**Limit5LowerLimitColor**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Bar

Color

可选项。用于指定“"Reserve5”下限的颜色的值或常量。

注释

可以使用“RGB”函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Bar (页 285)

Limit5LowerLimitEnabled

描述

指定是否要监视下限“Reserve5”。

运行系统中的访问权限：读和写

语法

Object.**Limit5LowerLimitEnabled**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

BOOLEAN

可选。若要监视 "Reserve5" 下限，则为 TRUE。

注释

通过属性“Limit5LowerLimit”、“Limit5LowerLimitColor”和“Limit5LowerLimitRelative”定义以下值：

- 限值
- 达到限值后的显示
- 估算的类型

参见

Bar (页 285)

Limit5LowerLimitRelative

描述

指定 "Reserve5" 下限是以百分比还是绝对值进行估算。

运行系统中可进行读写访问

语法

Object.**Limit5LowerLimitRelative**[=BOOLEAN]

Object

必需项。具有以下格式的 "ScreenItem" 类型的对象：

- Bar

BOOLEAN

可选项。

TRUE，将 "Reserve5" 下限以百分比形式估算。

FALSE，将 "Reserve5" 下限以绝对值形式估算。

参见

Bar (页 285)

Limit5UpperLimit

描述

指定“Reserve5”的下限。

“"Limit5UpperLimitEnabled”属性必须设为 TRUE 以监视“"Reserve5”限值。

运行系统中的访问权限：读和写

语法

Object.**Limit5UpperLimit**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

DOUBLE

可选项。用于指定 "Reserve5" 上限的值或常量。

注释

“TypeLimitHigh5”属性指定对象是以百分比还是绝对值形式进行评估。

参见

Bar (页 285)

Limit5UpperLimitColor

描述

指定 "Reserve5" 上限的颜色。

若滚动条颜色在达到限值后立即更改，则“Limit5UpperLimitEnabled”属性值必须为 TRUE。

运行系统中可进行读写访问

语法

Object.**Limit5UpperLimitColor**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Bar

Color

可选项。用于指定“"Reserve5”上限颜色的值或常量。

注释

可以使用“RGB”函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Bar (页 285)

Limit5UpperLimitEnabled

描述

指定是否要监视“Reserve5”上限。

运行系统中的访问权限：读和写

语法

Object.**Limit5UpperLimitEnabled**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

BOOLEAN

可选。若要监视 "Reserve5" 上限，则为 TRUE。

参见

Bar (页 285)

Limit5UpperLimitRelative

描述

指定"Reserve5"上限是以百分比还是绝对值进行估算。

运行系统中可进行读写访问

语法

Object.**Limit5UpperLimitRelative**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar

BOOLEAN

可选项。

TRUE，将"Reserve5"上限以百分比形式估算。

FALSE，将"Reserve5"上限以绝对值形式估算。

参见

Bar (页 285)

LimitRangeCollection

说明

在运行系统中无访问权限。

LineAlarmView

说明

在运行系统中无访问权限。

LineBackgroundColor

说明

在运行系统中无访问权限。

LineColor

描述

指定窗口分隔线的颜色。

运行系统中可进行读写访问

语法

Object.**LineColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Color

可选项。用于指定窗口分隔线颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

LineEndShapeStyle

描述

指定线端的形状。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**LineEndShapeStyle**[=LineEndShapeStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox
- Circle

- CircleSegment
- CircularArc
- ComboBox
- Connector
- Ellipse
- EllipseSegment
- EllipticalArc
- GraphicIOField
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton *
- SymbolicIOField *
- TextField
- WindowSlider

* 只读访问权限

在运行系统中您没有以下格式的访问权限：

- Switch
- TubeArcObject

LineEndShapeStyle

可选项。用于指定线端形状的值或常量。

值	描述
0	圆形
1	齐平
2	矩形

参见

Bar (页 285)

Button (页 296)

CheckBox (页 307)

Circle (页 312)

CircleSegment (页 316)

CircularArc (页 320)

ComboBox (页 326)

Connector (页 330)

Ellipse (页 340)

EllipseSegment (页 344)

EllipticalArc (页 348)

GraphicIOField (页 371)

GraphicView (页 375)

IOField (页 382)

Line (页 388)

Listbox (页 392)

MultiLineEdit (页 399)

OptionGroup (页 435)

Polygon (页 444)

Polyline (页 448)

Rectangle (页 467)

RoundButton (页 471)

Switch (页 499)

SymbolicIOField (页 504)

TextField (页 527)

TubeArcObject (页 554)

WindowSlider (页 586)

LinesPerDiagEntry

说明

在运行系统中无访问权限。

LineStyle

说明

在运行系统中无访问权限。

LineWidth

说明

指定线宽。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**LineWidth**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- CircularArc
- Connector
- EllipticalArc
- FunctionTrendControl
- Line *
- OnlineTableControl
- OnlineTrendControl
- Polyline *
- TrendRulerControl
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline
- UserArchiveControl

* RT Advanced 读写访问权限

在运行系统中您没有以下格式的访问权限：

- Polygon

Int32

可选项。用于指定以像素为单位的线宽的值或常量。

参见

AlarmControl (页 255)

CircularArc (页 320)

EllipticalArc (页 348)

FunctionTrendControl (页 351)

Line (页 388)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

Polyline (页 448)

TrendRulerControl (页 532)

TubeArcObject (页 554)

TubeDoubleTeeObject (页 557)

TubePolyline (页 560)

TubeTeeObject (页 563)

用户归档控件 (页 566)

Polygon (页 444)

LineWrap

说明

在运行系统中无访问权限。

ListAreaHeight

说明

在运行系统中无访问权限。

ListAreaLeft

说明

在运行系统中无访问权限。

ListAreaTop

说明

在运行系统中无访问权限。

ListAreaWidth

说明

在运行系统中无访问权限。

LoadDataImmediately

描述

指定在发生画面刷新时是否应从归档中下载所示时间范围内的变量值。

运行系统中的访问权限：读和写

语法

Object.**LoadDataImmediately**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl

BOOLEAN

可选。如果在发生画面刷新时应从归档中下载所示时间范围内的变量值，则为 TRUE。

参见

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

LocalCursor

描述

在运行系统中无访问权限。

Location

说明

在运行系统中无访问权限。

LockSquaredExtent

描述

指定是否可以通过鼠标调节时钟大小。

运行系统中的访问权限：读和写

语法

Object.**LockSquaredExtent**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Clock
- Gauge

BOOLEAN

可选。若通过拖动鼠标选择相应纵横比来调节时钟大小，则为 TRUE。

参见

Clock (页 323)

Gauge (页 366)

Logging

说明

返回“Logging”类型对象。

运行系统中可进行的访问：Read

语法

Object.**Logging**

Object

要求"HMIRuntime"对象。

参见

HMIRuntime（页 224）

LogOperation

描述

指定在操作对象后，是否在报警系统中输出报警。

运行系统中的访问权限：读和写

语法

Object.**LogOperation**[=BOOLEAN]

Object

必选项。具有以下格式的"ScreenItem"类型的对象：

- CheckBox
- ComboBox
- IOField
- ListBox
- OptionGroup
- SymbolicIOField *
- WindowSlider

*：只读访问权限

BOOLEAN

可选。若在操作对象后从报警系统中输出报警，则为 TRUE,。

LongDateTimeFormat

说明

在运行系统中无访问权限。

LongTermArchiveConsistency

说明

指定组态“历史报警列表（长期）”(Historical alarm list (long-term)) 时，报警在报警视图中的显示方式。

运行系统中可进行读写访问

语法

Object.**LongTermArchiveConsistency**[= BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl

BOOLEAN

可选项。

TRUE，所有服务器的客户端或冗余服务器对的“历史报警列表（长期）”(Historical alarm list (long-term))中将显示 1000 条最近的报警。

FALSE，单站系统、服务器或每台服务器的客户端或每个冗余服务器对的“历史报警列表”(Historical alarm list)（长期）中将显示 1000 条最近的报警。

参见

AlarmControl (页 255)

Look3D

说明

在运行系统中无访问权限。

LowerLimit

描述

指定输入值的下限。

运行系统中的访问权限：读和写

语法

Object.**LowerLimit**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- IOField

DOUBLE

可选项。用于指定输入值下限的值或常量。

参见

IOField (页 382)

LowLimitColor

描述

指定滚动条中底部或左侧滚动按钮的颜色。

运行系统中可进行读写访问

语法

Object.**LowLimitColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- WindowSlider

Color

可选项。用于指定滚动条中底部或左侧滚动按钮颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

WindowSlider (页 586)

1.5.5.9 属性 M-N

Machine

说明

在运行系统中无访问权限。

MachineName

描述

指定要监视的设备的程序段代码。

输入设备的名称或端口作为程序段代码。

运行系统中的访问权限：读和写

语法

Object.**MachineName**[=STRING]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- SmartClientView

STRING

可选项。其中包含程序段代码的值或常量。

参见

SmartClientView (页 490)

MaintainAspectRatio

描述

在运行系统中无访问权限。

MaintainOriginalSize

描述

在运行系统中无访问权限。

MarginToBorder

描述

指定 3D 边框的宽度，单位：像素。宽度数值取决于对象的大小。

运行系统中的访问权限：读和写

语法

Object.**MarginToBorder**[=Int32]

Object

必选项。具有以下格式的"ScreenItem"类型的对象：

- WindowSlider

Int32

可选项。用于指定以像素为单位的 3D 边框宽度的值或常量。

参见

WindowSlider (页 586)

MaximumNumberOfTimeAxes

说明

在运行系统中无访问权限。

MaximumNumberOfTimeColumns

说明

在运行系统中无访问权限。

MaximumNumberOfValueAxes

说明

在运行系统中无访问权限。

MaximumNumberOfValueColumns

说明

在运行系统中无访问权限。

MaximumValue

描述

指定所选对象的刻度范围的最大值。

运行系统中的访问权限：读和写

语法

Object.**MaximumValue**[=DOUBLE | Int32 | SINGLE]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- Bar
- Gauge
- Slider
- WindowSlider*

*：只读访问权限

DOUBLE | Int32 | SINGLE

指定最大值的选项 A 值或常量。数据类型取决于格式。

- DOUBLE: Bar
- Int32: Slider, WindowsSlider
- SINGLE：Gauge

MaxNrOfCurves

说明

在运行系统中无访问权限。

MaxNumberOfComboBoxCharacters

说明

在运行系统中无访问权限。

MaxToolbarRows

说明

在运行系统中无访问权限。

MenuButtonVisible

说明

在运行系统中无访问权限。

MenuToolBarConfig

说明

确定包含用户自定义菜单和工具栏的组态文件。

运行系统中可进行的访问：读取和写入

语法

Object.**MenuToolBarConfig**[=HmiObjectHandle]

Object

必选项。具有以下格式的"ScreenItem"类型的对象：

- ScreenWindow

HmiObjectHandle

可选项。用于指定包含用户定义菜单和工具栏的组态文件的值或常量。

MessageAreaHeight

说明

在运行系统中无访问权限。

MessageAreaLeft

说明

在运行系统中无访问权限。

MessageAreaTop

说明

在运行系统中无访问权限。

MessageAreaWidth

说明

在运行系统中无访问权限。

MessageBlockAlignment

描述

指定列标题中使用“MessageBlockIndex”引用的报警文本块的名称。

也可将“ApplyProjectSettings”设置为“FALSE”。

运行系统中可进行读写访问

语法

Object.**MessageBlockAlignment** [=HorizontalAlignment]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl

HorizontalAlignment

可选项。确定对齐方式的值或常量。

值	名称	描述
0	左对齐	列标题内的块名称左对齐。
1	居中	列标题内的块名称居中。
2	右对齐	列标题内的块名称右对齐。

参见

ApplyProjectSettings (页 621)

AlarmControl (页 255)

MessageBlockAutoPrecisions

描述

指定是否自动确定使用“MessageBlockIndex”引用的报警文本块的小数位数。

也可将“ApplyProjectSettings”设置为“FALSE”。

运行系统中可进行读写访问

语法

Object.**MessageBlockAutoPrecisions**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl

BOOLEAN

可选项。

TRUE，自动确定小数位数。

FALSE，使用“MessageBlockPrecisions”值。

参见

MessageBlockIndex (页 963)

ApplyProjectSettings (页 621)

AlarmControl (页 255)

MessageBlockCaption

描述

指定使用"MessageBlockIndex"引用的报警文本块的列标题内的块名称。输入的名称对所有运行系统语言有效。

也可将"ApplyProjectSettings"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**MessageBlockCaption**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

STRING

可选项。指定所选报警文本块的列标题内的名称。

参见

ApplyProjectSettings (页 621)

AlarmControl (页 255)

MessageBlockCount

描述

指定已组态的报警文本块的数目。

运行系统中可进行读写访问

语法

Object.**MessageBlockCount**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

Int32

可选项。用于指定已组态的报警文本块的值或常量。

参见

AlarmControl (页 255)

MessageBlockDateFormat

描述

指定用于显示报警的日期格式。

也可将"ApplyProjectSettings"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**MessageBlockDateFormat**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

STRING

可选项。指定用于显示报警的日期格式的值或常量。

值	描述
自动	自动设置日期格式。
dd.MM.yy	日.月.年，例如，24.12.10。
dd.MM.yyyy	日.月.年，例如，24.12.2010。
dd/MM/yy	日/月/年，例如，24/12/10。
dd/MM/yyyy	日/月/年，例如，24/12/2010。

参见

ApplyProjectSettings (页 621)

AlarmControl (页 255)

MessageBlockExponentialFormat

说明

指定是否以指数计数法显示使用“MessageBlockIndex”引用的报警文本块的值。

也可将“ApplyProjectSettings”设置为“FALSE”。

运行系统中可进行读写访问

语法

Object.**MessageBlockExponentialFormat**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl

BOOLEAN

可选项。

TRUE，以指数计数法显示值。

FALSE，以十进制计数法显示值。

参见

ApplyProjectSettings (页 621)

AlarmControl (页 255)

MessageBlockFlashOn

描述

指定当出现报警时使用“MessageBlockIndex”引用的报警文本块内容是否会在运行系统中闪烁。

也可将“ApplyProjectSettings”设置为“FALSE”。

运行系统中可进行读写访问

语法

Object.**MessageBlockFlashOn**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

BOOLEAN

可选项。

TRUE，报警文本块的内容闪烁。

FALSE，报警文本块的内容不闪烁。

参见

ApplyProjectSettings (页 621)

AlarmControl (页 255)

MessageBlockHideText

描述

指定是否将使用"MessageBlockIndex"引用的报警文本块的内容显示为文本。

也可将"ApplyProjectSettings"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**MessageBlockHideText**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

BOOLEAN

可选项。

TRUE，内容不以文本形式显示。

FALSE，内容以文本形式显示。

参见

ApplyProjectSettings (页 621)

AlarmControl (页 255)

MessageBlockHideTitleText

描述

指定是否将使用"MessageBlockIndex"引用的报警文本块的标题显示为文本。

也可将"ApplyProjectSettings"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**MessageBlockHideTitleText**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

BOOLEAN

可选项。

TRUE，标题不以文本形式显示。

FALSE，标题以文本形式显示。

参见

AlarmControl (页 255)

MessageBlockId

描述

指定报警视图中 ID 编号和报警文本块的分配。

运行系统中的访问权限：读和写

语法

Object.**MessageBlockId**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmControl

Int32

可选项。指定报警视图中 ID 编号和报警文本块的分配。

参见

AlarmControl (页 255)

MessageBlockIndex

描述

引用报警文本块。要访问报警文本块的属性，需要设置 "MessageBlockIndex"。

介于 0 至 (MessageBlockIndex - 1) 之间的值为“MessageBlockCount”的有效值。“MessageBlockCount”属性可指定可组态的报警文本块数量。

运行系统中可进行读写访问

语法

Object.**MessageBlockIndex**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl

Int32

可选项。用于通过索引指定要编辑的报警文本块的值或常量。

参见

MessageBlockCount (页 958)

AlarmControl (页 255)

MessageBlockLeadingZeros

描述

指定是否以前导零的方式显示使用"MessageBlockIndex"引用的报警文本块中的条目。

也可将"ApplyProjectSettings"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**MessageBlockLeadingZeros**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

Int32

可选项。

1，以前导零的方式显示报警文本块中的条目。

0，以不带前导零的方式显示报警文本块中的条目。

参见

ApplyProjectSettings (页 621)

AlarmControl (页 255)

MessageBlockLength

描述

指定使用"MessageBlockIndex"引用的报警文本块内容的字符长度。

也可将"ApplyProjectSettings"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**MessageBlockLength**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

Int32

可选项。用于指定报警文本块长度的值。

参见

ApplyProjectSettings (页 621)

AlarmControl (页 255)

MessageBlockName

描述

指定使用"MessageBlockIndex"引用的报警文本块的名称。

运行系统中的访问权限：读和写

语法

Object.**MessageBlockName**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

STRING

可选项。用于指定引用的报警文本块名称的值或常数。

MessageBlockPrecisions

描述

指定使用"MessageBlockIndex"引用的报警文本块中小数位数的值。

也可将"ApplyProjectSettings"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**MessageBlockPrecisions**[=Int16]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

Int16

可选项。指定小数位数。

参见

ApplyProjectSettings (页 621)

AlarmControl (页 255)

MessageBlockSelected

描述

指定是否已选择使用"MessageBlockIndex"引用的报警文本块。

也可将 "ApplyProjectSettings (页 621)"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**MessageBlockSelected**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

BOOLEAN

可选项。TRUE，已选报警文本块。

参见

AlarmControl (页 255)

MessageBlockShowDate

描述

指定是否在 "时间"(Time) 报警文本块中同时显示日期和时间。

也可将"ApplyProjectSettings"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**MessageBlockShowDate**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

BOOLEAN

可选项。

TRUE，日期和时间同时显示。

FALSE，仅显示时间。

参见

ApplyProjectSettings (页 621)

AlarmControl (页 255)

MessageBlockShowIcon

描述

指定是否以图标形式显示使用“MessageBlockIndex”引用的报警文本块的内容。

也可将“ApplyProjectSettings”设置为“FALSE”。

运行系统中可进行读写访问

语法

Object.**MessageBlockShowIcon**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl

BOOLEAN

可选项。

TRUE，内容以图标形式显示。

FALSE，内容不以图标形式显示。

参见

ApplyProjectSettings (页 621)

AlarmControl (页 255)

MessageBlockShowTitleIcon

描述

指定是否将使用“MessageBlockIndex”引用的报警文本块的标题显示为文本。

也可将“ApplyProjectSettings”设置为“FALSE”。

运行系统中可进行读写访问

语法

Object.**MessageBlockShowTitleIcon**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

BOOLEAN

可选项。

TRUE，标题以图标形式显示。

FALSE，标题不以图标形式显示。

参见

ApplyProjectSettings (页 621)

AlarmControl (页 255)

MessageBlockTextId

描述

指定使用文本库中的文本 ID 并通过"MessageBlockIndex"引用的报警文本块的名称。如果运行系统语言发生更改，则会自动调整该名称。

也可将"ApplyProjectSettings"设置为"FALSE"。

运行系统中的访问权限：读和写

语法

Object.**MessageBlockTextId**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

Int32

可选项。使用文本 ID 编号指定选定报警文本块的名称。

MessageBlockTimeFormat

描述

指定用于显示报警的时间格式或持续时间格式。

也可将"ApplyProjectSettings"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**MessageBlockTimeFormat**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

STRING

可选项。指定用于显示报警的日期格式或持续时间格式的值或常量。

下列时间格式可用：

值	描述
自动	自动设置时间格式。
HH:mm:ss	时:分:秒，例如 15:35:44
HH:mm:ss.ms	时:分:秒.毫秒，例如 15:35:44.240。
hh:mm:ss tt	时:分:秒 AM/PM，例如 03:35:44 PM。
hh:mm:ss.ms tt	时:分:秒.毫秒 AM/PM，例如 03:35:44.240 PM。

下列持续时间格式可用：

值	描述
自动	持续时间格式是自动确定的。
d H:mm:ss	日 小时:分钟:秒，例如 1 2:03:55。
H:mm:ss.	小时:分钟:秒，例如 26:03:55。

值	描述
m:ss	分钟:秒，例如：1563:55。
s	秒，例如 93835。

参见

ApplyProjectSettings (页 621)

AlarmControl (页 255)

MessageBlockType

描述

返回使用"MessageBlockIndex"引用的报警文本块的类型。

运行系统中可进行读写访问

语法

Object.**MessageBlockType**[=AlarmBlockType]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

AlarmBlockType

可选项。返回报警文本块类型的值或常量。

值	说明
0	报警文本块为系统块。
1	报警文本块为用户文本块。
2	报警文本块为参数块。
3	报警文本块属于报警统计的报警文本块。

参见

ApplyProjectSettings (页 621)

AlarmControl (页 255)

MessageColumnAdd

描述

创建新的报警文本块。新创建的报警文本块会自动使用“MessageColumnIndex”而引用。

运行系统中可进行读写访问

语法

Object.**MessageColumnAdd**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl

STRING

必需项。用于通过“MessageColumnName”指定新报警文本块名称的值或常数。

参见

MessageColumnIndex (页 973)

AlarmControl (页 255)

MessageColumnCount

描述

指定已组态的报警文本块的数目。

运行系统中可进行读写访问

语法

Object.**MessageColumnCount**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl

Int32

可选项。用于指定可组态的报警文本块的值或常量。

参见

AlarmControl (页 255)

MessageColumnIndex

描述

引用为报警列表选择的报警文本块。要访问报警文本块的属性，需要设置 "MessageColumnIndex"。

介于 0 至 (MessageColumnIndex - 1) 之间的值为"MessageColumnCount"的有效值。"MessageColumnCount"属性可指定可组态的报警文本块数量。

运行系统中可进行读写访问

语法

Object.**MessageColumnIndex**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

Int32

可选项。用于通过索引指定要编辑的报警文本块的值或常量。

参见

MessageColumnCount (页 972)

AlarmControl (页 255)

MessageColumnName

描述

指定使用"MessageColumnIndex"引用的报警文本块的名称。

运行系统中可进行读写访问

语法

Object.**MessageColumnName**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

STRING

可选项。用于指定引用的报警文本块名称的值或常数。

参见

MessageColumnIndex (页 973)

AlarmControl (页 255)

MessageColumnRemove

描述

使用引用的报警文本块名称可移除该报警文本块。

运行系统中可进行读写访问

语法

Object.**MessageColumnRemove**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

STRING

可选项。用于指定待移除的引用的报警文本块名称的值或常量。

参见

AlarmControl (页 255)

MessageColumnRepos

描述

对于多个报警文本块，指定使用“MessageColumnIndex”引用的报警文本块位置。

如果已使用“MessageColumnRepos”更改了报警文本块位置，“MessageColumnRepos”的值则会分配给“MessageColumnIndex”。

运行系统中可进行读写访问

语法

Object.**MessageColumnRepos**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl

Int32

可选项。用于指定引用的报警文本块位置的值或常量。值范围为0到(MessageColumnCount - 1)。超出该范围的值无效。

0：引用的报警文本块放置在左侧。

参见

MessageColumnIndex (页 973)

MessageColumnCount (页 972)

AlarmControl (页 255)

MessageColumnSort

描述

指定是否将使用“MessageColumnIndex”引用的报警文本块的内容显示为文本。

运行系统中可进行读写访问

语法

Object.**MessageColumnSort**[=SortMode]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

SortMode

可选项。用于指定引用的报警文本块的排序的值或常量。

值	说明
0	无排序
1	从最小值到最大值升序排列
2	从最大值到最小值降序排列

参见

AlarmControl (页 255)

MessageColumnSortIndex

描述

定义"MessageColumnIndex"中引用的报警文本块的排序顺序。 如果将该值设置为"0"，将从"MessageColumnSort"中移除排序标准。

运行系统中的访问权限：读和写

语法

Object.**MessageColumnSortIndex**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmControl

Int32

可选项。定义"MessageColumnIndex"中引用的报警文本块的排序顺序。

参见

AlarmControl (页 255)

MessageColumnVisible

描述

指定是否显示使用"MessageColumnIndex"引用的报警文本块。

运行系统中可进行读写访问

语法

Object.**MessageColumnVisible**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

BOOLEAN

可选项。

TRUE，显示引用的报警文本块。

FALSE，不显示引用的报警文本块。

参见

MessageColumnIndex (页 973)

AlarmControl (页 255)

MessageListType

描述

指定要显示的内容。

运行系统中可进行读写访问

语法

Object.**MessageListType**[=AlarmListType]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

AlarmListType

可选项。用于指定内容的值或常量。

值	描述
0	当前激活的报警随屏幕刷新显示。
1	"历史报警列表（短期）"(historical alarm list (short-term)) 显示随屏幕刷新激活。报警视图显示已记录的报警和当前未决的报警。 新消息激活时会立即更新该显示。
2	"历史报警列表（长期）"(historical alarm list (long-term)) 显示在屏幕更改后激活。报警视图仅显示已记录的报警，用户可对其进行注释。
3	屏幕更改后仅显示当前锁定的报警。
4	为已记录的报警组态的统计计算在屏幕更改后显示。
5	在屏幕更改后仅显示之前隐藏的报警。

参见

AlarmControl (页 255)

MiddleGradientColor

说明

在运行系统中无访问权限。

MinimumNumberOfTimeAxes

说明

在运行系统中无访问权限。

MinimumNumberOfTimeColumns

说明

在运行系统中无访问权限。

MinimumNumberOfValueAxes

说明

在运行系统中无访问权限。

MinimumNumberOfValueColumns

说明

在运行系统中无访问权限。

MinimumValue

描述

指定所选对象的刻度范围的最小值。

运行系统中的访问权限：读和写

语法

Object.**MinimumValue**[=DOUBLE | Int32 | SINGLE]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- Bar
- Gauge
- Slider
- WindowSlider*

*：只读访问权限

DOUBLE | Int32 | SINGLE

指定最小值的选项 A 值或常量。数据类型取决于格式。

- DOUBLE: Bar
- Int32: Slider, WindowsSlider
- SINGLE：Gauge

MinNrOfCurves

说明

在运行系统中无访问权限。

MinPasswordValue

说明

在运行系统中无访问权限。

MinuteNeedleHeight

描述

指定分针的长度。

运行系统中可进行读写访问

语法

Object.**MinuteNeedleHeight**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Clock

Int32

可选项。用于指定分针长度的值或常量。

指定分针的长度，作为与钟盘半径相关的百分值。

参见

Clock (页 323)

MinuteNeedleWidth

描述

指定分针的宽度。

运行系统中可进行读写访问

语法

Object.**MinuteNeedleWidth**[=Int32]

Object

必需项。具有以下格式的""ScreenItem"类型的对象：

- Clock

Int32

可选项。用于指定分针宽度的值或常量。

以两倍分针长度百分比的形式指定分针的宽度。

参见

Clock (页 323)

Mode

描述

指定运行系统中文本对象的行为模式。

运行系统中的访问权限

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**Mode**[=Type]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Button
- IOField
- RoundButton*
- SymbolicIOField*

*：只读访问权限

在运行系统中您没有以下格式的访问权限：

- DateTimeField
- GraphicIOField
- Switch

Type

可选项。用于指定运行系统中文本对象的行为模式的值或常量。

值	VB 常量	说明
0	hmiIOFieldOutput	输出域
1	hmiIOFieldInput	输入域
2	hmiIOFieldInOutput	输入和输出域

MonitorNumber

说明

返回在画面窗口中显示其内容的显示器的编号。
显示器编号是指在 Microsoft Windows 中设置的显示器编号："控制面板 > 显示 > 设置 > 显示"

运行系统中的访问权限：读和写

语法

Object.**MonitorNumber**[=Int32]

Object

必需项。“ScreenItem”对象，且具有以下特性：

- Screenwindow

Int32

可选项。用于指定所显示监视器的编号的值或常量。

参见

ScreenWindow (页 479)

Moveable

说明

指定关于运行系统中是否可以移动对象的信息。

运行系统中可进行读写访问

语法

Object.**Moveable**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，窗口可以在运行系统中移动。

FALSE，窗口在运行系统中固定。

参见

AlarmControl (页 255)

用户归档控件 (页 566)

TrendRulerControl (页 532)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

MsgFilterSQL

描述

针对用户自选的报警指定一个或多个 SQL 语句。多个自定义选择采用"OR"连接。若组态了固定选择"DefaultMsgFilterSQL"，则"DefaultMsgFilterSQL"和"MsgFilterSQL"的 SQL 语句采用"AND"连接。

运行系统中的访问权限：读和写

语法

Object.**MsgFilterSQL**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

STRING

可选项。用于指定用户自定义消息选择的 SQL 语句的值或常量。

参见

AlarmControl (页 255)

Name

说明

返回 STRING 形式的对象名称。返回的值取决于所使用的对象。

运行系统中的访问权限：读和写

语法

Object.**Name**[=STRING]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- Clock
- DiscSpaceView *
- Gauge
- Slider

* 只读访问权限

在运行系统中您没有以下格式的访问权限：

- AlarmControl
- AlarmView
- ApplicationWindow
- BatteryView
- Bar
- Button
- ChannelDiagnose
- CheckBox
- Circle
- CircleSegment
- CircularArc
- ComboBox
- Connector

- DateTimeField
- Ellipse
- EllipseSegment
- EllipticalArc
- FunctionTrendControl
- GraphicIOField
- GraphicView
- HTMLBrowser
- IOField
- Line
- ListBox
- MediaPlayer
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- OptionGroup
- PLCCodeViewer
- ProDiagOverview
- Polygon
- Polyline
- ProtectedAreaNameView
- RangeLabelView
- RangeQualityView
- RecipeView
- Rectangle
- RoundButton
- S7GraphOverview
- ScreenWindow
- SmartClientView

- StatusForce
- Switch
- SymbolLibrary
- SymbolicIOField
- SysDiagControl
- TextField
- TrendRulerControl
- TrendView
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline
- UserArchiveControl
- UserView
- WLanQualityView
- WindowSlider
- ZoneLabelView
- ZoneQualityView

String

可选项。用于返回对象名称的值或常量。

注释

根据指定对象的不同，将返回以下对象名称：

- Tag：不含服务器或变量前缀的变量名称。
- Project：当前运行系统项目的名称。
- DataItem：DataItem 对象的名称。

- Layer：层名称。
- FunctionTrendControl：“Index”属性引用的趋势名称。

说明

可以使用“Name”属性寻址“Tags”列表中的变量。在 WinCC 中，变量名称的结构基于以下方案：
<变量前缀><变量名称>
如果仅指定变量名称，将通过画面窗口快捷方式应用变量前缀。

示例

以下示例将当前运行系统项目的名称作为 Trace 返回：

表格 1-129

```
'VBS160
HMIRuntime.Trace "Name: " & HMIRuntime.ActiveProject.Name & vbNewLine
```

NameColumnWidth

说明

在运行系统中无访问权限。

NavigateTo

说明

指定要跳转至的模块。

运行系统中的访问权限：读和写

语法

Object.**NavigateTo**[=String]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- PLCCodeViewer
- SysDiagControl *

* RT Advanced 无访问权限，RT Professional 只读访问权限

String

可选项。用于指定模块的值或常量。

NavigationButtons

说明

在运行系统中无访问权限。

NavigationPath_Font

说明

在运行系统中无访问权限。

NavigationPath_RootText

说明

在运行系统中无访问权限。

NavigationPath_TextColor

说明

在运行系统中无访问权限。

NavigationpathDiagbufferDetailText

说明

在运行系统中无访问权限。

NavigationpathDiagbufferText

说明

在运行系统中无访问权限。

NeedleBorderColor

描述

指定指针的线颜色。

运行系统中可进行读写访问

语法

Object.**NeedleBorderColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Clock

Color

可选项。用于指定指针线颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Clock (页 323)

NeedleColor

描述

指定指针颜色。

也可通过""NeedleFillStyle"指定将要显示的指针颜色。

运行系统中可进行读写访问

语法

Object.**NeedleColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Clock

Color

可选项。用于指定指针颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Clock (页 323)

NeedleFillStyle

描述

指定指针显示为填充样式还是透明样式：

运行系统中可进行读写访问

语法

Object.**NeedleFillStyle**[=THmiFillStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Clock

THmiFillStyle

可选项。用于指定指针是否为填充样式或透明样式的值或常量。

值	VB 常量	说明
0	hmiFillStyleSolid	指针显示为透明，其通过前景色显示的边界来指示。
65536	hmiFillStyleTransparent	以指针填充色填充指针，指针边界显示为前景色。

参见

Clock (页 323)

NeedleHeight

说明

在运行系统中无访问权限。

NoHitTest

描述

在运行系统中无访问权限。

NormalColor

说明

指定内存空间利用率在非临界范围时，内存空间视图中已用内存的颜色。

运行系统中的访问权限：读取

语法

Object.**NormalColor**[=Color]

Object

必需项。"ScreenItem"对象，且具有以下特性：

- DiscSpaceView

Color

可选项。用于指定正常范围内内存显示颜色的值或常量。

注释

使用"RGB"函数来定义 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值指定相应的十进制值（值范围为 0 至 255）。"红色"显示如下：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

DiskSpaceView (页 337)

NormalRangeColor

描述

指定"Gauge"对象正常刻度范围的颜色。

若要显示正常范围，"NormalRangeVisible"属性值必须为 TRUE。

运行系统中可进行读写访问

语法

Object.**NormalRangeColor**[=Color]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Gauge

Color

可选项。用于指定正常范围颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Gauge (页 366)

NormalRangeVisible

描述

指定是否显示"Gauge"对象的正常刻度范围。

运行系统中的访问权限：读和写

语法

Object.**NormalRangeVisible**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Gauge

BOOLEAN

可选。若显示正常刻度范围，则为 TRUE。

注释

指定"NormalRangeColor"属性的正常范围的颜色。

参见

Gauge (页 366)

NumberOfButtons

描述

在运行系统中无访问权限。

NumberOfLines

说明

在运行系统中无访问权限。

NumberOfVisibleLines

说明

指定 PLC 代码显示中的可见行数。

运行系统中的访问权限：读和写

语法

Object.**NumberOfVisibleLines**[=Int32]

Object

必需项。 "ScreenItem"对象，且具有以下特性：

- PLCCodeViewer

Int32

可选项。 用于指定 PLC 代码显示中的行数的值或常量。

参见

PLCCodeViewer (页 442)

NumberStyle

说明

在运行系统中无访问权限。

1.5.5.10 属性 O-P

Object

说明

在运行系统中无访问权限。

ObjectSizeDeclutteringEnable

说明

指定是否只有在设定的尺寸范围内的对象才显示。

运行中可进行的访问：读取

语法

Object.**ObjectSizeDeclutteringEnable**

Object

必需。 “Screen”对象。

示例

该示例显示的是将画面"NewPDL1"的"Decluttering"属性作为 Trace 输出。

```
'VBS157
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
HMIRuntime.Trace "Min: " & objScreen.ObjectSizeDeclutteringMin & vbNewLine
HMIRuntime.Trace "Max: " & objScreen.ObjectSizeDeclutteringMax & vbNewLine
HMIRuntime.Trace "Enable: " & objScreen.LayerDeclutteringEnable & vbNewLine
```

参见

Screen (页 231)

ObjectSizeDeclutteringMax

说明

将指定画面的对象显示取消的上限范围返回为 LONG。

运行中可进行的访问：读取

语法

Object.**ObjectSizeDeclutteringMax**

Object

必需。 "Screen"对象。

示例

该示例显示的是将画面"NewPDL1"的"Decluttering"属性作为 Trace 输出。

```
'VBS157
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
HMIRuntime.Trace "Min: " & objScreen.ObjectSizeDeclutteringMin & vbNewLine
HMIRuntime.Trace "Max: " & objScreen.ObjectSizeDeclutteringMax & vbNewLine
HMIRuntime.Trace "Enable: " & objScreen.LayerDeclutteringEnable & vbNewLine
```

参见

Screen (页 231)

ObjectSizeDeclutteringMin

说明

将指定画面的对象显示取消的下限范围返回为 LONG。

运行中可进行的访问：读取

语法

Object.**ObjectSizeDeclutteringMin**

Object

必需。 “Screen”对象。

示例

该示例显示的是将画面“NewPDL1”的“Decluttering”属性作为 Trace 输出。

```
'VBS157
Dim objScreen
Set objScreen = HMIRuntime.Screens("NewPDL1")
HMIRuntime.Trace "Min: " & objScreen.ObjectSizeDeclutteringMin & vbNewLine
HMIRuntime.Trace "Max: " & objScreen.ObjectSizeDeclutteringMax & vbNewLine
HMIRuntime.Trace "Enable: " & objScreen.LayerDeclutteringEnable & vbNewLine
```

参见

Screen (页 231)

OcxGuid

说明

在运行系统中无访问权限。

OCXState

描述

在运行系统中无访问权限。

OcxStateForEs2Rt

说明

在运行系统中无访问权限。

Online

描述

指定更新的起始与停止。

运行系统中的访问权限：读和写

语法

Object.**Online**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl

BOOLEAN

可选项。

如果停止已更新的显示，则为 TRUE。再次单击该按钮时，会缓冲并更新相关值。

如果继续已更新的显示，则为 FALSE。

参见

OnlineTrendControl (页 418)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnValue

说明

在运行系统中无访问权限。

OperationSteps

描述

指定单击鼠标一次滚动条滑块移动的步数。

运行系统中的访问权限：读和写

语法

Object.**OperationSteps**[=Int32]

Object

必需项。具有以下格式的 "ScreenItem" 类型的对象：

- WindowSlider

Int32

可选项。用于指定单击一次鼠标时滚动条的滑块移动步数的值或常量。

参见

WindowSlider (页 586)

OperatorAlarms

说明

在运行系统中无访问权限。

OperatorMessageId

描述

指定报警视图中 ID 编号和触发事件的分配。

运行系统中的访问权限：读和写

语法

Object.**OperatorMessageId**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

Int32

可选项。用于指定报警视图中 ID 编号和触发事件的分配的值或常量。

值	描述	描述
0	锁定	触发事件“锁定”
1	解锁	触发事件“解锁”
2	隐藏	触发事件“隐藏”
3	取消隐藏	触发事件“取消隐藏”
4	确认	触发事件“确认”

参见

AlarmControl (页 255)

OperatorMessageIndex

描述

引用操作员消息事件。使用此属性可以将其它属性的值分配给特定的操作员消息。

运行系统中的访问权限：读和写

语法

Object.**OperatorMessageIndex**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

Int32

可选项。用于指定操作员输入报警的报警事件的值或常量。

值	描述
0	消息事件“锁定”
1	消息事件“解锁”
2	消息事件“隐藏”
3	消息事件“取消隐藏”
4	消息事件“确认”

参见

AlarmControl (页 255)

OperatorMessageName

描述

指定消息事件中由"OperatorMessageIndex"事件引用的操作员消息的名称。

运行系统中的访问权限：读和写

语法

Object.**OperatorMessageName**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

STRING

可选项。用于指定报警事件中使用"OperatorMessageIndex"事件引用的操作员输入报警的名称的值或常量。

值	描述
Lock	消息事件“锁定”
Unlock	消息事件“启用”
Hide	消息事件“隐藏”
Unhide	消息事件“取消隐藏”
Quit	消息事件“确认”

参见

AlarmControl (页 255)

OperatorMessageNumber

描述

在*不*使用 WinCC 操作员消息的情况下为所选消息事件的操作员消息指定消息编号。

运行系统中的访问权限：读和写

语法

Object.**OperatorMessageNumber**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

Int32

可选项。为所选消息事件的操作员消息指定消息编号。

参见

AlarmControl (页 255)

OperatorMessageSelected

描述

激活列表中可以触发操作员消息的消息事件。

运行系统中的访问权限：读和写

语法

Object.**OperatorMessageSelected**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

BOOLEAN

可选项。

TRUE，激活列表中可以触发操作员输入报警的报警事件事件。

参见

AlarmControl (页 255)

OperatorMessageSource1

描述

指定要添加到在此处组态的操作员消息的 "过程值块 1"的操作消息的报警文本块。

运行系统中的访问权限：读和写

语法

Object.**OperatorMessageSource1**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

STRING

可选项。指定要添加到在此处组态的操作员消息的"过程值块 1"的操作消息的报警文本块。

示例

将生成操作员消息以指示锁定了一条消息。已锁定消息的"用户文本块 1"的内容，（例如，"电机出现故障"）应显示在操作员消息的"过程值块 1"中。在过程值中选择"1"作为操作报警"用户文本块 1"的报警文本块锁。

参见

AlarmControl (页 255)

OperatorMessageSource2

描述

指定要添加到在此处组态的操作员消息的"过程值块 2"的操作消息的报警文本块。

运行系统中的访问权限：读和写

语法

Object.**OperatorMessageSource2**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

STRING

可选项。指定要添加到在此处组态的操作员消息的"过程值块 2"的操作消息的报警文本块。

示例

将生成操作员消息以指示锁定了一条消息。已锁定消息的“用户文本块 1”的内容，（例如，“电机出现故障”）应显示在操作员消息的“过程值块 1”中。在过程值中选择“2”作为操作报警“用户文本块 1”的报警文本块。

参见

AlarmControl (页 255)

OperatorMessageSource3

描述

指定要添加到在此处组态的操作员消息的“过程值块 3”的操作消息的报警文本块。

运行系统中的访问权限：读和写

语法

Object.**OperatorMessageSource3**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl

STRING

可选项。指定要添加到在此处组态的操作员消息的“过程值块 3”的操作消息的报警文本块。

示例

将生成操作员消息以指示锁定了一条消息。已锁定消息的“用户文本块 1”的内容，（例如，“电机出现故障”）应显示在操作员消息的“过程值块 1”中。在过程值中选择“3”作为操作报警“用户文本块 1”的报警文本块。

参见

AlarmControl (页 255)

OperatorMessageSource4

描述

指定要添加到在此处组态的操作员消息的“过程值块 4”的操作消息的报警文本块。

运行系统中的访问权限：读和写

语法

Object.**OperatorMessageSource4**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl

STRING

可选项。指定要添加到在此处组态的操作员消息的“过程值块 4”的操作消息的报警文本块。

示例

将生成操作员消息以指示锁定了一条消息。已锁定消息的“用户文本块 1”的内容，（例如，“电机出现故障”）应显示在操作员消息的“过程值块 1”中。在过程值中选择“4”作为操作报警“用户文本块 1”的报警文本块。

参见

AlarmControl (页 255)

OperatorMessageSource5

描述

指定要添加到在此处组态的操作员消息的“过程值块 5”的操作消息的报警文本块。

运行系统中的访问权限：读和写

语法

Object.**OperatorMessageSource5**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

STRING

可选项。指定要添加到在此处组态的操作员消息的"过程值块 5"的操作消息的报警文本块。

示例

将生成操作员消息以指示锁定了一条消息。已锁定消息的"用户文本块 1"的内容，（例如，"电机出现故障"）应显示在操作员消息的"过程值块 1"中。在过程值中选择"5"作为操作报警"用户文本块 1"的报警文本块。

参见

AlarmControl (页 255)

OperatorMessageSource6

描述

指定要添加到在此处组态的操作员消息的"过程值块 6"的操作消息的报警文本块。

运行系统中的访问权限：读和写

语法

Object.**OperatorMessageSource6**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

STRING

可选项。指定要添加到在此处组态的操作员消息的"过程值块 6"的操作消息的报警文本块。

示例

将生成操作员消息以指示锁定了一条消息。已锁定消息的“用户文本块 1”的内容，（例如，“电机出现故障”）应显示在操作员消息的“过程值块 1”中。在过程值中选择“6”作为操作报警“用户文本块 1”的报警文本块。

参见

AlarmControl (页 255)

OperatorMessageSource7

描述

指定要添加到在此处组态的操作员消息的“过程值块 7”的操作消息的报警文本块。

运行系统中的访问权限：读和写

语法

Object.**OperatorMessageSource7**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl

STRING

可选项。指定要添加到在此处组态的操作员消息的“过程值块 7”的操作消息的报警文本块。

示例

将生成操作员消息以指示锁定了一条消息。已锁定消息的“用户文本块 1”的内容，（例如，“电机出现故障”）应显示在操作员消息的“过程值块 1”中。在过程值中选择“7”作为操作报警“用户文本块 1”的报警文本块。

参见

AlarmControl (页 255)

OperatorMessageSource8

描述

指定要添加到在此处组态的操作员消息的“过程值块 8”的操作消息的报警文本块。

运行系统中的访问权限：读和写

语法

Object.**OperatorMessageSource8**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl

STRING

可选项。指定要添加到在此处组态的操作员消息的“过程值块 8”的操作消息的报警文本块。

示例

将生成操作员消息以指示锁定了一条消息。已锁定消息的“用户文本块 1”的内容，（例如，“电机出现故障”）应显示在操作员消息的“过程值块 1”中。在过程值中选择“8”作为操作报警“用户文本块 1”的报警文本块。

参见

AlarmControl (页 255)

OperatorMessageSource9

描述

指定要添加到在此处组态的操作员消息的“过程值块 9”的操作消息的报警文本块。

运行系统中的访问权限：读和写

语法

Object.**OperatorMessageSource9**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

STRING

可选项。指定要添加到在此处组态的操作员消息的"过程值块 9"的操作消息的报警文本块。

示例

将生成操作员消息以指示锁定了一条消息。已锁定消息的"用户文本块 1"的内容，（例如，"电机出现故障"）应显示在操作员消息的"过程值块 1"中。在过程值中选择"9"作为操作报警"用户文本块 1"的报警文本块。

参见

AlarmControl (页 255)

OperatorMessageSource10

说明

指定要添加到在此处组态的操作员消息的"过程值块 10"的操作消息的报警文本块。

运行系统中的访问权限：读和写

语法

Object.**OperatorMessageSource10**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

STRING

可选项。指定要添加到在此处组态的操作员消息的"过程值块 10"的操作消息的报警文本块。

示例

将生成操作员消息以指示锁定了一条消息。已锁定消息的“用户文本块 1”的内容，（例如，“电机出现故障”）应显示在操作员消息的“过程值块 1”中。在过程值中选择“10”作为操作报警“用户文本块 1”的报警文本块。

参见

AlarmControl (页 255)

OperatorMessageSourceType1

描述

指定操作员输入报警中的报警文本块内容是以文本形式是传送还是以值形式传送。

运行系统中可进行读写访问

语法

Object.**OperatorMessageSourceType1**[=TransferAs]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl

TransferAs

可选项。用于指定报警文本块在操作员输入报警中的显示方式的值或常量。

有下列设置可用：

值	名称	描述
0	文本	报警文本块以文本形式显示。
1	值	报警文本块以值形式显示。

参见

AlarmControl (页 255)

OperatorMessageSourceType2

描述

指定操作员输入报警中的报警文本块内容是以文本形式是传送还是以值形式传送。

运行系统中可进行读写访问

语法

Object.**OperatorMessageSourceType2**[=TransferAs]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

TransferAs

可选项。用于指定报警文本块在操作员输入报警中的显示方式的值或常量。

有下列设置可用：

值	名称	描述
0	文本	报警文本块以文本形式显示。
1	值	报警文本块以值形式显示。

参见

AlarmControl (页 255)

OperatorMessageSourceType3

描述

指定操作员输入报警中的报警文本块内容是以文本形式传送还是以值形式传送。

运行系统中的访问权限：读和写

语法

Object.**OperatorMessageSourceType3**[=TransferAs]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

TransferAs

可选项。用于指定报警文本块在操作员输入报警中的显示方式的值或常量。

有下列设置可用：

值	名称	描述
0	文本	报警文本块以文本形式显示。
1	值	报警文本块以值形式显示。

OperatorMessageSourceType4

描述

指定操作员输入报警中的报警文本块内容是以文本形式是传送还是以值形式传送。

运行系统中可进行读写访问

语法

Object.**OperatorMessageSourceType4**[=TransferAs]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

TransferAs

可选项。用于指定报警文本块在操作员输入报警中的显示方式的值或常量。

有下列设置可用：

值	名称	描述
0	文本	报警文本块以文本形式显示。
1	值	报警文本块以值形式显示。

参见

AlarmControl (页 255)

OperatorMessageSourceType5

描述

指定操作员输入报警中的报警文本块内容是以文本形式是传送还是以值形式传送。

运行系统中可进行读写访问

语法

Object.**OperatorMessageSourceType5**[=TransferAs]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

TransferAs

可选项。用于指定报警文本块在操作员输入报警中的显示方式的值或常量。

有下列设置可用：

值	名称	描述
0	文本	报警文本块以文本形式显示。
1	值	报警文本块以值形式显示。

参见

AlarmControl (页 255)

OperatorMessageSourceType6

描述

指定操作员输入报警中的报警文本块内容是以文本形式是传送还是以值形式传送。

运行系统中可进行读写访问

语法

Object.**OperatorMessageSourceType6**[=TransferAs]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

TransferAs

可选项。用于指定报警文本块在操作员输入报警中的显示方式的值或常量。

有下列设置可用：

值	名称	描述
0	文本	报警文本块以文本形式显示。
1	值	报警文本块以值形式显示。

参见

AlarmControl (页 255)

OperatorMessageSourceType7

描述

指定操作员输入报警中的报警文本块内容是以文本形式是传送还是以值形式传送。

运行系统中可进行读写访问

语法

Object.**OperatorMessageSourceType7**[=TransferAs]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

TransferAs

可选项。用于指定报警文本块在操作员输入报警中的显示方式的值或常量。

有下列设置可用：

值	名称	描述
0	文本	报警文本块以文本形式显示。
1	值	报警文本块以值形式显示。

参见

AlarmControl (页 255)

OperatorMessageSourceType8

描述

指定操作员输入报警中的报警文本块内容是以文本形式是传送还是以值形式传送。

运行系统中可进行读写访问

语法

Object.**OperatorMessageSourceType8**[=TransferAs]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

TransferAs

可选项。用于指定报警文本块在操作员输入报警中的显示方式的值或常量。

有下列设置可用：

值	名称	描述
0	文本	报警文本块以文本形式显示。
1	值	报警文本块以值形式显示。

参见

AlarmControl (页 255)

OperatorMessageSourceType9

描述

指定操作员输入报警中的报警文本块内容是以文本形式是传送还是以值形式传送。

运行系统中可进行读写访问

语法

Object.**OperatorMessageSourceType9**[=TransferAs]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

TransferAs

可选项。用于指定报警文本块在操作员输入报警中的显示方式的值或常量。

有下列设置可用：

值	名称	描述
0	文本	报警文本块以文本形式显示。
1	值	报警文本块以值形式显示。

参见

AlarmControl (页 255)

OperatorMessageSourceType10

描述

指定操作员输入报警中的报警文本块内容是以文本形式是传送还是以值形式传送。

运行系统中可进行读写访问

语法

Object.**OperatorMessageSourceType10**[=TransferAs]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

TransferAs

可选项。用于指定报警文本块在操作员输入报警中的显示方式的值或常量。

有下列设置可用：

值	名称	描述
0	文本	报警文本块以文本形式显示。
1	值	报警文本块以值形式显示。

参见

AlarmControl (页 255)

OutputAddressText

说明

在运行系统中无访问权限。

PaddingBottom

说明

在运行系统中无访问权限。

PaddingLeft

说明

在运行系统中无访问权限。

PaddingRight

说明

在运行系统中无访问权限。

PaddingTop

说明

在运行系统中无访问权限。

PageMode

描述

在长期归档列表中启用分页。允许在长期归档列表中显示短期归档的所有消息。使用“PageModeMessageNumber”属性确定每页显示的消息数。如果启用了分页，则可使用工具栏中的上一页/下一页按钮。

运行系统中的访问权限：读和写

语法

Object.**PageMode**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl

BOOLEAN

可选项。

TRUE，在“历史报警列表（长期）”(historical alarm list (long-term)) 中可以滚动。

FALSE，在“历史报警列表（长期）”(historical alarm list (long-term)) 中不可以滚动。

参见

AlarmControl (页 255)

PageModeMessageNumber

描述

定义对长期归档列表分页时每页显示的消息数。

运行系统中的访问权限：读和写

语法

Object.**PageModeMessageNumber**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmControl

Int32

可选项。用于指定每页消息数量的值或常量。

参见

AlarmControl (页 255)

Password

描述

确定装载远程监控的口令。

运行系统中的访问权限：读和写

语法

Object.**Password**[=STRING]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- SmartClientView

STRING

可选项。其中包含用于建立远程监视的密码的值或常量。

参见

SmartClientView (页 490)

PasswordsMustBeEncrypted

说明

在运行系统中无访问权限。

Path

说明

将不包含文件名的当前项目的路径返回为 STRING。 对于没有所属路径的 WinCC 客户端，路径以 UNC 格式返回，否则返回局部路径。

运行中可进行的访问：读取

语法

Object.**Path**

Object

必需。 “Project”对象。

示例

下列示例将项目路径返回为 Trace：

```
'VBS161
HMIRuntime.Trace "Path: " & HMIRuntime.ActiveProject.Path & vbNewLine
```

参见

Project (页 230)

PathHeaderBackColor

说明

在运行系统中无访问权限。

PathHeaderFont

说明

设置标题字体。

运行系统中可进行读写访问

语法

Object.**PathHeaderFont**[=Font]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- PLCCodeViewer
- S7GraphOverview

Font

可选项。用于指定标题字体的值或常量。

参见

PLCCodeViewer (页 442)

S7GraphOverview (页 476)

PathHeaderTextColor

说明

指定 GRAPH 概览标题中的字体颜色。

运行系统中可进行读写访问

语法

Object.**PathHeaderTextColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- S7GraphOverview

Color

可选项。用于指定 S7-GRAPH 概览标题字体颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

S7GraphOverview (页 476)

PercentageAxis

说明

指定在趋势视图中显示带百分比刻度的附加轴。

运行系统中的访问权限：读和写

语法

Object.**PercentageAxis**[=BOOLEAN]

Object

必需项。"ScreenItem"对象，且具有以下特性：

- OnlineTrendControl

BOOLEAN

可选项。

TRUE，显示带百分比刻度的轴。

FALSE，仅显示值和时间轴。

参见

OnlineTrendControl (页 418)

PercentageAxisAlignment

说明

指定百分比轴对齐方式。

运行系统中的访问权限：读和写

语法

Object.**PercentageAxisAlignment**[=AxisAlignment]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- OnlineTrendControl

AxisAlignment

可选项。 用于指定百分比轴对齐方式的值或常量。

值	名称	说明
0	左对齐	左对齐带百分比刻度的轴。
1	右对齐	右对齐带百分比刻度的轴。

参见

OnlineTrendControl (页 418)

PercentageAxisColor

说明

指定百分比轴的字体颜色和线条颜色。

运行系统中的访问权限：读和写

语法

Object.**PercentageAxisColor**[=Color]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- OnlineTrendControl

Color

可选项。 用于指定百分比轴的颜色的值或常量。

注释

使用“RGB”函数来定义 RGB 格式（红、绿、蓝）的颜色。 为此，可分别为三个 RGB 值指定相应的十进制值（值范围为 0 至 255）。 “红色”显示如下：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

OnlineTrendControl (页 418)

Picture

描述

指定要在运行系统的对象中显示的 WinCC 项目图形的画面。

运行系统中的访问权限：读和写

语法

Object.**Picture**[=HmiObjectHandle]

Object

必需项。具有以下格式的" ScreenItem"类型的对象：

- Clock
- GraphicView

HmiObjectHandle

可选项。用于指定要在运行系统的对象中显示的 WinCC 项目图形的画面的值或常量。

PictureAlignment

描述

指定过程映像中背景画面的显示类型。

运行系统中的访问权限：读和写

语法

Object.**PictureAlignment**[=PictureAlignment]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- Button
- RoundButton*

*：只读访问权限

PictureAlignment

可选项。用于指定过程映像中背景画面显示类型的值或常量。

PictureAreaBottomMargin

说明

在运行系统中无访问权限。

PictureAreaLeftMargin

说明

在运行系统中无访问权限。

PictureAreaRightMargin

说明

在运行系统中无访问权限。

PictureAreaTopMargin

说明

在运行系统中无访问权限。

PictureAutoSizing

说明

在运行系统中无访问权限。

PictureDeactivated

描述

指定在“禁用”状态下显示的 WinCC 项目图形文件。

运行系统中可进行读访问

语法

Object.**PictureDeactivated**[=HmiObjectHandle]

Object

必需项。具有以下格式的""ScreenItem"类型的对象：

- Roundbutton

HmiObjectHandle

可选项。用于指定在“取消激活”(Deactivated) 状态下显示的画面的值或常量。

参见

RoundButton (页 471)

PictureList

说明

在运行系统中无访问权限。

PictureOff

描述

指定在“关闭”(Off) 状态下显示的画面。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**PictureOff**[=HmiObjectHandle]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- Button
- GraphicIOField
- RoundButton*

*：只读访问权限

在运行系统中您没有以下格式的访问权限：

- Switch

HmiObjectHandle

可选项。用于指定在“关闭”(Off) 状态下显示画面的值或常量。

注释

画面（*.BMP 或 *.DIB）必须位于当前项目的“GraCS”文件夹中，以便进行集成。

PictureOn

描述

指定在“打开”状态下显示的画面。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**PictureOn**[=HmiObjectHandle]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- Button
- GraphicIOField
- RoundButton*

*：只读访问权限

在运行系统中您没有以下格式的访问权限：

- Switch

HmiObjectHandle

可选项。用于指定在“打开”(on) 状态下显示画面的值或常量。

注释

画面（*.BMP 或 *.DIB）必须位于当前项目的"GraCS"文件夹中才能进行集成。

PictureRotation

说明

在运行系统中无访问权限。

PictureSizeMode

说明

指定媒体播放器和要显示的内容之间的大小调整方式。

运行系统中的访问权限：读和写

语法

Object.**PictureSizeMode**[=PictureSizeMode]

Object

必需项。 "ScreenItem"对象，且具有以下特性：

- MediaPlayer

PictureSizeMode

可选项。 用于指定媒体播放器和要显示的内容之间的大小调整方式的值或常量。

值	名称	说明
0	按内容调整对象大小	指定按要显示的内容调整媒体播放器大小。
1	按对象大小调整内容	指定按媒体播放器的大小调整要显示的内容。

参见

MediaPlayer (页 396)

PlayCount

说明

在运行系统中无访问权限。

PlayEndless

说明

指定以无限循环方式重复播放媒体文件。

运行系统中的访问权限：读和写

语法

Object.**PlayEndless**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- MediaPlayer

BOOLEAN

可选项。

TRUE，以无限循环方式重复播放媒体文件。

FALSE，仅播放一次媒体文件。

参见

MediaPlayer (页 396)

PLCFilter

说明

在运行系统中无访问权限。

PlcUDTFilter

说明

在运行系统中无访问权限。

PointerColor

描述

指定"Gauge"对象的指针颜色。

运行系统中可进行读写访问

语法

Object.**PointerColor**[=Color]

Object

必需项。具有以下格式的""ScreenItem"类型的对象：

- Gauge

Color

可选项。用于指定指针颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Gauge (页 366)

Points

说明

在运行系统中无访问权限。

PointsCount

描述

指定折线或多边形角点的个数。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**PointsCount**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Polygon
- Polyline
- Tubepolyline

在运行系统中您没有以下格式的访问权限：

- Connector
- Line

Int32

可选项。用于指定折线拐角点个数的值或常量。

PopupMenuEnabled

说明

在运行系统中无访问权限。

PositionFont

说明

设置滑块标签的字体。

运行系统中的访问权限：读和写

语法

Object.**PositionFont**[=Font]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Slider

Font

可选项。用于指定字体的值或常量。

Precision

描述

确定小数位的个数（0 ～ 20）。

运行系统中的访问权限：读和写

语法

Object.**Precision**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

Int32

c 可选项。用于指定小数位个数（0 至 20）的值或常量。

参见

Bar (页 285)

PreferredUseOnAck

说明

在运行系统中无访问权限。

Pressed

描述

指定所选对象是否在“已按下”状态下显示。

运行系统中的访问权限：读和写

语法

Object.**Pressed**[=BOOLEAN]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- Button
- RoundButton*

*：只读访问权限

BOOLEAN

可选项。如果所选对象显示为“已按下”状态，则为 TRUE。

PrintJob

说明

指定在“报表”(Reports) 编辑器中创建的打印作业。

运行系统中的访问权限：读和写

语法

Object.**PrintJob**[=HmiObjectHandle]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

HmiObjectHandle

可选项。 用于指定打印输出布局的值或常量。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ProcessTag

描述

在运行系统中无访问权限。

ProcessValue

描述

指定要显示的数值的默认值。

若起动画面时未连接或更新相关变量，则在运行系统中使用该值。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**ProcessValue**[=DOUBLE | Int32 | Object | SINGLE]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- CheckBox
- Gauge
- GraphicIOField
- IOField
- OptionGroup
- Slider
- SymbolicIOField*
- WindowsSlider*

*：只读访问权限

在运行系统中您没有以下格式的访问权限：

- Button
- DateTimeField
- Switch
- SymbolLibrary

DOUBLE | Int32 | SINGLE

可选项。其中包含默认值的值或常量。数据类型取决于格式。

- DOUBLE: Bar
- Int32: CheckBox, GraphicIOField, OptionGroup, Slider, SymbolicIOField, WindowsSlider
- Object: IOField
- SINGLE: Gauge

注释

若要分配"ProcessValue"SmartTags 属性，则必须按如下方式确定分配：

'SmartTags 分配示例

```
'示例 1
IOField.ProcessValue = SmartTags("TagName").Value
'示例 2
HmiRuntime.Screens("Screen_1").ScreenItems("IOField_1").ProcessValue =
SmartTags("Tag_1").Value
```

ProgID

描述

对于非 WinCC 控件，与版本无关的 ProgID 作为类型返回。

ProhibitDataRecordTagInOnlySimpleView

说明

在运行系统中无访问权限。

1.5.5.11 属性 Q-R

QualityCode

说明

读取作为 SHORT 的变量后，返回变量值的特性。 在写入变量后，该值无效。

运行中可进行的访问：读取

语法

Object.**QualityCode**

Object

必需。它是"HMIRuntime"类型的对象。

示例

以下示例显示在读取过程期间无错误发生时，读取值的特性：

```
'VBS83
Dim objTag
Dim lngLastErr
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
lngLastErr = objTag.LastError
If 0 = lngLastErr Then
MsgBox objTag.QualityCode
End If
```

参见

变量 (页 244)

Radius

描述

指定半径。

运行系统中的访问权限：读和写

语法

Object.**Radius**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Circle
- CircleSegment
- CircularArc
- RoundButton

Int32

可选项。用于确定以像素为单位的半径的值或常量。

RadiusHeight

描述

指定副轴。

运行系统中的访问权限：读和写

语法

Object.**RadiusHeight**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Ellipse
- EllipseSegment
- EllipticalArc
- TubeArcObject

在运行系统中您没有以下格式的访问权限：

- Circle

Int32

可选项。用于确定以像素为单位的副轴的值或常量。

参见

Ellipse (页 340)

EllipseSegment (页 344)

EllipticalArc (页 348)

TubeArcObject (页 554)

Circle (页 312)

RadiusWidth

描述

指定主轴。

运行系统中的访问权限：读和写

语法

Object.**RadiusWidth**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Ellipse
- EllipseSegment
- EllipticalArc
- TubeArcObject

在运行系统中您没有以下格式的访问权限：

- Circle

Int32

可选项。用于确定以像素为单位的主轴的值或常量。

参见

Ellipse (页 340)

EllipseSegment (页 344)

EllipticalArc (页 348)

TubeArcObject (页 554)

Circle (页 312)

Recipe

说明

在运行系统中无访问权限。

RecipeName

描述

返回当前在“配方视图”(Recipe view) 中显示的配方的名称。

运行系统中的访问权限：读和写

语法

Object.**RecipeName**[=STRING]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- RecipeView

STRING

可选项。可返回配方名称的值或常量。

参见

RecipeView (页 458)

RecipeNameCaption

说明

在运行系统中无访问权限。

RecipeNrCaption

说明

在运行系统中无访问权限。

RecipeNrColFirst

说明

在运行系统中无访问权限。

RecipeNumber

描述

返回当前在“配方视图”(Recipe view) 中显示的配方的编号。

运行系统中的访问权限：读和写

语法

Object.**RecipeNumber**[=Int32]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- RecipeView

Int32

可选项。可返回配方编号的值或常量。

参见

RecipeView (页 458)

RecordName

描述

返回当前在“配方视图”(Recipe view) 中显示的配方数据记录的名称。

运行系统中的访问权限：读和写

语法

Object.**RecordName**[=STRING]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- RecipeView

STRING

可选项。可返回配方数据记录名称的值或常量。

参见

RecipeView (页 458)

RecordNrColFirst

说明

在运行系统中无访问权限。

RecordNumber

描述

返回当前在“配方视图”(Recipe view) 中显示的配方数据记录的编号。

运行系统中的访问权限：读

语法

Object.**RecordNumber**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- RecipeView

Int32

可选项。可返回配方数据记录编号的值或常量。

参见

RecipeView (页 458)

RelativeFillLevel

描述

指定对象的填充百分比。

运行系统中的访问权限：读和写

语法

Object.**RelativeFillLevel**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Button
- CheckBox
- Circle
- Ellipse
- EllipseSegment
- GraphicView
- OptionGroup
- Polygon
- Rectangle
- RoundButton*

- TextField
- WindowsSlider*

*：只读访问权限

在运行系统中您没有以下格式的访问权限：

- CircleSegment

Int32

可选项。用于指定对象填充百分比的值或常量。

RenameButtonVisible

说明

在运行系统中无访问权限。

ReSizeable

说明

在运行系统中无访问权限。

RightMargin

说明

在运行系统中无访问权限。

Rotation

描述

以度为单位指定旋转角度。逆时针方向测量旋转角度。

运行系统中的访问权限：读和写

语法

Object.**Rotation**[=SymbolLibraryRotation]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- SymbolLibrary

SymbolLibraryRotation

可选项。用于指定以度为单位的旋转角度的值或常量。

值	VB 常量	说明
0	hmiSymbolLibraryRotationNone	对象旋转 0 度。
90	hmiSymbolLibraryRotation90Degree	对象旋转 90 度。
180	hmiSymbolLibraryRotation180Degree	对象旋转 180 度。
270	hmiSymbolLibraryRotation270Degree	对象旋转 270 度。

RotationAngle

描述

以度为单位指定旋转角度。

运行系统中的访问权限：读和写

语法

Object.**RotationAngle**[=Int32]

Object

必选项。具有以下格式的"ScreenItem"类型的对象：

- Line
- Polygon
- Polyline

- TextField
- TubeTeeObject

Int32

可选项。用于指定以度为单位的旋转角度的值或常量。

注释

在运行系统中，对象按参考点顺时针旋转。

参见

Line (页 388)

Polygon (页 444)

Polyline (页 448)

TextField (页 527)

TubeTeeObject (页 563)

RotationCenterLeft

说明

指定运行系统中对象旋转中心点的 X 坐标。

运行系统中的访问权限：读和写

语法

Object.**RotationCenterLeft**[=Int32]

Object

必选。具有以下格式的“ScreenItem”类型的对象：

- Line
- Polygon
- Polyline
- TextField*

*：只读访问权限

Int32

可选项。用于指定支点的 X 坐标以在运行系统中旋转对象的值或常量。

注释

X 坐标的值相对于对象宽度。以百分比形式指定以围绕对象的矩形左边缘为起点的值。

RotationCenterTop

说明

指定运行系统中对象旋转中心点的 Y 坐标。

运行系统中的访问权限：读和写

语法

Object.**RotationCenterTop**[=Int32]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- Line
- Polygon
- Polyline
- TextField*

*：只读访问权限

Int32

可选项。用于指定支点的 Y 坐标以在运行系统中旋转对象的值或常量。

注释

Y 坐标的值相对于对象宽度。以百分比形式指定以围绕对象的矩形上边缘为起点的值。

RoundCornerHeight

描述

指定转角半径。输入对象一半高度百分比形式的值。

运行系统中的访问权限：读和写

语法

Object.**RoundCornerHeight**[=Int32]

Object

必选项。具有以下格式的"ScreenItem"类型的对象：

- Rectangle

Int32

可选项。用于指定角半径的值或常量。

RoundCornerWidth

描述

指定转角半径。输入对象一半宽度百分比形式的值。

运行系统中的访问权限：读和写

语法

Object.**RoundCornerWidth**[=Int32]

Object

必选项。具有以下格式的"ScreenItem"类型的对象：

- Rectangle

Int32

可选项。用于指定角半径的值或常量。

RowScrollbar

描述

指定将要显示垂直滚动条的时间。

运行系统中可进行读写访问

语法

Object.**RowScrollbar**[=ScrollbarVisibility]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

ScrollbarVisibility

可选项。用于指定垂直滚动条显示时间的值或常量。

值	说明
0	不显示垂直滚动条。
1	如果显示区太小不足以显示所有行，则会显示垂直滚动条。
2	始终显示垂直滚动条。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

RowTitleAlignment

说明

指定行标题对齐的类型。

运行系统中可进行读写访问

语法

Object.**RowTitleAlignment**[=HorizontalAlignment]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

HorizontalAlignment

可选项。用于指定行标题对齐方式的值或常量。

值	说明	说明
0	左对齐	将行标题调整为左对齐。
1	居中	将行标题调整为居中对齐。
2	右对齐	将行标题调整为右对齐。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

RowTitles

描述

指定是否显示已编号的列标题。

运行系统中可进行读写访问

语法

Object.**RowTitles**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。TRUE，显示已编号的列标题。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

RTPersistence

说明

指定在画面更改后是否保留在线组态。

也可将"RTPersistenceType"设置为 1 或 2。

运行系统中可进行的访问：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**RTPersistence**[=RTPersistence]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

在运行系统中您没有以下格式的访问权限：

- SysDiagControl
- UserView

RTPersistence

可选项。用于指定在画面更改后是否保留在线组态的值或常量。

值	解释
0	下次画面更改时放弃当前在线组态。
1	下次画面更改时保留当前在线组态。
2	所有已进行的在线组态均会丢失。将画面设为最初组态的内容。

RTPersistenceAuthorization

说明

指定运行系统中在线组态所需的权限。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**RTPersistenceAuthorization**[=HmiObjectHandle]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

在运行系统中您没有以下特性的访问权限：

- SysDiagControl
- UserView

HmiObjectHandle

可选项。 用于指定运行系统中用户权限的值或常量。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

SysDiagControl (页 514)

UserView (页 581)

RTPersistenceType

说明

指定如何保留 WinCC 的在线组态。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读取

语法

Object.**RTPersistenceType**

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

在运行系统中您没有以下格式的访问权限：

- SysDiagControl
- UserView

RTPersistenceType

可选项。用于指定 WinCC 如何保留在线组态的值或常量。

值	解释
0	已放弃在线组态。这些组态在下次画面更改时会丢失。
1	运行期间保留在线组态。退出时在线组态会丢失。
2	永久保留在线组态。重新启动后它们也可用。

RulerColor

描述

指定轴标签的刻度分度（帮助线）颜色。

运行系统中可进行读写访问

语法

Object.**RulerColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- TrendView

Color

可选项。用于指定刻度分度颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

TrendView (页 546)

RulerColumns

说明

在运行系统中无访问权限。

RulerType

说明

指定数值表的显示模式。

运行系统中的访问权限：读和写

语法

Object.**RulerType**[=TrendRulerControlType]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- TrendRulerControl

TrendRulerControlType

可选项。 用于指定数值表显示模式的值或常量。

值	名称	说明
0	标尺	指定显示标尺窗口。标尺窗口显示了标尺上趋势的坐标值或表格中选定行的值。
1	极值	指定显示统计区域窗口。 统计区域窗口显示两个标尺之间的趋势的上下限或表格中选定区域的上下限。
2	统计	指定显示统计窗口。 统计窗口显示了两个标尺之间的趋势统计评估或表格中选定值的统计评估。

参见

TrendRulerControl (页 532)

1.5.5.12 属性 S

S7Device

说明

在运行系统中无访问权限。

ScaleColor

描述

指定刻度的颜色。

运行系统中的访问权限：读和写

语法

Object.**ScaleColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- TrendView

Color

可选项。用于指定刻度颜色的值或常量。

注释

"Bar"格式的 ScreenItem 对象：关于显示的颜色，属性""ShowScale"值必须为 TRUE。

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

ScaleDenominator

描述

定义客户端上的标定计数器。

语法

Object.ScaleDenominator=[Int]

Object

必需项。“ScreenItem”对象，且具有“SmartClientView”格式。

Int

可选项。 用于指定值的数值或常量。

参见

SmartClientView (页 490)

ScaleDenominator

说明

在运行系统中无访问权限。

ScaleGradation

描述

指定两个主要刻度标记长度之间的距离。

运行系统中的访问权限：读和写

语法

Object.**ScaleGradation**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

DOUBLE

可选项。用于指定两个主要刻度标记长度之间距离的值或常量。

参见

Bar (页 285)

ScaleLabelColor

描述

指定“Gauge”对象刻度标记标签的颜色。

运行系统中可进行读写访问

语法

Object.**ScaleLabelColor**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Gauge

Color

可选项。用于指定刻度分度标签颜色的值或常量。

注释

可以使用“RGB”函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Gauge (页 366)

ScaleLabelFieldLength

说明

在运行系统中无访问权限。

ScaleLabelFont

说明

指定刻度标签的字体。

运行系统中的访问权限：读和写

语法

Object.**ScaleLabelFont**[=Font]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Gauge

Font

可选项。用于指定刻度标签字体的值或常量。

ScaleLabelingDoubleLined

描述

在运行系统中无访问权限。

ScaleNumerator

描述

在运行系统中无访问权限。

ScalePosition

说明

指定刻度的位置。

也可将“ShowScale”设置为“TRUE”。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**ScalePosition**[=ScalePosition]

Object

必选。具有以下格式的“ScreenItem”类型的对象：

- Bar

在运行系统中您没有以下格式的访问权限：

- Slider

ScalePosition

可选项。用于指定刻度位置的值或常量。

值	VB 常量	说明
0	hmiScalePositionLeftUp	对于垂直滚动条，刻度在顶部显示。对于水平滚动条，刻度在左侧显示。
1	hmiScalePositionRightDown	对于垂直滚动条，刻度在底部显示。对于水平滚动条，刻度在右侧显示。

ScaleStart

说明

在运行系统中无访问权限。

ScaleTickColor

描述

指定"Gauge"对象刻度分度的颜色。

运行系统中的访问权限：读和写

语法

Object.**ScaleTickColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Gauge

Color

可选项。用于指定刻度分度颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

ScaleTickLabelPosition

描述

指定刻度标签所在的理论圆的直径。

运行系统中的访问权限：读和写

语法

Object.**ScaleTickLabelPosition**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Gauge

DOUBLE

可选项。用于指定刻度标签所在的理论圆直径的值或常量。

值范围为 0 到 1。

0: 标签位于刻度的中心位置。

1: 针对标签的理论圆的直径是几何属性“Width”“或“Height”的较小值。 标签的一部分不在对象极值的范围内，因此不可见。

参见

Gauge (页 366)

ScaleTickLength

描述

指定刻度主要标记的长度。 该值指几何属性“”Width”或“Height"."较小值的一半。

微调刻度的标记长度的长度值是 0.5* 刻度宽度。

运行系统中的访问权限：读和写

语法

Object.**ScaleTickLength**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Gauge

DOUBLE

可选项。用于指定刻度主要标记长度的值或常量。

值范围介于 0 到最大刻度值

0: 没有刻度。 该区域的刻度不可见。

最大刻度值： 刻度范围介于刻度盘的中点到最大刻度值指定的值。

参见

Gauge (页 366)

ScaleTickPosition

描述

指定刻度所在的理论圆的直径。

刻度的主要标记取决于在该圆上的外端点位置。

运行系统中的访问权限：读和写

语法

Object.**ScaleTickPosition**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Gauge

DOUBLE

可选项。用于指定刻度所在的理论圆直径的值或常量。

值范围为 0 到 1。

0: 刻度位于刻度装置的中心位置。

1: 针对标签的理论圆的直径是刻度属于几何属性“"Width"”或“Height”较小值的情况。

参见

Gauge (页 366)

Scaling

描述

在运行系统中无访问权限。

ScalingType

描述

指定条缩放比例的类型。

也可将“ShowScale”设置为“TRUE”。

运行系统中可进行读写访问

语法

Object.**ScalingType**[=BarScalingType]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Bar

BarScalingType

可选项。用于指定棒图缩放比例的值或常量。

值	VB 常量	说明	说明
0	hmiBarScalingLinear	线性	主要标记在刻度尺上均匀分布。主要标记之间的距离对应于“LargeTicksSize”的值。
1	hmiBarScalingLogarithmic	对数	主要标记在刻度尺上呈对数函数分布。 特别突出较小值的表示。
2	hmiBarScalingNegativeLogarithmic	负对数	主要标记在刻度尺上呈负对数函数分布。特别突出较大值的表示。
3	hmiBarScalingAutomatic	自动	主要标记在刻度尺上均匀分布。自动指定主要标记之间的距离。
4	hmiBarScalingTangent	正切	刻度尺上主要标记的分布形式同时突出较小值和较大值的表示。

值	VB 常量	说明	说明
5	hmiBarScalingQuadratic	平方	主要标记呈二次函数分布。突出较大值的表示。
6	hmiBarScalingCubic	三次方	主要标记在刻度尺上呈三次函数分布。这样可以突出较大值的表示。

参见

Bar (页 285)

ScreenItems

说明

返回 ScreenItems 列表。

运行系统中可进行的访问：Read

语法

Object.ScreenItems

Object

要求"ScreenItems"."对象。

参见

ScreenItem (页 234)

ScreenItems（列表） (页 236)

ScreenName

描述

指定在运行系统画面窗口中显示的画面。

运行系统中的访问权限：读和写

语法

Object.**ScreenName**[=HmiObjectHandle]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- ScreenWindow

HmiObjectHandle

可选项。用于指定将在运行系统画面窗口中显示的画面的值或常量。

参见

ScreenWindow (页 479)

Screens

说明

返回 Screens 列表。Screens 列表包含两种元素：第一个元素，索引为“0”，表示永久区域。另一个元素包含索引 1，表示根画面。另外，两个元素都可以通过其名称寻址。使用永久区域的“Overview”和根画面的“Base”。

注意

报警窗口和报警指示器并不包含在 Screens 列表中，即使在运行系统中它们有焦点。

运行系统中的访问权限：Read

语法

Object.Screens

Object

必需项。“Screens”类型的对象。

参见

Screen 对象(列表) (页 238)

ScreenScaleMode

描述

指定所显示画面的缩放模式。

运行系统中的访问权限：读和写

语法

Object.ScreenScaleMode[=ScreenScaleModeType]

Object

必选项。具有以下格式的"ScreenItem"类型的对象：

- ScreenWindow

ScreenScaleModeType

可选项。用于指定所显示画面的缩放模式的值或常量。

ScrollBarOrientation

说明

在运行系统中无访问权限。

SecondGradientColor

说明

在运行系统中无访问权限。

SecondGradientOffset

说明

在运行系统中无访问权限。

SecondNeedleHeight

描述

指定秒针的长度。

运行系统中可进行读写访问

语法

Object.**SecondNeedleHeight**[=Int32]

Object

必需项。具有以下格式的""ScreenItem"类型的对象：

- Clock

Int32

可选项。用于指定秒针长度的值或常量。

指定秒针的长度，作为与钟盘半径相关的百分值。

参见

Clock (页 323)

SecondNeedleWidth

描述

指定秒针的宽度。

运行系统中可进行读写访问

语法

Object.**SecondNeedleWidth**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Clock

Int32

可选项。用于指定秒针宽度的值或常量。以两倍秒针长度的百分比形式指定秒针宽度。

参见

Clock (页 323)

SecurityForSimpleViewEnabled

说明

在运行系统中无访问权限。

SegmentColoring

描述

指定超出限制值时要显示的颜色更改类型。

运行系统中可进行读写访问

语法

Object.**SegmentColoring**[=THmiBarColorType]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar

THmiBarColorType

可选项。用于指定颜色变化类型的值或常量。值范围为 0 到 1。

值	VB 常量	说明
0	hmiBarColorEntire	颜色更改应用到整个棒图。
1	hmiBarColorSegmented	颜色更改应用到段中。

参见

Bar (页 285)

SelectArchiveName

说明

指定最初在运行系统中显示配方视图数据源的选择对话框。

运行系统中的访问权限：读和写

语法

Object.**SelectArchiveName**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- UserArchiveControl

BOOLEAN

可选项。

TRUE，在画面中显示配方视图数据源的选择对话框。

FALSE，在画面中不显示配方视图数据源的选择对话框。

参见

用户归档控件 (页 566)

SelectBackColor

描述

指定所选文本条目的背景色。

运行系统中可进行读访问

语法

Object.**SelectBackColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- SymbolicIOField

Color

可选项。用于指定所选文本条目背景色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

SymbolicIOField (页 504)

SelectedCellColor

描述

指定所选单元格的背景色。

运行系统中可进行读写访问

语法

Object.**SelectedCellColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

可选项。用于指定背景色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

SelectedCellForeColor

描述

指定所选单元格的字体颜色。

运行系统中可进行读写访问

语法

Object.**SelectedCellForeColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

可选项用于指定字体颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

SelectedID

说明

返回在配方视图中选择的数据记录的 ID。

运行系统中可进行读写访问

语法

Object.**SelectedID**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- UserArchiveControl

Int32

可选项。用于返回所选数据记录 ID 的值或常量。

值	说明
0	选择的数据记录无效，例如，在连接错误事件中。
-1	选择编辑行。

参见

用户归档控件 (页 566)

SelectedIndex

描述

定义其关联文本在组合框或列表框中高亮显示的索引。

最大值是对象的行数 (NumberLines)。

运行系统中的访问权限：读和写

语法

Object.**SelectedIndex**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- ComboBox
- ListBox

Int32

可选项。用于指定高亮显示的文本索引的值或常量。

参见

ComboBox (页 326)

Listbox (页 392)

SelectedRowColor

描述

指定所选行的背景色。

运行系统中可进行读写访问

语法

Object.**SelectedRowColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

可选项。用于指定背景色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

SelectedRowForeColor

描述

指定所选行的字体颜色。

运行系统中可进行读写访问

语法

Object.**SelectedRowForeColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

可选项。用于指定字体颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

SelectedText

说明

指定使用“SelectedIndex”引用的条目的文本。

运行系统中可进行读写访问

语法

Object.**SelectedText**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- ComboBox
- ListBox

STRING

可选项。用于指定条目文本的值或常量。

参见

ComboBox (页 326)

Listbox (页 392)

SelectedTitleColor

描述

指定所选表格标题的背景色。

也可将“UseSelectedTitleColor”设置为“TRUE”。

运行系统中可进行读写访问

语法

Object.**SelectedTitleColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

可选项。用于指定背景色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

SelectedTitleForeColor

描述

指定所选表格标题的字体颜色。

也可将"UseSelectedTitleColor"设置为"TRUE"。

运行系统中可进行读写访问

语法

Object.**SelectedTitleForeColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

可选项。用于指定字体颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

用户归档控件 (页 566)

TrendRulerControl (页 532)

OnlineTableControl (页 402)

AlarmControl (页 255)

SelectForeColor

描述

指定所选文本条目的颜色。

运行系统中可进行读访问

语法

Object.**SelectForeColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- SymbolicIOField

Color

可选项。用于指定所选文本条目颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

SymbolicIOField (页 504)

SelectionBackColor

描述

指定所选单元格的背景色。

运行系统中可进行读写访问

语法

Object.**SelectionBackColor**[=Color]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmView
- RecipeView*
- StatusForce
- UserView

*：只读访问权限

在运行系统中您没有以下格式的访问权限：

- TrendView

Color

可选项。用于指定所选行背景色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

SelectionColoring

描述

指定是否对单元格或行使用选择颜色。

运行系统中可进行读写访问

语法

Object.**SelectionColoring**[=GridSelectionColoring]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

GridSelectionColoring

可选项。指定是否突出显示单元格或行的值或常量。

值	描述	说明
0	无	没有单元格或行的选择颜色。
1	单元格	单元格的选择颜色。
2	行	行的选择颜色。
3	单元格和行	单元格和行的选择颜色。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

SelectionForeColor

描述

指定所选单元格的前景色。

运行系统中可进行读写访问

语法

Object.**SelectionForeColor**[=Color]

对象

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmView
- RecipeView*
- StatusForce
- UserView

*：只读访问权限

在运行系统中您没有以下格式的访问权限：

- TrendView

Color

可选项。用于指定所选行背景色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

SelectionRect

描述

指定是否对所选单元格或行使用选择框架。

运行系统中可进行读写访问

语法

Object.**SelectionRect**[=GridSelectionBorder]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

GridSelectionBorder

可选项。用于指定是否对所选单元格或行使用选择边框的值或常量。

值	描述	说明
0	无	不为所选的单元格或行绘制选择边框。
1	单元格	为所选的单元格绘制选择边框。
2	行	为所选的行绘制选择边框。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

SelectionRectColor

描述

指定报警窗口中选择矩形的颜色。

也可将“SelectionType”设置为“1”或“2”。

运行系统中可进行读写访问

语法

Object.**SelectionRectColor**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

可选项。用于指定颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

SelectionRectWidth

描述

如果 SelectionType 等于"1"，则指定报警窗口中选择矩形的线宽。

运行系统中的访问权限：读和写

语法

Object.**SelectionRectWidth**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于指定线宽的值或常量。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

SelectionType

描述

指定可标记的行数。

运行系统中的访问权限：读和写

语法

Object.**SelectionType**[=GridSelectionType]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

GridSelectionType

可选项。用于指定可选行数的值或常量。

有下列设置可用：

值	描述	描述
0	无	未选择任何行。
1	单个选择	可选择一行。
2	多重选择	可选择多行。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

SeparateLineForAlarmText

说明

在运行系统中无访问权限。

SeparatorBackColor

描述

指定选择列表中虚线的背景色。

运行系统中可进行读访问

语法

Object.**SeparatorBackColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- SymbolicIOField

Color

可选项。用于指定选择列表中的虚线背景色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

SymbolicIOField (页 504)

SeparatorColor

描述

指定选择列表中分隔线的颜色。

运行系统中的访问权限：读和写

语法

Object.**SeparatorColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- S7GraphOverview
- SymbolicIOField *

* 只读访问权限

Color

可选项。用于指定选择列表中的分隔线颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

SeparatorCornerStyle

描述

返回分隔线的角形状。

运行系统中的访问权限：读取

语法

Object.**SeparatorCornerStyle**[=CornerStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- SymbolicIOField

CornerStyle

可选项。用于返回分隔线的角形状的值或常量。

值	说明
0	实线
1	短划线
2	虚线
3	点划线
4	双点划线

SeparatorLineEndShapeStyle

描述

返回分隔线的线端形状。

运行系统中的访问权限：读取

语法

Object.**SeparatorLineEndShapeStyle**[=LineEndShapeStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- SymbolicIOField

LineEndShapeStyle

可选项。用于返回分隔线的线端形状的值或常量。

值	名称
-1	齐平
0	圆形
1	矩形

SeparatorStyle

描述

指定选择列表中分隔线的线类型。

运行系统中的访问权限：读取

语法

Object.**SeparatorStyle**[=LineStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- SymbolicIOField

LineStyle

可选项。用于指定选择列表中分隔线类型的值或常量。

值	VB 常量	说明
-1	hmiLineStyleNone	选择列表无分隔线。
0	hmiLineStyleSolid	选择列表的分隔线为实心样式。
1	hmiLineStyleDash	选择列表的分隔线为虚线。
2	hmiLineStyleDot	选择列表的分隔线为点线。
3	hmiLineStyleDashDot	选择列表分隔线为点划线。
4	hmiLineStyleDashDotDot	选择列表分隔线为双点划线。

SeparatorWidth

描述

指定选择列表中分隔线的宽度。

运行系统中可进行读访问

语法

Object.**SeparatorWidth**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- SymbolicIOField

Int32

可选项。用于指定选择列表中分隔线宽度的值或常量。

参见

SymbolicIOField (页 504)

ServerNames

描述

指定对象接收数据时所采用的分布式系统的服务器。信息以下方式表示：NameServer1;NameServer2;NameServer3。

运行系统中可进行读写访问

语法

Object.**ServerNames**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

STRING

可选项。用于指定对象接收数据时所采用的分布式系统的服务器的值或常量。

参见

AlarmControl (页 255)

ServerPrefix

描述

指定在运行系统画面窗口中显示的画面所在的服务器，或返回服务器名称。

输入服务器名称，并在名称后面输入两个冒号："<服务器名称>::"。不会检查服务器是否真实存在。

运行系统中的访问权限：读和写

语法

Object.**ServerPrefix**[=String]

Object

必选项。具有以下格式的"ScreenItem"类型的对象：

- ScreenWindow

String

可选项。用于指定服务器名称的值或常量。

ServerScale

描述

在运行系统中无访问权限。

SetOfVisibleColumns

说明

在运行系统中无访问权限。

Shared

描述

在运行系统中无访问权限。

ShareSpaceWithSourceControl

说明

指定按数据表大小调整数据源的显示区域。

运行系统中可进行读写访问

语法

Object.**ShareSpaceWithSourceControl**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- TrendRulerControl

BOOLEAN

可选项。

TRUE，按照数值表大小调整数据源大小。

FALSE，不按数据表大小调整数据源大小。

参见

TrendRulerControl (页 532)

ShiftDecimalPoint

说明

在运行系统中无访问权限。

ShowAcknowledgeButton

说明

在运行系统中无访问权限。

ShowAlarmsFromDate

描述

指定仅显示该变量中保存的那些消息事件。

运行系统中的访问权限：读和写

语法

Object.**ShowAlarmsFromDate**[=HmiObjectHandle]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmView

HmiObjectHandle

可选项。用于指定仅显示该变量中保存的消息事件的值或常量。

参见

AlarmView (页 274)

ShowAlarmsToAcknowledge

说明

在运行系统中无访问权限。

ShowBadTagState

描述

定义检测到不良质量代码或变量状态时对象是否变成灰色。

运行系统中的访问权限：读和写

语法

Object.**ShowBadTagState**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Bar
- CheckBox
- IOField
- OptionGroup
- SymbolicIOField *
- WindowSlider *

* 只读访问权限

BOOLEAN

可选项。

在质量代码或变量状态不良时，如果对象变成灰色或使用网格颜色设置，则为 TRUE。

ShowBar

描述

指定显示的过程值是否也可通过填充的棒图显示。

运行系统中可进行读写访问

语法

Object.**ShowBar**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Slider

BOOLEAN

可选项。TRUE，过程值也通过填充的棒图显示。

参见

Slider (页 485)

ShowCaption

描述

指定是否显示工具栏。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**ShowCaption**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- ApplicationWindow
- ScreenWindow

在运行系统中您没有以下格式的访问权限：

- Switch

BOOLEAN

可选。如果显示了标题则为 TRUE。

ShowColumnHeaders

说明

在运行系统中无访问权限。

ShowControls

说明

指定显示工具栏。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**ShowControls**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- MediaPlayer

在运行系统中您没有以下格式的访问权限：

- SmartClientView

BOOLEAN

可选项。

TRUE，显示工具栏。

如果未显示工具栏，则为 FALS E。

ShowDate

说明

在运行系统中无访问权限。

ShowDecimalPoint

描述

指定刻度是否标识为小数值（小数点后一位）或整数值。

运行系统中的访问权限：读和写

语法

Object.**ShowDecimalPoint**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Gauge

BOOLEAN

可选。若刻度标识为小数值（小数点后一位），则为 TRUE。

参见

Gauge (页 366)

ShowDropDownButton

说明

返回是否显示用于选择列表的按钮。

运行系统中可进行读访问

语法

Object.**ShowDropDownButton**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- SymbolicIOField

BOOLEAN

可选项。

TRUE，显示用于选择列表的按钮。

FALSE，不显示用于选择列表的按钮。

参见

SymbolicIOField (页 504)

ShowDropDownList

说明

在运行系统中无访问权限。

ShowFeatureBackward

说明

指定在运行系统中要显示 "后退"(Backward) 按钮。

运行系统中可进行读写访问

语法

Object.**ShowFeatureBackward**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- MediaPlayer

BOOLEAN

可选项。

TRUE，显示"后退"(Backward) 按钮。

FALSE，不显示"后退"(Backward) 按钮。

参见

MediaPlayer (页 396)

ShowFeatureForward

说明

指定在运行系统中显示"快进"(Forward) 按钮。

运行系统中可进行读写访问

语法

Object.**ShowFeatureForward**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- MediaPlayer

BOOLEAN

可选项。

TRUE，显示"快进"(Forward) 按钮。

FALSE，不显示"快进"(Forward) 按钮。

参见

MediaPlayer (页 396)

ShowFeatureFullScreen

说明

指定可以全屏显示媒体播放器。

运行系统中可进行读写访问

语法

Object.**ShowFeatureFullScreen**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- MediaPlayer

BOOLEAN

可选项。

TRUE，在媒体播放器的工具栏中显示“全屏"(Full screen) 按钮。

FALSE，不在媒体播放器的工具栏中显示“全屏"(Full screen) 按钮。

参见

MediaPlayer (页 396)

ShowFeatureFullVolume

说明

在运行系统中无访问权限。

ShowFeaturePause

说明

指定在运行系统中显示“暂停”(Pause) 按钮。

运行系统中的访问权限：读和写

语法

Object.**ShowFeaturePause**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- MediaPlayer

BOOLEAN

可选项。

TRUE，在运行系统中显示“暂停”(Pause) 按钮。

FALSE，不在运行系统中显示“暂停”(Pause) 按钮。

参见

MediaPlayer (页 396)

ShowFeaturePlay

说明

指定在运行系统中显示“播放”(Play) 按钮。

运行系统中的访问权限：读和写

语法

Object.**ShowFeaturePlay**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- MediaPlayer

BOOLEAN

可选项。

TRUE，在运行系统中显示“播放”(Play) 按钮。

FALSE，不在运行系统中显示“播放”(Play) 按钮。

参见

MediaPlayer (页 396)

ShowFeatureStop

说明

指定在运行系统中显示“停止”(Stop) 按钮。

运行系统中的访问权限：读和写

语法

Object.**ShowFeatureStop**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- MediaPlayer

BOOLEAN

可选项。

TRUE，在运行系统中显示“停止”(Stop) 按钮。

FALSE，不在运行系统中显示“停止”(Stop) 按钮。

参见

MediaPlayer (页 396)

ShowFillLevel

描述

指定所选对象是否填充。

运行系统中的访问权限：读和写

语法

Object.**ShowFillLevel**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Button
- CheckBox
- Circle
- CircleSegment
- Ellipse
- EllipseSegment
- GraphicView
- OptionGroup
- Polygon
- Rectangle
- RoundButton*
- TextField
- WindowsSlider*

*：只读访问权限

BOOLEAN

可选。如果已对所选对象进行了填充，则为 TRUE。

ShowFocusRectangle

描述

指定按钮在运行系统中激活时，是否指定所选的边框。

运行系统中的访问权限：读和写

语法

Object.**ShowFocusRectangle**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Clock

BOOLEAN

可选。若按钮在运行系统中激活时指定所选的边框，则为 TRUE。

参见

Clock (页 323)

ShowHelpButton

说明

在运行系统中无访问权限。

ShowHorizontalGridlines

说明

在运行系统中无访问权限。

ShowInnerDial

说明

在运行系统中无访问权限。

ShowLargeTicksOnly

描述

指定是否仅显示大刻度线。

运行系统中的访问权限：读和写

语法

Object.**ShowLargeTicksOnly**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

BOOLEAN

可选。如果仅显示大刻度线，则为 TRUE。

参见

Bar (页 285)

ShowLeadingZeros

说明

在运行系统中无访问权限。

ShowLimitLines

说明

在运行系统中无访问权限。

ShowLimitRanges

说明

在运行系统中无访问权限。

ShowLimitMarkers

描述

指定极值是否显示为刻度值。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**ShowLimitMarkers**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar

在运行系统中您没有以下格式的访问权限：

- Slider

BOOLEAN

可选。如果限值显示为刻度值，则为 TRUE。

ShowLoopInAlarmButton

说明

在运行系统中无访问权限。

ShowMilliseconds

说明

在运行系统中无访问权限。

ShowNavigationButtons

说明

在运行系统中无访问权限。

ShowPathInformation

说明

在运行系统中无访问权限。

ShowPeakValuePointer

描述

指定从属指针是否将用于所选的对象。

在加载进程画面时，从属指针显示运行系统具有的最大指针偏移量。 若重新加载进程画面，则从属指针将复位。

运行系统中的访问权限：读和写

语法

Object.**ShowPeakValuePointer**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Gauge

BOOLEAN

可选。若使用从属指针，则为 TRUE。

参见

Gauge (页 366)

ShowPendingAlarms

说明

在运行系统中无访问权限。

ShowPosition

描述

指定是否当前滑块位置值还可以数字显示。 该值显示在滑块下方。

运行系统中的访问权限：读和写

语法

Object.**ShowPosition**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Slider

BOOLEAN

可选。若该值还以数字显示，则为 TRUE。

参见

Slider (页 485)

ShowProcessValue

说明

在运行系统中无访问权限。

ShowReadButton

说明

在运行系统中无访问权限。

ShowRuler

描述

指定是否为对象的轴标签显示刻度分度（帮助线）。

运行系统中的访问权限：读和写

语法

Object.**ShowRuler**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl
- TrendView

BOOLEAN

可选。如果显示刻度分度，则为 TRUE。

ShowRulerInAxis

描述

指定是否在时间轴中显示标尺。

运行系统中可进行读写访问

语法

Object.**ShowRulerInAxis**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

可选项。

TRUE，时间轴上也显示标尺。

FALSE，时间轴上不显示标尺。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

ShowScale

描述

指定数值是否还显示在刻度范围内。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**ShowScale**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar

在运行系统中您没有以下格式的访问权限：

- Slider

BOOLEAN

可选。如果数值还在刻度范围内显示，则为 TRUE。

ShowScrollBar

说明

在运行系统中无访问权限。

ShowScrollbars

描述

指定是否显示滚动条。

运行系统中可进行的访问：读和写

语法

Object.**ShowScrollbars**[=BOOLEAN]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl
- ScreenWindow

BOOLEAN

可选。如果显示滚动条则为 TRUE。

ShowSignForPositiveLabel

说明

在运行系统中无访问权限。

ShowSortButton

描述

指定是否在垂直滚动条上显示排序按钮。单击此排序按钮可基于已组态的排序标准对所选列排序。如果表格中不包含垂直滚动条，则不会显示排序按钮。

运行系统中可进行读写访问

语法

Object.**ShowSortButton**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，显示排序按钮。可对所选列排序。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ShowSortIcon

描述

指定是否显示排序图标。

运行系统中可进行读写访问

语法

Object.**ShowSortIcon**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，显示排序图标。

FALSE，不显示排序图标。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ShowSortIndex

描述

指定是否显示排序索引。

运行系统中可进行读写访问

语法

Object.**ShowSortIndex**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，显示排序索引。

FALSE，不显示排序索引。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ShowSplittedView

说明

在运行系统中无访问权限。

ShowStatisticRuler

说明

指定是否显示定义统计区域的线条。

运行系统中的访问权限：读和写

语法

Object.**ShowStatisticRuler**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- OnlineTrendControl

BOOLEAN

可选项。

TRUE，显示统计区域的线条。

FALSE，不显示统计区域的线条。

参见

OnlineTrendControl (页 418)

ShowStatusBar

描述

指定是否显示状态栏。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**ShowStatusBar**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- HTMLBrowser

在运行系统中您没有以下格式的访问权限：

- MediaPlayer

BOOLEAN

可选。如果显示状态栏，则为 TRUE。

ShowTableGridlines

描述

指定是否在表格中显示网格线。

运行系统中的访问权限：

- RT Advanced：读和写
- RT Professional：无访问权

语法

Object.**ShowTableGridlines**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- StatusForce

在运行系统中您没有以下格式的访问权限：

- TrendView
- UserView

BOOLEAN

可选。如果网格线显示在表格中则为 TRUE。

ShowThumb

描述

指定是否显示“Slider”对象的滑块。

运行系统中的访问权限：

- RT Advanced：读和写
- RT Professional：读取

语法

Object.**ShowThumb**[=BOOLEAN]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- Slider

BOOLEAN

可选项。如果显示滑块，则为 TRUE。

参见

Slider (页 485)

ShowTickLabels

说明

指定是否以刻度显示标签。

运行系统中的访问权限

- RT Advanced：读和写
- RT Profession al：读取

语法

Object.**ShowTickLabels**[=BOOLEAN]

Object

必选。具有以下格式的“ScreenItem”类型的对象：

- Slider

在运行系统中您没有以下格式的访问权限：

- Bar

BOOLEAN

可选项。TRUE，显示标签。

注释

根据指定的测量范围和对象大小，自动确定测量值的增量。

ShowTicks

描述

指定是否显示刻度中的标记。

运行系统中可进行读写访问

语法

Object.**ShowTicks**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Clock
- Slider*

* RT Professional，只读访问权限

BOOLEAN

可选项。TRUE，显示标记。

参见

Clock (页 323)

Slider (页 485)

ShowTime

说明

在运行系统中无访问权限。

ShowTimeAxis

说明

在运行系统中无访问权限。

ShowTimeAxisLabeling

说明

在运行系统中无访问权限。

ShowTitle

描述

指定对象的窗口边框和窗口标题的样式。

运行系统中可进行读写访问

语法

Object.**ShowTitle**[=WindowHeaderStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl

- TrendRulerControl
- UserArchiveControl

WindowHeaderStyle

可选项。用于指定对象窗口边框样式的值或常量。

值	描述
0	不显示窗口边框和标题。
1	显示带有标题的窄窗口边框。
2	显示带有标题的标准窗口边框。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ShowToolBar

描述

在运行系统中无访问权限。

ShowToolbarBackgroundColor

说明

在运行系统中无访问权限。

ShowTracker

说明

指定显示播放列表。

运行系统中的访问权限：读和写

语法

Object.**ShowTracker**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- MediaPlayer

BOOLEAN

可选项。

TRUE，显示播放列表。

FALSE，不显示播放列表。

参见

MediaPlayer (页 396)

ShowTrendIcon

描述

指定是否在数值轴下显示图标。图标指示当前显示在前景中的趋势。

运行系统中可进行读写访问

语法

Object.**ShowTrendIcon**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

可选项。

TRUE，显示该图标。

FALSE，不显示该图标。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

ShowTrendIndicator

描述

指定要监视的测量值的趋势（增大或减小）是否显示有小箭头。

运行系统中的访问权限：读和写

语法

Object.**ShowTrendIndicator**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

BOOLEAN

可选。若要监视的测量值的趋势（增大或减小）显示有小箭头，则为 TRUE。

参见

Bar (页 285)

ShowValueAxis1

说明

在运行系统中无访问权限。

ShowValueAxis1Label

说明

在运行系统中无访问权限。

ShowValueAxis2

说明

在运行系统中无访问权限。

ShowValueAxis2Label

说明

在运行系统中无访问权限。

ShowValueTable

说明

在运行系统中无访问权限。

ShowWriteButton

说明

在运行系统中无访问权限。

ShowY1HlpLine

说明

在运行系统中无访问权限。

ShowY2HlpLine

说明

在运行系统中无访问权限。

SimpleView

说明

在运行系统中无访问权限。

SimpleViewToolbar

说明

在运行系统中无访问权限。

Size

说明

在运行系统中无访问权限。

Sizeable

说明

指定可以在运行系统中更改对象的大小。

运行系统中的访问权限：读和写

语法

Object.**Sizeable**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Sizable

可选项。 用于指定是否可更改对象大小的值或常量。

参见

TrendRulerControl (页 532)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

AlarmControl (页 255)

FunctionTrendControl (页 351)

用户归档控件 (页 566)

SmartTags

说明

返回 SmartTags 列表。

运行系统中可进行的访问：Read

语法

Object.SmartTags

Object

要求""HMIRuntime"对象。

参见

HMIRuntime（页 224）

SortByTimeDirection

说明

指定是否在顶部显示最后接收到的消息（按升序顺序排序）。

运行系统中可进行读写访问

语法

Object.**SortByTimeDirection**[=SortByTimeDirection]

Object

必需项。具有以下格式的"ScreenItems"类型的对象：

- AlarmView

SortByTimeDirection

可选项。 用于指定对象排列顺序的值或常量。

值	名称	说明
0	降序	在顶部显示最后接收到的报警。
1	升序	在底部显示最后接收到的报警。

参见

AlarmView (页 274)

SortByTimeEnabled

描述

指定是否按照时间改变报警的排序。

运行系统中可进行读写访问

语法

Object.**SortByTimeEnabled**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItems"类型的对象：

- AlarmView

BOOLEAN

可选项。TRUE，设备操作员可以改变排序方式。

参见

AlarmView (页 274)

SortSequence

描述

如果操作员在运行系统中单击列标题，指定排序顺序会如何改变。

运行系统中的访问权限：读和写

语法

Object.**SortSequence**[=GridSortSequence]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

GridSortSequence

可选项。用于指定排列顺序如何通过鼠标单击进行更改的值或常量。

值	描述	说明
0	升/降/无	可通过鼠标单击在升序、降序和无排序之间进行切换。
1	升/降	可通过鼠标单击在升序和降序排序顺序之间进行切换。

SourceControl

说明

指定连接到值表的趋势或表格视图。

运行系统中的访问权限：读和写

语法

Object.**SourceControl**[=HmiObjectHandle]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- TrendRulerControl

HmiObjectHandle

可选项。 用于指定连接到值表的趋势或表格视图的值或常量。

参见

TrendRulerControl (页 532)

SourceControlType

说明

指定数据源的类型。

运行系统中的访问权限：读和写

语法

Object.**SourceControlType**[=TrendRulerControlSourceControlType]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- TrendRulerControl

TrendRulerControlSourceControlType

可选项。 用于指定值表数据源的类型的值或常量。

值	名称	说明
0	无	不为值表的数据源指定任何类型。
1	f(t) 趋势视图	指定数据源为 f(t) 趋势视图
2	表格视图	指定数据源为表格视图。
3	f(x) 趋势视图	指定数据源为 f(x) 趋势视图

参见

TrendRulerControl (页 532)

SplittedViewRatio

说明

在运行系统中无访问权限。

StartAngle

描述

指定起始点偏离零位置 (0°) 的角度。

运行系统中的访问权限：读和写

语法

Object.**StartAngle**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- CircleSegment
- CircularArc
- EllipseSegment
- EllipticalArc
- TubeArcObject

Int32

可选项。用于指定起始点偏离零位置 (0°) 的角度的值或常量。

参见

CircleSegment (页 316)

CircularArc (页 320)

EllipseSegment (页 344)

EllipticalArc (页 348)

TubeArcObject (页 554)

StartLeft

说明

在运行系统中无访问权限。

StartStyle

描述

指定如何显示直线起始端。

运行系统中的访问权限：读和写

语法

Object.**StartStyle**[=LineEndStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Connector
- Line
- Polyline

LineEndStyle

可选项。用于确定直线起始端的值或常量。值范围为 0 到 6。

值	VB 常量	说明
0	hmiLineEndStyleNone	线起始端无符号。
1	hmiLineEndStyleArrow	线起始端为箭头。
2	hmiLineEndStyleFilledArrow	线起始端为填充箭头。
3	hmiLineEndStyleFilledArrowReversed	线起始端为反向箭头。
4	hmiLineEndStyleLine	线起始端为垂直线。
5	hmiLineEndStyleCircle	线起始端为圆。
6	hmiLineEndStyleFilledCircle	线起始端为实心圆。

StartTop

描述

在运行系统中无访问权限。

StartValue

描述

定义刻度指示器上零点的绝对值。

运行系统中的访问权限：读和写

语法

Object.**StartValue**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

DOUBLE

可选项。用于指定刻度指示器零点的绝对值的值或常量。

参见

Bar (页 285)

State

说明

返回消息的状态。

下表列出了可能的消息状态：

状态	报警记录状态
1	已到达
2	已离去
5	已到达和注释
6	已离去和注释

StatisticAreaColumns

说明

在运行系统中无访问权限。

StatisticResultColumns

说明

在运行系统中无访问权限。

StatusbarBackColor

描述

指定状态栏的背景色。

也可将"StatusbarUseBackColor"设置为"TRUE"。

运行系统中可进行读写访问

语法

Object.**StatusbarBackColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Color

可选项。用于指定状态栏背景色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarElementAdd

描述

创建一个新的用户自定义状态栏元素。新创建的元素会自动使用“StatusbarElementIndex”而引用。

运行系统中可进行读写访问

语法

Object.**StatusbarElementAdd**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

必需项。用于通过“StatusbarElementName”指定状态栏新用户自定义的元素名称的值或常数。

参见

StatusbarElementIndex (页 1144)

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarElementAutoSize

描述

指定是否自动设置使用"StatusbarElementIndex"引用的状态栏元素的宽度。

运行系统中可进行读写访问

语法

Object.**StatusbarElementAutoSize**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，将自动设置所选元素的宽度。

FALSE，不会自动设置所选元素的宽度。

参见

用户归档控件 (页 566)

TrendRulerControl (页 532)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

AlarmControl (页 255)

StatusbarElementCount

描述

指定已组态的状态栏元素的数量。

运行系统中可进行读写访问

语法

Object.**StatusbarElementCount**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于指定可组态的状态栏元素数量的值或常量。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarElementIconId

描述

使用其图标 ID 来引用状态栏元素。

元素使用其图标 ID 进行引用不会依赖于元素的实际顺序。

运行系统中可进行读写访问

语法

Object.**StatusbarElementIconId**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于指定使用图标 ID 引用的要编辑的状态栏元素的值或常量。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarElementID

描述

使用其元素 ID 引用状态栏元素。要访问状态栏元素的属性，需要设置 ""StatusbarElementID""。

元素使用其元素 ID 进行引用不会依赖于元素的实际顺序。元素 ID 列于对象的巡视窗口中“属性 > 属性 > 状态栏 > 状态栏 - 元素”(Properties > Properties > Status bar > Status bar - Elements) 下。

运行系统中可进行读写访问

语法

Object.**StatusbarElementID**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于指定待移除的用户自定义状态栏元素名称的值或常量。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarElementIndex

描述

引用状态栏元素。要访问状态栏元素的属性，需要设置""StatusbarElementIndex""。

介于 0 至 (StatusbarElementCount - 1) 之间的值为"StatusbarElementIndex"的有效值。"StatusbarElementCount"属性指定可组态的状态栏元素的数量。

运行系统中可进行读写访问

语法

Object.**StatusbarElementIndex**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于通过索引指定要编辑的状态栏元素的值或常量。

参见

StatusbarElementCount (页 1141)

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarElementName

描述

指定使用"StatusbarElementIndex"引用的状态栏元素的名称。

运行系统中可进行读写访问

语法

Object.**StatusbarElementName**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

可选项。用于指定待移除的用户自定义状态栏元素名称的值或常量。

参见

StatusbarElementIndex (页 1144)

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarElementRemove

描述

使用引用的用户自定义状态栏元素的名称可移除该状态栏元素。

运行系统中可进行读写访问

语法

Object.**StatusbarElementRemove**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

可选项。用于指定要移除的引用的用户定义的状态栏元素名称的值或常数。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarElementRename

说明

指定使用"StatusbarElementIndex"引用的用户自定义的状态栏元素的新名称。

运行系统中可进行读写访问

语法

Object.**StatusbarElementRename**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

可选项。用于指定所选用户自定义的状态栏元素名称的值或常量。

参见

StatusbarElementIndex (页 1144)

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarElementRepos

说明

指定对象状态栏中使用"StatusbarElementIndex"引用的元素位置。

如果已使用"StatusbarElementRepos"更改了元素位置，"StatusbarElementRepos"的值则会分配给"StatusbarElementIndex"。

运行系统中可进行读写访问

语法

Object.**StatusbarElementRepos**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于指定状态栏中引用的元素的位置的值或常量。值范围为 0 到 (StatusbarElementCount - 1)。超出该范围的值无效。

0：引用的元素放置在左侧。

参见

StatusbarElementIndex (页 1144)

StatusbarElementCount (页 1141)

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarElements

描述

在运行系统中无访问权限。

StatusbarElementText

说明

指定使用"StatusbarElementIndex"引用的用户定义的状态栏元素的文本。

运行系统中可进行读写访问

语法

Object.**StatusbarElementText**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

可选项。用于指定状态栏中所选元素的文本的值或常量。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarElementTooltipText

描述

指定使用"StatusbarElementIndex"引用的用户定义的状态栏元素的工具提示文本。

运行系统中可进行读写访问

语法

Object.**StatusbarElementTooltipText**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

可选项。用于指定所选用户自定义状态栏元素的工具提示文本的值或常量。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarElementUserDefined

描述

指定组态工程师是否已添加使用"StatusbarElementIndex"引用的状态栏元素作为新的用户自定义元素。

运行系统中可进行读写访问

语法

Object.**StatusbarElementUserDefined**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，由用户定义状态栏元素。

FALSE，由系统指定状态栏元素。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarElementVisible

说明

指定是否在对象中显示使用"StatusbarElementIndex"引用的状态栏元素。

运行系统中可进行读写访问

语法

Object.**StatusbarElementVisible**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。TRUE，显示引用的状态栏元素。

参见

StatusbarElementIndex (页 1144)

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarElementWidth

说明

指定使用"StatusbarElementIndex"引用的状态栏元素的宽度（以像素为单位）。

也可将 "StatusbarElementAutoSize (页 1140)"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**StatusbarElementWidth**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于指定所引用状态栏元素宽度（以像素为单位）的值或常量。

参见

StatusbarElementIndex (页 1144)

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarFont

说明

指定状态栏中文本的字体。

运行系统中的访问权限：读和写

语法

Object.**StatusbarFont**[=Font]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Font

可选项。 用于指定状态栏中文本字体的值或常量。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarFontColor

描述

指定状态栏中文本的颜色。

运行系统中可进行读写访问

语法

Object.**StatusbarFontColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Color

可选项。用于指定状态栏中文本颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarShowTooltips

描述

指定是否在运行系统中显示状态栏元素的工具提示。

运行系统中可进行读写访问

语法

Object.**StatusbarShowTooltips**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，显示工具提示。

FALSE，不显示工具提示。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarText

描述

指定状态栏中的默认文本。

运行系统中的访问权限：读和写

语法

Object.**StatusbarText**[=STRING]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

可选项。用于指定状态栏中默认文本的值或常量。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarUseBackColor

描述

指定是否显示状态栏的背景色。

运行系统中可进行读写访问

语法

Object.**StatusbarUseBackColor**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，显示状态栏的背景色。

FALSE，不显示状态栏的背景色。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatusbarVisible

描述

指定是否显示控件的状态栏。

运行系统中可进行读写访问

语法

Object.**StatusbarVisible**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，显示状态栏。

FALSE，不显示状态栏。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

StatuslineFont

说明

在运行系统中无访问权限。

StepBackColor

说明

指定步的背景色。

运行系统中可进行读写访问

语法

Object.**StepBackColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- S7GraphOverview

在运行系统中您没有以下格式的访问权限：

Color

可选项。用于指定步背景色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

S7GraphOverview (页 476)

StepFont

说明

指定步的前景色。

运行系统中可进行读写访问

语法

Object.**StepFont**[=Font]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- S7GraphOverview

Font

可选项。用于指定字体的值或常量。

参见

S7GraphOverview (页 476)

StepSeconds

说明

指定启动 "向前"(Forward) 或 "向后"(Backward) 按钮后的步间隔（以秒为单位）。

运行系统中的访问权限：读和写

语法

Object.**StepSeconds**[=Int32]

Object

必需项。 "ScreenItem"对象，且具有以下特性：

- MediaPlayer

Int32

可选项。 用于指定启动“向前”(Forward) 或“向后”(Backward) 按钮后的步间隔（以秒为单位）的值或常量。

参见

MediaPlayer (页 396)

StepTextColor

说明

指定步的文本颜色。

运行系统中可进行读写访问

语法

Object.**StepTextColor**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- S7GraphOverview

在运行系统中您没有以下格式的访问权限：

Color

可选项。用于指定步背景色的值或常量。

注释

可以使用“RGB”函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

S7GraphOverview (页 476)

Style

描述

指定线样式。

运行系统中的访问权限：读和写

语法

Object.**Style**[=LineStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- CircularArc
- Connector
- EllipticalArc
- Line
- Polyline

LineStyle

可选项。用于指定线样式的值或常量。值范围为 0 到 4。

值	VB 常量	说明
0	hmiLineStyleSolid	实线
1	hmiLineStyleDash	短划线
2	hmiLineStyleDot	虚线
3	hmiLineStyleDashDot	点划线
4	hmiLineStyleDashDotDot	双点划线

StyleItem

说明

在运行系统中无访问权限。

StyleSettings

描述

指定对象的显示样式。

运行系统中可进行读写访问

语法

Object.**StyleSettings**[=WinCCStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Button
- RoundButton*
- Window Slider*

*：只读访问权限

WinCCStyle

可选项。用于指定以何种样式显示对象的值或常量。

值	名称	说明
0	用户定义	指定根据设置显示对象。
1	全局	指定按全局设置的设计显示对象。
2	Windows 样式	指定以 Windows 样式显示对象。

SupportsInplaceEdit

说明

在运行系统中无访问权限。

SupportsS7DiagnosticsInSimpleView

说明

在运行系统中无访问权限。

SupportsUserDefinedToolbarButtons

说明

在运行系统中无访问权限。

SwapFirstWithLastConnection

描述

指定应交换第一个连接和最后一个连接。

运行系统中的访问权限：读和写

语法

Object.**SwapFirstWithLastConnection**[=Boolean]

Object

必选项。具有以下格式的"ScreenItem"类型的对象：

- Connector

Boolean

可选项。用于指定交换第一个连接和最后一个连接的值或常量。

参见

Connector (页 330)

SwitchOrientation

说明

在运行系统中无访问权限。

SymbolTableFont

说明

指定 PLC 代码显示中符号表的字体。

运行系统中的访问权限：读和写

语法

Object.**SymbolTableFont**[=Font]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- PLCCodeViewer

Font

可选项。 用于指定 PLC 代码显示中符号表字体的值或常量。

参见

PLCCodeViewer (页 442)

SysDiagBuffButtonVisible

说明

在运行系统中无访问权限。

1.5.5.13 属性 T

TabIndex

说明

指定在 Alpha 光标模式下使用跳格键在对象之间进行切换时该对象在顺序中所处的位置。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读取

语法

Object.**TabIndex**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- DiscSpaceView

在运行系统中您没有以下格式的访问权限：

-

Int32

可选项。用于指定跳格键顺序中的位置的值或常量。

TabIndexAlpha

说明

指定在 Alpha 光标模式下使用 tab 键在对象之间进行切换时该对象在顺序中所处的位置。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读取

语法

Object.**TabIndexAlpha**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- DiscSpaceView

在运行系统中您没有以下格式的访问权限：

- AlarmControl
- AlarmView
- ApplicationWindow
- Bar
- BatteryView
- Button
- ChannelDiagnose
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Clock
- ComboBox
- Connector
- DateTimeField
- Ellipse
- EllipseSegment
- EllipticalArc
- FunctionTrendControl
- Gauge
- GraphicIOField
- GraphicView
- HTMLBrowser
- IOField
- Line

- ListBox
- MediaPlayer
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- OptionGroup
- PLCCodeViewer
- Polygon
- Polyline
- ProtectedAreaNameView
- RangeLabelView
- RangeQualityView
- RecipeView
- Rectangle
- RoundButton
- S7GraphOverview
- ScreenWindow
- Slider
- SmartClientView
- StatusForce
- Switch
- SymbolLibrary
- SymbolicIOField
- SysDiagControl
- TextField
- TrendRulerControl
- TrendView
- TubeArcObject
- TubeDoubleTeeObject

- TubeTeeObject
- Tubepolyline
- UserArchiveControl
- UserView
- WLanQualityView
- WindowsSlider
- ZoneLabelView
- ZoneQualityView

Int32

可选项。用于指定 tab 键顺序中的位置的值或常量。

TableBackColor

描述

指定表格单元格的背景色。

运行系统中可进行读写访问

语法

Object.**TableBackColor**[=Color]

对象

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmView
- RecipeView*
- StatusForce
- TrendView
- UserView

*：只读访问权限

颜色

可选项。用于确定表格单元格背景色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

TableColor

说明

指定对象中表格行的背景色。

运行系统中可进行读写访问

语法

Object.**TableColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

可选项。用于指定表格行背景色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

TableColor2

说明

指定对象中表格行的第二种背景色。

也可将“UseTableColor2 (页 1364)”设置为“TRUE”。

运行系统中可进行读写访问

语法

Object.**TableColor2**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

可选项。用于指定表格行第二种背景色的值或常量。

注释

可以使用“RGB”函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

TableColumnsWidthAndOrder

说明

在运行系统中无访问权限。

TableEvenRowBackColor

说明

在运行系统中无访问权限。

TableFont

说明

在运行系统中无访问权限。

TableForeColor

说明

指定对象表格单元格中采用的文本颜色。

运行系统中的访问权限：读和写

语法

Object.**TableForeColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- AlarmView
- OnlineTableControl
- RecipeView*
- StatusForce
- TrendRulerControl
- UserArchiveControl
- UserView

*：只读访问权限

Color

可选项。用于指定表格单元格采用的文本颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

TableForeColor2

说明

指定对象表格单元格中采用的第二种文本颜色。

也可将"UseTableColor2 (页 1364)"设置为"TRUE"。

运行系统中可进行读写访问

语法

Object.**TableForeColor2**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

可选项。用于指定表格单元格中所用第二种字体颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

TableGridLineColor

说明

指定表格中网格线的颜色。

运行系统中的访问权限：

- RT Advanced：读和写
- RT Professional：无访问权

语法

Object.**TableGridLineColor**[=Color]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- RecipeView*
- TrendView*
- UserView

*：只读访问权限

在运行系统中您没有以下格式的访问权限：

- StatusForce
- SysDiagControl

Color

可选项。用于确定表格网格线颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

TableHeaderBackColor

描述

指定表格标题的背景色。

运行系统中可进行读写访问

语法

Object.**TableHeaderBackColor**[=Color]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmView
- RecipeView*
- StatusForce
- TrendView*
- UserView

*：只读访问权限

Color

可选项。用于指定标题背景色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

TableHeaderBackFillStyle

说明

在运行系统中无访问权限。

TableHeaderBorderBackColor

说明

在运行系统中无访问权限。

TableHeaderBorderColor

说明

在运行系统中无访问权限。

TableHeaderBorderWidth

说明

在运行系统中无访问权限。

TableHeaderCornerRadius

说明

在运行系统中无访问权限。

TableHeaderEdgeStyle

说明

在运行系统中无访问权限。

TableHeaderFirstGradientColor

说明

在运行系统中无访问权限。

TableHeaderFirstGradientOffset

说明

在运行系统中无访问权限。

TableHeaderFont

说明

在运行系统中无访问权限。

TableHeaderForeColor

描述

指定表格标题的文本颜色。

运行系统中可进行读写访问

语法

Object.**TableHeaderForeColor**[=Color]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmView
- RecipeView*
- StatusForce*
- TrendView*
- UserView

*：只读访问权限

Color

可选项。用于指定标题文本颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

TableHeaderMiddleGradientColor

说明

在运行系统中无访问权限。

TableHeaderPaddingBottom

说明

在运行系统中无访问权限。

TableHeaderPaddingLeft

说明

在运行系统中无访问权限。

TableHeaderPaddingRight

说明

在运行系统中无访问权限。

TableHeaderPaddingTop

说明

在运行系统中无访问权限。

TableHeaderSecondGradientColor

说明

在运行系统中无访问权限。

TableHeaderSecondGradientOffset

说明

在运行系统中无访问权限。

Tag4DataRecord

说明

在运行系统中无访问权限。

Tag4RecipeNumber

说明

在运行系统中无访问权限。

TagForExternalTime

说明

在运行系统中无访问权限。

TagPrefix

说明

指定变量前缀，以添加到画面窗口中所包含的所有变量前。这样一来，当另一个画面访问其它变量时，嵌入在画面窗口中的画面仍能访问其自己的变量。

重新加载画面之后，变量前缀的更改才会生效。画面变化期间，此更改会自动生效。如果不更换画面，则只有重新分配画面名称时此更改才会生效。

前缀可自由定义，但必须与结构变量的名称相匹配。

说明

分配用户数据类型

还可以为集成在画面窗口中的画面选择一个用户数据类型。之后，会直接从所选用户数据类型中分配数据类型元素。

说明

变量前缀与间接寻址的变量

避免同时使用变量前缀和间接寻址的变量。

运行系统中的访问权限：读和写

语法

Object.**TagPrefix**[= STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- ScreenWindow

STRING

可选项。用于指定变量前缀的值或常量。

示例

画面窗口中应显示有"InOutput"画面。"InOutput"画面包含 3 个与结构变量相连的 I/O 域。结构变量包含元素 .IO1、.IO2、.IO3；每个 I/O 域对应一个元素。元素名称前面含有时间。该时间是指使用正确的语法将结构变量的元素作为结构元素进行寻址所需的时间。

例如，在项目中定义了结构名称为 Struct1、Struct2 和 Struct3 这三个结构变量。

在这种情况下，变量前缀即为结构名称。例如，将"Struct2"指定为变量前缀。I/O 域随即可链接到结构变量"Struct2"的元素。

变量前缀："Struct2"：

- 输出值（第一个 IO 域）：.IO1
- 输出值（第二个 I/O 域）：.IO2
- 输出值（第三个 I/O 域）：.IO3

因此画面窗口中当前的变量连接为：

- 输出值（第一个 I/O 域）：Struct2.EA1
- 输出值（第二个 I/O 域）：Struct2.EA2
- 输出值（第三个 I/O 域）：Struct2.EA3

参见

ScreenWindow (页 479)

变量

说明

返回“Tags”类型对象。

运行中可进行的访问：读取

语法

Object.**Tags**

Object

必需。它是“HMIRuntime”类型的对象。

示例

以下示例为访问变量“Tag1”：

```
'VBS86
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
```

参见

HMIRuntime（页 224）

TcpPortNr

说明

在运行系统中无访问权限。

Template

说明

指定在“打印作业/脚本诊断”对象中显示窗口内容所使用的模板。

运行系统中的访问权限： 读和写

语法

Object.**Template**[=TemplateType]

Object

必需项。 具有以下格式的“ScreenItem”类型的对象：

- ApplicationWindow

TemplateType

可选项。 用于指定模板的值或常量。

视属性“WindowsContents (页 1439)”的值而定，可有以下模板：

窗口内容 = 全局脚本

- “GSC 诊断”
  “脚本诊断”对象由全局脚本的应用程序提供。 显示诊断系统的结果。
- “GSC 运行系统”
  “脚本诊断”对象由全局脚本的应用程序提供。 显示运行系统中关于特征的分析结果。

窗口内容 = 打印作业

- “所有作业”：
  “打印工作”对象由记录系统提供。 可用报表按列表显示。
- “所有作业 - 上下文菜单”：
  “打印工作”对象由记录系统提供。 可用报表按列表显示。 可通过快捷菜单选择打印选项、显示打印预览以及打印输出记录。

- “作业详细信息视图”：
  “打印工作”对象由记录系统提供。 可用报表在选择菜单中显示。 显示所选报表的详细信息。
- “所选作业 - 上下文菜单”：
  “打印工作”对象由记录系统提供。 可用报表按列表显示。 该列表只包含已经在“打印作业属性”(Print Job Properties) 对话框中激活“标记打印作业列表”(Mark for print job list) 选项的报表。 可通过快捷菜单选择打印选项、显示打印预览以及打印输出记录。

参见

ApplicationWindow (页 282)

Text

描述

指定文本域的标签。

运行系统中的访问权限：读和写

语法

Object.**Text**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- CheckBox
- ComboBox
- ListBox
- MultiLineEdit
- OptionGroup
- RoundButton
- TextField

STRING

可选项。用于指定标签的值或常量。

TextAreaBottomMargin

说明

在运行系统中无访问权限。

TextAreaLeftMargin

说明

在运行系统中无访问权限。

TextAreaRightMargin

说明

在运行系统中无访问权限。

TextAreaTopMargin

说明

在运行系统中无访问权限。

TextHandles

说明

在运行系统中无访问权限。

TextList

描述

返回为对象提供值的文本列表。

运行系统中的访问权限

- RT Advanced：无访问权
- RT Professional：读取

语法

Object.**TextList**[=HmiObjectHandle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- SymbolicIOField

在运行系统中您没有以下格式的访问权限：

- Button

HmiObjectHandle

可选项。用于返回为对象提供值的文本列表的值或常量。

参见

SymbolicIOField (页 504)

Button (页 296)

TextOff

描述

指定在"关闭"状态下将显示的文本。

运行系统中可进行读写访问

语法

Obje ct.**TextOff**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Button
- Switch*

* RT Advanced 只读访问权限，RT Professional 无访问权限

在运行系统中您没有以下格式的访问权限：

- SymbolicIOField

STRING

可选项。用于指定“关闭”状态标签的值或常量。

注释

只有引用的对象为"Text"类型时，该属性才可用。

TextOn

描述

指定在“打开”状态下将显示的文本。

运行系统中的访问权限：读和写

语法

Object.**TextOn**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Button
- Switch*

* RT Advanced 只读访问权限，RT Professional 无访问权限

在运行系统中您没有以下格式的访问权限：

- SymbolicIOField

STRING

可选项。用于指定“打开”状态标签的值或常量。

注释

只有引用的对象为“Text”类型时，该属性才可用。

TextOrientation

描述

指定文本方向。

运行系统中可进行的访问：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**TextOrientation**[=TextOrientation]

Object

必选。具有以下格式的“ScreenItem”类型的对象：

- Button
- CheckBox
- IOField
- OptionGroup
- RoundButton*
- SymbolicIOField*
- TextField
- WindowSlider*

*：只读访问权限

在运行系统中您没有以下格式的访问权限：

- DateTimeField
- Switch

TextOrientation

可选项。用于指定文本方向的值或常量。

值	VB 常量	说明
0	hmiTextHorizontal	文本水平显示。
-1	hmiTextRotated90Degree	文本垂直显示且左对齐。
1	hmiTextRotated270Degree	文本垂直显示且右对齐。

Texts

说明

在运行系统中无访问权限。

TextualObjectPositions

说明

在运行系统中无访问权限。

TextualObjectsAutoSize

说明

在运行系统中无访问权限。

TextualObjectsBorderBackColor

说明

在运行系统中无访问权限。

TextualObjectsBorderColor

说明

在运行系统中无访问权限。

TextualObjectsBorderWidth

说明

在运行系统中无访问权限。

TextualObjectsCornerRadius

说明

在运行系统中无访问权限。

TextualObjectsEdgeStyle

说明

在运行系统中无访问权限。

TextualObjectsPaddingBottom

说明

在运行系统中无访问权限。

TextualObjectsPaddingLeft

说明

在运行系统中无访问权限。

TextualObjectsPaddingRight

说明

在运行系统中无访问权限。

TextualObjectsPaddingTop

说明

在运行系统中无访问权限。

ThumbBackColor

描述

指定滑块的背景色。

运行系统中可进行读访问

语法

Object.**ThumbBackColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Slider
- WindowSlider

Color

可选项。用于指定滚动条背景色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Slider (页 485)

WindowSlider (页 586)

ThumbPicture

说明

为滑块中的滑块元素指定图形。

运行系统中的访问权限：读取

语法

Object.**ThumbPicture**[=HmiObjectHandle]

Object

必需项。“ScreenItem”对象，且具有以下特性：

- Slider

HmiObjectHandle

可选项。用于指定滑块中滑块元素图形的值或常量。

参见

Slider (页 485)

TickDistance

说明

在运行系统中无访问权限。

TicksColor

描述

指定“Clock”对象盘面的小时标记的颜色。

运行系统中可进行读写访问

语法

Object.**TicksColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Clock

Color

可选项。用于指定小时标记线颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Clock (页 323)

TickStyle

描述

指定如何在刻度中显示标记。

也可将"ShowTicks"设置为"TRUE"。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**TickStyle**[=SliderTickStyle]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Slider

在运行系统中您没有以下格式的访问权限：

- Clock

SliderTickStyle

可选项。用于指定如何在刻度中显示标记的值或变量。

值	VB 常量	说明
0	hmiSliderTickStyleNone	刻度没有任何标记。
1	hmiSliderTickStyleEffect1	以白色并带有黑色阴影的标记显示大刻度线。以黑色显示小刻度线。
2	hmiSliderTickStyleEffect2	以黑色并带有白色阴影的标记显示大刻度线。以黑色显示小刻度线。
3	hmiSliderTickStyleNormal	以黑色显示所有刻度线。

注释

自动标定会在刻度中显示两条彼此紧挨的线，以使其看起来像一条粗线。要修正这种情况，请延长或缩短滚动条。

TimeAxes

说明

在运行系统中无访问权限。

TimeAxisAdd

说明

创建新的时间轴。新创建的时间轴会自动使用“TimeAxisIndex”而引用。

运行系统中的访问权限：读和写

语法

Object.**TimeAxisAdd**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

STRING

必需项。用于通过"TimeAxisName"指定新时间轴名称的值或常量。

TimeAxisAlignment

说明

指定时间轴的对齐方式。

运行系统中的访问权限：读和写

语法

Object.**TimeAxisAlignment**[=AxisAlignment]

Object

必需项。 "ScreenItem"对象，且具有以下特性：

- OnlineTrendControl

AxisAlignment

可选项。 用于指定对齐方式的值或常量。

值	名称	说明
0	底部	所选时间轴显示在趋势下方。
1	顶部	所选时间轴显示在趋势上方。

参见

OnlineTrendControl (页 418)

TimeAxisBegin

说明

在运行系统中无访问权限。

TimeAxisBeginTime

说明

指定用于显示通过"TimeAxisIndex"所引用的时间轴的开始时间。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**TimeAxisBeginTime**[=DateTime]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

在运行系统中您没有以下格式的访问权限：

- TrendView

DateTime

可选项。用于指定显示特定趋势起始时间的值或常量。

TimeAxisColor

说明

指定使用"TimeAxisIndex"引用的时间轴的颜色。

运行系统中可进行读写访问

语法

Object.**TimeAxisColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

Color

可选项。用于指定时间轴颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

OnlineTrendControl (页 418)

TimeAxisCount

说明

指定所组态的时间轴数。

运行系统中可进行读写访问

语法

Object.**TimeAxisCount**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

Int32

可选项。用于指定所组态的时间轴数的值或常量。

参见

OnlineTrendControl (页 418)

TimeAxisCountPoints

说明

在运行系统中无访问权限。

TimeAxisDateFormat

说明

指定用于显示使用"TimeAxisIndex"引用的时间轴的日期格式。

运行系统中可进行读写访问

语法

Object.**TimeAxisDateFormat**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

STRING

可选项。用于指定日期格式的值或常量。

参见

OnlineTrendControl (页 418)

TimeAxisEnd

说明

在运行系统中无访问权限。

TimeAxisEndTime

描述

指定显示所选趋势的结束时间。是否评估该信息取决于“Autorange”、“UseTimeRange(i)”和“ShareTimeAxis”属性。

运行系统中的访问权限：读和写

语法

Object.**TimeAxisEndTime**[=DateTime]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- OnlineTrendControl

DateTime

可选项。用于指定显示特定趋势的结束时间的值或常量。

参见

OnlineTrendControl (页 418)

TimeAxisIndex

说明

引用时间轴。要访问时间轴的属性，需要设置“"TimeAxisIndex"”。

介于 0 至 (TimeAxisIndex - 1) 之间的值为“TimeAxisCount”的有效值。“TimeAxisCount”属性指定所组态时间轴的数目。

运行系统中可进行读写访问

语法

Object.**TimeAxisIndex**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

Int32

可选项。用于通过索引指定要编辑的时间轴的值或常量。

参见

TimeAxisCount (页 1198)

OnlineTrendControl (页 418)

TimeAxisInTrendColor

说明

指定使用"TimeAxisIndex"引用的时间轴的颜色是否与趋势颜色相对应。

运行系统中可进行读写访问

语法

Object.**TimeAxisInTrendColor**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

BOOLEAN

可选项。

TRUE，以趋势颜色显示引用的轴。"TimeAxisColor"值无效。

FALSE，以"TimeAxisColor"指定的颜色显示引用的轴。

参见

TimeAxisIndex (页 1200)

OnlineTrendControl (页 418)

TimeAxisLabel

描述

指定使用"TimeAxisIndex"引用的时间轴的标签文本。

运行系统中可进行读写访问

语法

Object.**TimeAxisLabel**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

STRING

可选项。用于指定引用的时间轴的标签文本的值或常量。

参见

TimeAxisIndex (页 1200)

OnlineTrendControl (页 418)

TimeAxisMeasurePoints

说明

指定使用"TimeAxisIndex"引用的时间轴显示的测量点数。

运行系统中可进行读写访问

语法

Object.**TimeAxisMeasurePoints**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

Int32

可选项。用于指定测量点数量的值或常量。

参见

OnlineTrendControl (页 418)

TimeAxisMode

说明

在运行系统中无访问权限。

TimeAxisName

说明

指定使用"TimeAxisIndex"引用的时间轴的名称。

运行系统中可进行读写访问

语法

Object.**TimeAxisName**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

STRING

可选项。用于指定引用的时间轴名称的值或常量。

参见

TimeAxisIndex (页 1200)

OnlineTrendControl (页 418)

TimeAxisOnline

说明

指定是否持续更新时间轴。

运行系统中的访问权限：读和写

语法

Object.**TimeAxisOnline**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- OnlineTrendControl

BOOLEAN

可选项。

TRUE，以持续更新时间轴。

FALSE，不持续更新时间轴。

参见

OnlineTrendControl (页 418)

TimeAxisRange

描述

在运行系统中无访问权限。

TimeAxisRangeType

说明

定义使用“TimeAxisIndex”引用的时间轴的时间范围设置。

运行系统中可进行读写访问

语法

Object.**TimeAxisRangeType**[=TimeRangeMode]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

TimeRangeMode

可选项。用于指定时间范围设置的值或常量。

有下列设置可用：

值	名称	说明
0	时间范围	指定时间列的开始时间和时间范围。
1	结束时间	指定时间列的开始时间和结束时间。
2	测量点	定义时间列的开始时间和测量点数量。

参见

OnlineTrendControl (页 418)

TimeAxisRemove

说明

使用引用的时间轴的名称可移除该时间轴。

运行系统中可进行读写访问

语法

Object.**TimeAxisRemove**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

STRING

可选项。可返回要移除的引用的时间轴名称的值或常量。

参见

OnlineTrendControl (页 418)

TimeAxisRename

说明

指定使用"TimeAxisIndex"引用的时间轴的新名称。

运行系统中可进行读写访问

语法

Object.**TimeAxisRename**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

STRING

可选项。用于指定所选时间轴新名称的值或常量。

参见

TimeAxisIndex (页 1200)

OnlineTrendControl (页 418)

TimeAxisRepos

说明

在对象的趋势图中指定使用"TimeAxisIndex"引用的时间轴位置。

如果已使用"TimeAxisRepos"更改了元素位置，"TimeAxisRepos"的值则会分配给"TimeAxisIndex"。

运行系统中可进行读写访问

语法

Object.**TimeAxisRepos**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

Int32

可选项。用于指定所引用时间轴位置的值或常量。值范围为 0 到 (TimeAxisCount - 1)。超出该范围的值无效。

0：引用的时间轴放置于外部。

参见

TimeAxisIndex (页 1200)

TimeAxisCount (页 1198)

OnlineTrendControl (页 418)

TimeAxisShowDate

说明

指定是否使用日期和时间显示使用"TimeAxisIndex"引用的时间轴。

运行系统中可进行读写访问

语法

Object.**TimeAxisShowDate**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

BOOLEAN

可选项。

TRUE，显示日期和时间。使用"TimeAxisDateFormat"属性指定日期格式。

FALSE，不显示日期。仅显示时间。

参见

OnlineTrendControl (页 418)

TimeAxisSide

说明

在运行系统中无访问权限。

TimeAxisTimeFormat

描述

指定沿所选趋势的时间轴的信息格式。

运行系统中可进行读写访问

语法

Object.**TimeAxisTimeFormat**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

STRING

可选项。用于定义时间轴格式的值或常量。

参见

OnlineTrendControl (页 418)

TimeAxisTimeRange

说明

在运行系统中无访问权限。

TimeAxisTimeRangeBase

说明

指定用于定义时间范围和时间因素"TimeAxisTimeRangeFactor"的时间单位。

运行系统中可进行读写访问

语法

Object.**TimeAxisTimeRangeBase**[=TagLoggingTimeUnit]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

TagLoggingTimeUnit

可选项。用于指定时间单位的值或常量。

有下列设置可用：

值	名称
500	500 ms
1000	1 秒
60000	1 分钟
3600000	1 小时
86400000	1 天

参见

OnlineTrendControl (页 418)

TimeAxisTimeRangeFactor

说明

指定与时间单位"TimeAxisTimeRangeBase"一起确定时长的时间因数。

运行系统中的访问权限：读和写

语法

Object.**TimeAxisTimeRangeFactor**[=Int16]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- OnlineTrendControl

Int16

可选项。用于指定时间因数的值或常量。

参见

OnlineTrendControl (页 418)

TimeAxisTrendWindow

说明

指定使用“TimeAxisIndex”引用的轴在其中显示的趋势图。

运行系统中可进行读写访问

语法

Object.**TimeAxisTrendWindow**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- OnlineTrendControl

STRING

可选项。用于指定趋势图名称的值或常量。

参见

TrendWindowCount (页 1329)

TimeAxisIndex (页 1200)

OnlineTrendControl (页 418)

TimeAxisVisible

说明

指定是否在对象中显示使用“TimeAxisIndex”引用的时间轴。

运行系统中可进行读写访问

语法

Object.**TimeAxisVisible**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- OnlineTrendControl

BOOLEAN

可选项。

TRUE，显示引用的时间轴。

FALSE，隐藏引用的时间轴。

参见

TimeAxisIndex (页 1200)

OnlineTrendControl (页 418)

TimeBase

描述

指定显示时间值的时区。

运行系统中可进行读写访问

语法

Object.**TimeBase**[=TimeBase]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- UserArchiveControl

TimeBase

可选项。用于指定时区的值或常量。

值	VB 常量	说明
0	hmiTimeBaseLocalTimezone	本地时间
1	hmiTimeBaseServerTimezone	服务器时区
2	hmiTimeBaseUTC	UTC (Universal Time Coordinated)
3	hmiTimeBaseProjectSetting	项目设置

注释

可在项目树，HMI 设备的对象属性中为个别 HMI 设备指定时间模式。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

用户归档控件 (页 566)

TimeColumnActualize

描述

指定是否更新使用"TimeColumnIndex"引用的时间列中的值。

运行系统中可进行读写访问

语法

Object.**TimeColumnActualize**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

BOOLEAN

可选项。

TRUE，更新时间列。

FALSE，不更新时间列。该设置对于比较表格有用。

参见

OnlineTableControl (页 402)

TimeColumnAdd

描述

创建新的时间列。新创建的时间列会自动使用"TimeColumnIndex"而引用。

运行系统中可进行读写访问

语法

Object.**TimeColumnAdd**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

STRING

必需项。用于通过"TimeColumnName"指定新时间列名称的值或常数。

参见

TimeColumnIndex (页 1221)

OnlineTableControl (页 402)

TimeColumnAlignment

描述

指定使用"TimeColumnIndex"引用的时间列中的文本如何对齐。

运行系统中可进行读写访问

语法

Object.**TimeColumnAlignment** [=HorizontalAlignment]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

HorizontalAlignment

可选项。用于指定引用的时间列中的文本如何对齐的值或常量。

值	VB 常量	描述
0	hmiAlignmentLeft	文本左对齐。
1	hmiAlignmentCentered	文本居中。
2	hmiAlignmentRight	文本右对齐。

参见

OnlineTableControl (页 402)

TimeColumnBackColor

描述

指定使用"TimeColumnIndex"引用的时间列的背景色。

也可将 "UseColumnBackColor (页 1351)"设置为"TRUE"，以及将 "TimeColumnUseValueColumnColors (页 1231)"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**TimeColumnBackColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

Color

可选项。用于指定所选时间列背景色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

OnlineTableControl (页 402)

TimeColumnBeginTime

描述

为使用"TimeColumnIndex"引用的时间列指定时间范围的起始点。

运行系统中可进行读写访问

语法

Object.**TimeColumnBeginTime**[=DateTime]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

DateTime

可选项。用于指定所选时间列起始时间的值或常量。

参见

OnlineTableControl (页 402)

TimeColumnCaption

描述

指定使用"TimeColumnIndex"引用的时间列的标签。

运行系统中可进行读写访问

语法

Object.**TimeColumnCaption**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

STRING

可选项。用于指定时间列标题的值或常量。

参见

OnlineTableControl (页 402)

TimeColumnCount

描述

指定已组态的时间列数。

运行系统中的访问权限：读和写

语法

Object.**TimeColumnCount**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- OnlineTableControl

Int32

可选项。用于指定已组态的时间列数的值或常量。

参见

OnlineTableControl (页 402)

TimeColumnDateFormat

描述

指定用于显示使用“TimeColumnIndex”引用的时间列的日期格式。

运行系统中的访问权限：读和写

语法

Object.**TimeColumnDateFormat**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- OnlineTableControl

STRING

可选项。用于指定显示所选时间列的日期格式的值或常量。

以下日期格式可用：

值	描述
dd.MM.yy	日.月.年，例如，24.12.13。
dd.yyyyd.MM	日.月.年，例如，24.12.2013。
dd/MM/yy	日/月/年，例如，24/12/13。
dd/MM/yyyy	日/月/年，例如，24/12/2013。

TimeColumnEndTime

描述

为使用“TimeColumnIndex”引用的时间列定义时间范围的终点。

运行系统中可进行读写访问

语法

Object.**TimeColumnBeginTime**[=DateTime]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- OnlineTableControl

DateTime

可选项。用于指定结束时间的值或常量。

参见

OnlineTableControl (页 402)

TimeColumnForeColor

描述

指定使用“TimeColumnIndex”引用的时间列的字体颜色。

也可将“UseColumnForeColor (页 1352)”设置为“TRUE”，以及将“TimeColumnUseValueColumnColors (页 1231)”设置为“FALSE”。

运行系统中可进行读写访问

语法

Object.**TimeColumnForeColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

Color

可选项。用于指定所选时间列字体颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

OnlineTableControl (页 402)

TimeColumnHideText

描述

指定是否将使用"TimeColumnIndex"引用的时间列的内容显示为文本。

运行系统中可进行读写访问

语法

Object.**TimeColumnHideText**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

BOOLEAN

可选项。

TRUE，内容不以文本形式显示。

FALSE，内容以文本形式显示。

参见

OnlineTableControl (页 402)

TimeColumnHideTitleText

描述

指定是否将使用"TimeColumnIndex"引用的时间列的标题显示为文本。

运行系统中可进行读写访问

语法

Object.**TimeColumnHideTitleText**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

BOOLEAN

可选项。

TRUE，标题不以文本形式显示。

FALSE，标题以文本形式显示。

参见

OnlineTableControl (页 402)

TimeColumnIndex

描述

引用一个已组态的时间列。要访问时间列的属性，需要设置""TimeColumnIndex""。

介于 0 至 (TimeColumnCount - 1) 之间的值为"TimeColumnIndex"的有效值。属性"TimeColumnCount"指定所组态时间列的数目。

运行系统中的访问权限：读和写

语法

Object.**TimeColumnIndex**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

Int32

可选项。用于通过索引指定要编辑的时间列的值或常量。

TimeColumnLength

描述

指定使用"TimeColumnIndex"引用的时间列的宽度。

运行系统中可进行读写访问

语法

Object.**TimeColumnLength**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

Int32

可选项。用于指定所选时间列宽度的值或常量。

参见

OnlineTableControl (页 402)

TimeColumnMeasurePoints

描述

指定要显示在使用"TimeColumnIndex"引用的时间列中的测量点的数量。

运行系统中可进行读写访问

语法

Object.**TimeColumnMeasurePoints**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

Int32

可选项。用于指定测量点数量的值或常量。

参见

OnlineTableControl (页 402)

TimeColumnName

描述

指定使用"TimeColumnIndex"引用的时间列的名称。

运行系统中可进行读写访问

语法

Object.**TimeColumnName**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

STRING

可选项。用于指定引用的时间列名称的值或常量。

参见

TimeColumnIndex (页 1221)

OnlineTableControl (页 402)

TimeColumnRangeType

描述

为使用"TimeColumnIndex"引用的时间列指定时间范围。

运行系统中可进行读写访问

语法

Object.**TimeColumnRangeType**[=TimeRangeMode]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

TimeRangeMode

可选项。用于指定所选时间列时间范围的值或常量。

值	描述	说明
0	时间范围	指定时间列的开始时间和时间范围。
1	开始到结束的时间	指定时间列的开始时间和结束时间。
2	测量点数量	定义时间列的开始时间和测量点数量。

参见

OnlineTableControl (页 402)

TimeColumnRemove

描述

使用引用的时间列的名称可移除该时间列。

运行系统中可进行读写访问

语法

Object.**TimeColumnRemove**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

STRING

可选项。可返回要移除的引用的时间列名称的值或常量。

参见

OnlineTableControl (页 402)

TimeColumnRename

描述

指定使用"TimeColumnIndex"引用的时间列的新名称。

运行系统中可进行读写访问

语法

Object.**TimeColumnRename**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

STRING

可选项。用于指定所选时间列新名称的值或常量。

参见

TimeColumnIndex (页 1221)

OnlineTableControl (页 402)

TimeColumnRepos

描述

对于带有对应数值列的多个时间列，指定使用“TimeColumnIndex”引用的时间列的位置。

如果已使用“TimeColumnRepos”更改了列位置，“TimeColumnRepos”的值则会分配给“TimeColumnIndex”。

运行系统中可进行读写访问

语法

Object.**TimeColumnRepos**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- OnlineTableControl

Int32

可选项。用于指定带有对应数值列的引用的时间列位置的值或常量。值范围为 0 到 (TimeColumnCount - 1)。超出该范围的值无效。

0：引用的列放置在左侧。

参见

TimeColumnIndex (页 1221)

TimeColumnCount (页 1217)

OnlineTableControl (页 402)

TimeColumns

说明

在运行系统中无访问权限。

TimeColumnShowDate

描述

指定是否使用日期和时间显示使用"TimeColumnIndex"引用的时间列。

也可使用 "TimeColumnDateFormat (页 1217)"设置日期格式。

运行系统中可进行读写访问

语法

Object.**TimeColumnShowDate**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

BOOLEAN

可选项。

TRUE，同时显示日期和时间。

FALSE，仅显示时间。

参见

OnlineTableControl (页 402)

TimeColumnShowIcon

描述

指定是否将使用"TimeColumnIndex"引用的时间列的内容显示为图标。

运行系统中可进行的访问：读和写

语法

Object. [=BOOLEAN]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

BOOLEAN

可选项。

TRUE，内容以图标形式显示。

FALSE，内容不以图标形式显示。

TimeColumnShowTitleIcon

描述

指定是否将使用"TimeColumnIndex"引用的时间列的标题以图标形式显示。

运行系统中可进行读写访问

语法

Object.**TimeColumnShowTitleIcon**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

BOOLEAN

可选项。

TRUE，标题以图标形式显示。

FALSE，标题不以图标形式显示。

参见

OnlineTableControl (页 402)

TimeColumnSort

描述

指定使用“TimeColumnIndex”引用的时间列如何排序。

运行系统中可进行读写访问

语法

Object.**TimeColumnSort**[=SortMode]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- OnlineTableControl

SortMode

可选项。指定“TimeColumnIndex”中引用的时间列如何排序。

值	描述	说明
0	无	无排序
1	升序	升序，从最小值开始。
2	降序	降序，从最大值开始。

参见

OnlineTableControl (页 402)

TimeColumnSortIndex

描述

指定“TimeColumnIndex”所引用的时间列的排序顺序。如果将该值设置为“0”，将从“TimeColumnSort”中移除排序标准。

运行系统中的访问权限：读和写

语法

Object.**TimeColumnSortIndex**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

Int32

可选项。用于指定"TimeColumnIndex"所引用的时间列排序顺序的值或常量。如果将该值设置为"0"，将从"TimeColumnSort"中移除排序标准。

TimeColumnTimeFormat

描述

指定用于显示使用"TimeColumnIndex"引用的时间列的时间格式。

运行系统中可进行读写访问

语法

Object.**TimeColumnTimeFormat**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

STRING

可选项。用于指定显示所选时间列的时间格式的值或常量。

值	描述
自动	自动设置时间格式。
HH:mm:ss.ms	时:分:秒，例如，15:35:44.240。
hh:mm:ss tt	时:分:秒 AM/PM，例如 03:35:44 PM。
hh:mm:ss.ms tt	时:分:秒.毫秒 AM/PM，例如 03:35:44.240 PM。

参见

OnlineTableControl (页 402)

TimeColumnTimeRangeBase

描述

指定用于计算在使用"TimeColumnIndex"引用的时间列中显示的时间范围的时间单位。

使用时间单位和时间因素可计算时间范围。时间因素在"TimeColumnTimeRangeFactor (页 1230)"中指定。

运行系统中可进行读写访问

语法

Object.**TimeColumnTimeRangeBase**[=TagLoggingTimeUnit]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

TagLoggingTimeUnit

可选项。用于指定确定时间范围的时间单位的值或常量。

值	描述
500	500 ms
1000	1 秒
60000	1 分钟
3600000	1 小时
86400000	1 天

参见

OnlineTableControl (页 402)

TimeColumnTimeRangeFactor

描述

定义用于计算时间范围的系数。 只有整数系数有效。

运行系统中的访问权限：读和写

语法

Object.**TimeColumnTimeRangeFactor**[=Int16]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- OnlineTableControl

Int16

可选项。用于指定确定时间范围的系数的值或常量。

参见

OnlineTableControl (页 402)

TimeColumnUseValueColumnColors

描述

指定是否以数值列的颜色显示使用"TimeColumnIndex"引用的时间列。

运行系统中可进行读写访问

语法

Object.**TimeColumnUseValueColumnColors**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

BOOLEAN

可选项。

TRUE，数值列颜色中显示所选时间列。"TimeColumnBackColor (页 1215)""TimeColumnForeColor (页 1218)"中的设置禁用。

FALSE，"TimeColumnBackColor (页 1215)"和"TimeColumnForeColor (页 1218)"指定的颜色中显示所选时间列。

参见

OnlineTableControl (页 402)

TimeColumnVisible

描述

指定是否在表格视图中显示使用"TimeColumnIndex"引用的时间列。

运行系统中可进行读写访问

语法

Object.**TimeColumnVisible**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

BOOLEAN

可选项。

TRUE，表格中显示引用的时间列。

FALSE，表格中不显示引用的时间列。

参见

TimeColumnIndex (页 1221)

OnlineTableControl (页 402)

TimeDisplayMode

说明

在运行系统中无访问权限。

TimeStamp

说明

将最后一次读取访问变量的本地时间的时间戳返回为 DATE。

运行系统中可进行的访问：Read

语法

Object.**TimeStamp**

Object

要求"Tag"对象。

注释

为了以纯文本形式显示 TimeStamp 属性，使用 VBS 函数"FormatDateTime(Date[, NamedFormat])"。 输出由语言设置决定。 要调整语言，使用 VBS 函数"SetLocale´()"。

若要按日期、工作日和时间返回时间戳，使用 NamedFormat 参数或 VBS 函数，例如 Year、WeekDay、Day、Hour、Minute、Second。 工作日名称可以通过 VBS 函数 WeekdayName 获取。

示例

以下示例通过函数“FormatDateTime”获取变量“Tag11”的时间戳：

```
'VBS87
Dim objTag
Dim lngCount
lngCount = 0
Set objTag = HMIRuntime.Tags("Tag11")
objTag.Read
SetLocale("en-gb")
MsgBox FormatDateTime(objTag.TimeStamp)     '输出： 例如 06/08/2002 9:07:50
MsgBox Year(objTag.TimeStamp)     '输出： 例如 2002
MsgBox Month(objTag.TimeStamp)     '输出： 例如 8
MsgBox Weekday(objTag.TimeStamp)     '输出： 例如 3
MsgBox WeekdayName(Weekday(objTag.TimeStamp))     '输出： 例如 星期二
MsgBox Day(objTag.TimeStamp)     '输出： 例如 6
MsgBox Hour(objTag.TimeStamp)     '输出： 例如 9
MsgBox Minute(objTag.TimeStamp)     '输出： 例如 7
MsgBox Second(objTag.TimeStamp)     '输出： 例如 50
For lngCount = 0 To 4
MsgBox FormatDateTime(objTag.TimeStamp, lngCount)
Next
'lngCount = 0: 输出： 例如 06/08/2002 9:07:50
'lngCount = 1: 输出： 例如 06 August 2002
'lngCount = 2: 输出： 例如 06/08/2002
'lngCount = 3: 输出： 例如 9:07:50
'lngCount = 4: 输出： 例如 9:07
```

以下示例为获取变量“Tag1”的时间戳：

```
'VBS88
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Read
MsgBox objTag.TimeStamp
```

参见

变量 (页 244)

TimeStepBase

说明

指定影响表中显示的时间戳精度的时间单位。 时间因数与时间单位的乘积决定精度。

运行系统中的访问权限：读和写

语法

Object.**TimeStepBase**[=TimeStepBase]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- OnlineTableControl

TimeStepBase

可选项。 用于指定影响时间戳精度的时间单位的值或常数。

值	名称	说明
0	精确	指定在表格行中只显示时间戳完全相同的那些值。
1000	Base1s	指定时间范围是基数 1 s 的倍数。
100	Base100ms	指定时间范围是基数 100 ms 的倍数。
250	Base250ms	指定时间范围是基数 250 ms 的倍数。
500	Base500ms	指定时间范围是基数 500 ms 的倍数。

参见

OnlineTableControl (页 402)

TimeStepFactor

说明

指定使用时间单位“TimeStepBase”来确定的时间戳的精度。

通过时间因数与时间单位的乘积来确定精度。 例如，如果要显示在同一行中 3 秒内生成的所有值，则输入因数“3”和时间单位“Base1s”。

如果选择“精确”作为时间单位，则输入的因数无效。

运行系统中的访问权限：读和写

语法

Object.**TimeStepFactor**[=Int32]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- OnlineTableControl

Int32

可选项。 用于指定影响时间戳精度的时间因数的值或常数。

参见

OnlineTableControl (页 402)

TitleColor

说明

指定表格标题的背景色。

运行系统中可进行读写访问

语法

Object.**TitleColor**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

可选项。用于指定表格标题背景色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

TitleCut

描述

指定列宽过窄时是否缩减标题栏中的字段内容。

运行系统中的访问权限：读和写

语法

Object.**TitleCut**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选。如果因列宽过窄而缩短标题栏中的列标题，则为 TRUE。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

TitleDarkShadowColor

说明

为对象表格中的列和行标题指定 3D 底纹的暗边颜色。

也可将“TitleStyle (页 1243)”设置为“1”。

运行系统中可进行读写访问

语法

Object.**TitleDarkShadowColor**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

可选项。用于指定 3D 底纹暗边颜色的值或常量。

注释

可以使用“RGB”函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

TitleForeColor

说明

指定对象中表格列和行标题的文本颜色。

运行系统中的访问权限： 读和写

语法

Object.**TitleForeColor**[=Color]

Object

必需项。 具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

可选项。 用于指定表格列和行标题字体颜色的值或常量。

注释

使用"RGB"函数来定义 RGB 格式（红、绿、蓝）的颜色。分别为三个 RGB 值输入相应的十进制数值（范围从 0 到 255）。"红色"显示如下：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

TitleGridLineColor

说明

指定表格标题栏中分隔线的颜色。

运行系统中可进行读写访问

语法

Object.**TitleGridLineColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

可选项。用于指定表格标题栏中分隔线颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

TitleLightShadowColor

说明

指定对象表格列和行标题中 3D 底纹亮边的颜色。

也可将“TitleStyle (页 1243)”设置为“1”。

运行系统中可进行读写访问

语法

Object.**TitleLightShadowColor**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

Color

可选项。用于指定 3D 底纹亮边颜色的值或常量。

注释

可以使用“RGB”函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

TitleSort

描述

指定如何触发按列标题排序。

运行系统中可进行读写访问

语法

Object.**TitleSort**[=GridSortTrigger]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

GridSortTrigger

可选项。用于指定如何触发按列标题排序的值或常量。

值	说明
0	禁用按列标题排序。
1	通过单击列标题来触发排序。
2	通过双击列标题来触发排序。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

TitleStyle

描述

指定是否使用列标题文本的底纹颜色。

运行系统中可进行读写访问

语法

Object.**TitleStyle**[=GridHeaderStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

GridHeaderStyle

可选项。用于指定是否显示列标题文本的底纹的值或常量。

值	说明
0	不使用底纹颜色。无色标题样式。
1	启用底纹颜色。标题的 3D 表示。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

Toggle

描述

指定在运行系统中对所选对象操作完后是否仍占用该对象。

运行系统中的访问权限：读和写

语法

Object.**Toggle**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Button
- RoundButton*

*：只读访问权限

BOOLEAN

可选项。如果在运行系统中对所选对象操作完后仍占用该对象，则为 TRUE。

Tolerance

描述

指定自报告出现偏差起存储空间画面的极限。

运行系统中的访问权限：读取

语法

Object.**Tolerance**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- DiscSpaceView

Int32

可选项。用于指定自此起将报告出现偏差的磁盘空间视图限值的值或常量。

参见

DiskSpaceView (页 337)

ToleranceColor

描述

指定超过容差范围存储空间显示画面滚动条的显示颜色。

运行系统中可进行读访问

语法

Object.**ToleranceColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- DiscSpaceView

Color

可选项。用于指定超过容差范围存储空间画面滚动条显示颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

DiskSpaceView (页 337)

ToleranceLowerLimit

描述

设置容差 1 的下限。

运行系统中的访问权限：读和写

语法

Object.**ToleranceLowerLimit**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

DOUBLE

可选项。用于指定容差 1 下限的值或常量。

注释

通过属性“ToleranceLowerLimit”、“ToleranceLowerLimitColor”和“ToleranceLowerLimitRelative”可定义以下值：

- 限值
- 达到限值后的显示
- 估算的类型

参见

Bar (页 285)

ToleranceLowerLimitColor

描述

指定“ToleranceLowerLimit””下限的颜色。

若滚动条颜色在达到限值后立即更改，则“ToleranceLowerLimitEnabled”属性值必须为 TRUE。

运行系统中可进行读写访问

语法

Object.**ToleranceLowerLimitColor**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Bar

Color

可选项。用于指定 "ToleranceLowerLimit" 下限的颜色的值或常量。

注释

可以使用“RGB”函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Bar (页 285)

ToleranceLowerLimitEnabled

描述

指定是否监视“ToleranceLowerLimit”限值。 达到极限值时，显示极值，通过属性“ToleranceLowerLimit”、“ToleranceLowerLimitColor”和“ToleranceLowerLimitRelative”设置判断的类型。

运行系统中的访问权限：读和写

语法

Object.**ToleranceLowerLimitEnabled**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

BOOLEAN

可选。如果监视"ToleranceLowerLimit"限值，则为 TRUE。

参见

Bar (页 285)

ToleranceLowerLimitRelative

描述

指定"ToleranceLowerLimit"下限是以百分比还是绝对值进行估算。

运行系统中可进行读写访问

语法

Object.**ToleranceLowerLimitRelative**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar

BOOLEAN

可选项。

TRUE，将"ToleranceLowerLimit"下限以百分比形式估算。

FALSE，将"ToleranceLowerLimit"下限以绝对值形式估算。

参见

Bar (页 285)

ToleranceUpperLimit

描述

设置容差 1 的上限。

运行系统中的访问权限：读和写

语法

Object.**ToleranceUpperLimit**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

DOUBLE

可选项。用于指定容差 1 上限的值或常量。

注释

通过属性“ToleranceUpperLimit”、“ToleranceUpperLimitColor”和“ToleranceUpperLimitRelative”可指定以下值：

- 限值
- 达到限值后的显示
- 估算的类型

参见

Bar (页 285)

ToleranceUpperLimitColor

描述

指定“ToleranceUpperLimit”上限值的颜色。

运行系统中可进行读写访问

语法

Object.**ToleranceUpperLimitColor**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Bar

Color

可选项。用于指定"ToleranceUpperLimit""上限的颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Bar (页 285)

ToleranceUpperLimitEnabled

描述

指定是否监视"ToleranceUpperLimit""极限。

运行系统中的访问权限：读和写

语法

Object.**ToleranceUpperLimitEnabled**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

BOOLEAN

可选。若要监视"ToleranceUpperLimit"极限，则为 TRUE。

注释

通过属性“ToleranceUpperLimit",”、“ToleranceUpperLimitColor”和“ToleranceUpperLimitRelative”指定以下值：

- 限值
- 达到限值后的显示
- 估算的类型

参见

Bar (页 285)

ToleranceUpperLimitRelative

描述

指定“ToleranceUpperLimit""上限是以百分比还是绝对值形式进行估算。

运行系统中可进行读写访问

语法

Object.**ToleranceUpperLimitRelative**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Bar

BOOLEAN

可选项。

TRUE，将“ToleranceUpperLimit”上限以百分比形式估算。

FALSE，将“ToleranceUpperLimit”上限以绝对值形式估算。

参见

Bar (页 285)

ToolBar_ButtonsHeight

说明

在运行系统中无访问权限。

ToolBar_ButtonsWidth

说明

在运行系统中无访问权限。

ToolbarAlignment

说明

指定工具栏的位置。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**ToolbarAlignment**[=ToolbarPosition]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

在运行系统中您没有以下格式的访问权限：

- SysDiagControl

ToolbarPosition

可选项。用于指定对象中工具栏位置的值或常量。

值	名称
0	顶部
1	底部
2	左对齐
3	右对齐

ToolbarBackColor

描述

指定工具栏的背景色。

也可将“ToolbarUseBackColor (页 1275)”设置为“TRUE”。

运行系统中可进行读写访问

语法

Object.**ToolbarBackColor**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Color

可选项。用于指定工具栏背景色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarBackgroundColor

说明

在运行系统中无访问权限。

ToolbarButtonActive

描述

指定是否在运行系统中激活按钮的相应功能。在运行系统中单击按钮将触发相应的功能。

运行系统中可进行读写访问

语法

Object.**ToolbarButtonActive**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，激活分配给键的功能。

FALSE，不激活分配给键的功能。可以通过局部脚本将自定义功能与键相连。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarButtonAdd

说明

在对象的工具栏中创建新的用户自定义按钮。新创建的按钮会自动使用"ToolbarButtonIndex"而引用。

运行系统中可进行读写访问

语法

Object.**ToolbarButtonAdd**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

必需项。用于通过"ToolbarButtonName"指定工具栏中新按钮名称的值或常数。

参见

ToolbarButtonIndex (页 1263)

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarButtonAuthorization

描述

指定所选键功能的权限。权限在用户管理中进行组态。

运行系统中可进行读写访问

语法

Object.**ToolbarButtonAuthorization**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于指定所选键功能权限的值或常量。

参见

用户归档控件 (页 566)

TrendRulerControl (页 532)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

ToolbarButtonBeginGroup

描述

在工具栏上，为所选键功能插入前导分隔符（垂直线）。这些分隔符可用于对按钮功能的图标进行分组。

运行系统中可进行读写访问

语法

Object.**ToolbarButtonBeginGroup**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，在所选键功能前面插入分隔符。

FALSE，不在所选键功能前面插入分隔符。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarButtonClick

说明

单击工具栏按钮。可使用按钮的 ID 来调用相应按钮的功能。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**ToolbarButtonClick**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- SysDiagControl*
- TrendRulerControl
- UserArchiveControl

*：只读访问权限

Int32

可选项。用于指定所单击按钮的 ID 的值或常量。

ToolbarButtonCount

描述

指定工具栏中的已组态按钮数目。

运行系统中可进行读写访问

语法

Object.**ToolbarButtonCount**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl

- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于指定工具栏中可组态的按钮数的值或常量。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarButtonEnabled

描述

启用自定义工具栏按钮的操作。

运行系统中的访问权限：读和写

语法

Object.**ToolbarButtonEnabled**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选。如果启用了工具栏中选定的用户自定义键的操作，则为 TRUE。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarButtonHeight

说明

在运行系统中无访问权限。

ToolbarButtonHotKey

说明

指定所选对象按钮的热键。以 ASCII 编码的方式输入热键，例如，对于 <F1> 键，输入“112”。

也可将“ToolbarUseHotKeys (页 1276)”设置为“TRUE”。

运行系统中可进行读写访问

语法

Object.**ToolbarButtonHotKey**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- FunctionTrendControl

- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于指定所选按钮快捷方式的值或常量。

参见

TrendIndex (页 1299)

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarButtonID

说明

使用其 ID 引用按钮。要访问状态栏元素的属性，需要设置""ToolbarButtonID""。

按钮使用其 ID 进行引用不会依赖于按钮的实际顺序。ID 列于对象的巡视窗口中“属性 > 属性 > 工具栏 > 工具栏 - 按钮”(Properties > Properties > Toolbar > Toolbar - Buttons) 下。

运行系统中可进行读写访问

语法

Object.**ToolbarButtonID**[=Int32]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- FunctionTrendControl

- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于指定要使用 ID 编辑的工具栏按钮的值或常量。

参见

ToolbarButtonClick (页 1258)

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarButtonIndex

描述

引用按钮。要访问按钮的属性，需要设置""ToolbarButtonIndex""。

介于 0 至 (ToolbarButtonCount - 1) 之间的值为"ToolbarButtonIndex"的有效值。"ToolbarButtonCount"属性指定可组态按钮的数目。

运行系统中可进行读写访问

语法

Object.**ToolbarButtonIndex**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl

- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于通过索引指定要编辑的时间轴的值或常量。

参见

ToolbarButtonCount (页 1259)

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarButtonLocked

描述

指定是否显示使用"ToolbarButtonIndex"引用的用户自定义按钮的锁定、按下状态。

运行系统中可进行读写访问

语法

Object.**ToolbarButtonLocked**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl

- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

TRUE，工具栏中引用的用户自定义按钮显示为按下状态。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarButtonName

描述

指定使用“ToolbarButtonIndex”引用的用户自定义按钮的名称。

运行系统中可进行读写访问

语法

Object.**ToolbarButtonName**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl

- TrendRulerControl
- UserArchiveControl

STRING

可选项。用于指定引用的用户自定义按钮名称的值或常量。

参见

ToolbarButtonIndex (页 1263)

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarButtonRemove

描述

使用引用的用户自定义按钮的名称可移除该按钮。

运行系统中可进行读写访问

语法

Object.**ToolbarButtonRemove**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl

- TrendRulerControl
- UserArchiveControl

STRING

可选项。用于指定待移除的引用的用户自定义按钮名称的值或常量。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarButtonRename

说明

指定使用“ToolbarButtonIndex”引用的用户自定义按钮的新名称。

运行系统中可进行读写访问

语法

Object.**ToolbarButtonRename**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

可选项。用于指定所选用户自定义按钮新名称的值或常数。

参见

ToolbarButtonIndex (页 1263)

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarButtonRepos

说明

在对象的工具栏中指定使用"ToolbarButtonIndex"引用的按钮位置。

如果已使用"ToolbarButtonRepos"更改了按钮位置，"ToolbarButtonRepos"的值则会分配给"ToolbarButtonIndex"。

运行系统中可进行读写访问

语法

Object.**ToolbarButtonRepos**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Int32

可选项。用于指定工具栏中引用按钮的位置的值或常量。值范围为 0 到 (ToolbarButtonCount - 1)。超出该范围的值无效。

0：引用的按钮放置在左侧。

参见

ToolbarButtonCount (页 1259)

ToolbarButtonIndex (页 1263)

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarButtons

说明

在运行系统中无访问权限。

ToolbarButtonSettings

说明

在运行系统中无访问权限。

ToolbarButtonsForMigration

说明

在运行系统中无访问权限。

ToolbarButtonTooltipText

描述

指定工具栏中用户自定义按钮的工具提示文本。

运行系统中的访问权限：读和写

语法

Object.**ToolbarButtonTooltipText**[=STRING]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

STRING

可选项。用于指定所选用户定义按钮的工具提示文本的值或常量。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarButtonUserDefined

描述

指定项目工程师是否已将此工具栏按钮添加为新的用户自定义按钮。

运行系统中的访问权限：读和写

语法

Object.**ToolbarButtonUserDefined**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

Boolean

可选项。

TRUE， 由用户自定义工具栏按键。

FALSE， 由系统指定工具栏按键。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarButtonVisible

说明

指定是否在工具栏中显示使用“ToolbarButtonIndex”引用的按钮。

运行系统中可进行读写访问

语法

Object.**ToolbarButtonVisible**[=<BOOLEAN>]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，显示引用的按钮。

FALSE，隐藏引用的按钮。

参见

ToolbarButtonIndex (页 1263)

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarButtonWidth

说明

在运行系统中无访问权限。

ToolbarEnabled

说明

在运行系统中无访问权限。

ToolbarHeight

说明

在运行系统中无访问权限。

ToolbarIconStyle

说明

在运行系统中无访问权限。

ToolbarLeft

说明

在运行系统中无访问权限。

ToolbarShowTooltips

描述

指定是否在运行系统中显示按键功能的工具提示。可以使用名称 ToolbarShowTooltips 使该属性动态化。“ToolbarButtonTooltipText”属性用于指定工具提示文本。

运行系统中可进行读写访问

语法

Object.**ToolbarShowTooltips**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，显示工具提示。

FALSE，不显示工具提示。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarStyle

说明

在运行系统中无访问权限。

ToolbarTop

说明

在运行系统中无访问权限。

ToolbarUseBackColor

描述

指定是否显示工具栏的背景色。

运行系统中可进行读写访问

语法

Object.**ToolbarUseBackColor**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，显示工具栏的背景色。

FALSE，不显示工具栏的背景色。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarUseHotKeys

描述

指定是否激活工具栏中按钮的热键。可使用“ToolbarButtonHotKey (页 1261)”为每个按钮定义热键。

运行系统中可进行读写访问

语法

Object.**ToolbarUseHotKeys**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。TRUE，激活热键。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarVisible

描述

指定是否显示控件的工具栏。

运行系统中可进行读写访问

语法

Object.**ToolbarVisible**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- FunctionTrendControl
- OnlineTableControl
- OnlineTrendControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，显示工具栏。

FALSE，不显示工具栏。

参见

AlarmControl (页 255)

FunctionTrendControl (页 351)

OnlineTableControl (页 402)

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

用户归档控件 (页 566)

ToolbarWidth

说明

在运行系统中无访问权限。

ToolTipText

说明

指定工具提示文本。

运行系统中的访问权限：读取

语法

Object.**ToolTipText**[=STRING]

ToolTipText

必选。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- ComboBox
- Connector
- Ellipse
- EllipseSegment
- EllipticalArc

- GraphicIOField
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton *
- StatusForce *
- Switch *
- SymbolicIOField *
- TextField
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline
- WindowSlider *

* 只读访问权限

STRING

可选项。用于指定工具提示文本的值或常量。

Top

描述

指定 Y 坐标的值。

运行系统中的访问权限：读取

语法

Object.**Top**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- AlarmView
- Bar
- BatteryView
- Button
- ChannelDiagnose
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Clock
- ComboBox
- Connector
- DateTimeField
- DiscSpaceView
- Ellipse
- EllipseSegment
- EllipticalArc
- FunctionTrendControl
- Gauge
- GraphicIOField
- GraphicView
- HTMLBrowser
- IOField
- Line

- ListBox
- MediaPlayer
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- OptionGroup
- PLCCodeViewer
- Polygon
- Polyline
- ProDiagOverview
- ProtectedAreaNameView
- RangeLabelView
- RangeQualityView
- RecipeView *
- Rectangle
- RoundButton *
- S7GraphOverview
- ScreenWindow
- Slider **
- SmartClientView
- StatusForce *
- Switch *
- SymbolLibrary
- SymbolicIOField *
- SysDiagControl *
- TextField
- TrendRulerControl
- TrendView *
- TubeArcObject

- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline
- UserArchiveControl
- UserView **
- WLanQualityView
- WindowSlider *
- ZoneLabelView
- ZoneQualityView

* 只读访问权限

** RT Professional 只读访问权限

在运行系统中您没有以下格式的访问权限：

- ApplicationWindow

Int32

可选项。其中包含以像素为单位的 Y 坐标值的值或常量（从画面左上方测量）。

注释

Y 坐标指包围对象的矩形左上方的值。在运行系统中还监视画面的极值。如果指定的坐标值超出显示大小，则用户自定义函数会中断，并显示一条错误消息。

TopMargin

说明

在运行系统中无访问权限。

TopOffset

说明

指定大于画面窗口的画面显示的零点垂直位移。 此位移参考画面窗口的顶部边缘。

显示的画面为剪切画面。 画面滚动条位于画面左边缘和上边缘。

如果希望使用画面滚动条的水平和垂直偏移在画面窗口中显示画面，应对偏移使用“HorizontalScrollBarPosition (页 884)”和“VerticalScrollbarPosition (页 1415)”属性。

运行系统中的访问权限： 读和写

语法

Object.**TopOffset**[=Int32]

Object

必需项。 具有以下格式的“ScreenItem”类型的对象：

- Screenwindow

Int32

可选项。 用于指定画面显示的零点自画面窗口顶部边缘的垂直位移的值或常量。

参见

ScreenWindow (页 479)

Total

描述

指定存储器容量。

运行系统中的访问权限：读取

语法

Object.**Total**[=DOUBLE]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- DiskSpaceView

DOUBLE

可选项。用于指定存储器容量的值或常量。

TransitionHeaderFont

说明

指定此 PLC 代码显示的信息区域中的字体。

运行系统中的访问权限：读和写

语法

Object.**TransitionHeaderFont**[=Font]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- PLCCodeViewer

Font

可选项。 用于指定字体的值或常量。

参见

PLCCodeViewer (页 442)

Transparency

描述

指定对象透明度（百分数形式）。

0 = 不透明；100 = 完全透明（不可见）。

图形对象的文本和字段仅在“100”时透明。

在运行系统中，完全透明的对象（不可见）也很有用。

运行系统中的访问权限：读和写

语法

Object.**Transparency**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Clock
- ComboBox
- Connector
- Ellipse
- EllipseSegment
- EllipticalArc
- Gauge
- GraphicIOField
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton *
- Slider *
- SymbolicIOField *
- TextField

- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline
- WindowSlider *

* 只读访问权限

** RT Professional，只读访问权限

Int32

可选项。用于指定对象透明度（百分数形式）的值或常量。

TransparentColor

描述

指定将已分配图形（*.bmp 或 *dib）的哪种颜色设置为“透明”。

也可通过“UseTransparentColor”指定要显示为透明的颜色。

运行系统中的访问权限：读和写

语法

Object.**TransparentColor**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- GraphicIOField
- GraphicView

Color

可选项。用于指定要显示为透明的颜色的值或常量。

注释

可以使用“RGB”函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 vbRed 和 vbGreen。

TransparentColorDeactivatedPicture

描述

在“已禁用”状态下将已分配位图对象的颜色设置为“transparent”。

运行系统中的访问权限：读取

语法

Object.**TransparentColorDeactivatedPicture**[=Color]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- RoundButton

Color

可选项。用于指定在“已禁用”状态下将已分配位图对象的何种颜色设置为“transparent”的值或常量。

注释

可以使用“RGB”函数来定义 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：RGB(255, 0, 0)。还可以使用 VBS 颜色常量，如 vbRed 和 vbGreen。

“PicDeactUseTransColor”属性的值必须为 TRUE 才能将颜色设置为“transparent”。

参见

RoundButton (页 471)

TransparentColorPictureOff

描述

指定在“关闭”状态下将已分配位图对象的何种颜色设置为“transparent”。

运行系统中可进行读写访问

语法

Object.**TransparentColorPictureOff**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Button
- RoundButton*

*：只读访问权限

Color

可选项。用于指定在"关闭"状态下将已分配位图对象的何种颜色设置为"transparent"的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

TransparentColorPictureOn

描述

指定在"打开"状态下将已分配位图对象的何种颜色设置为"transparent"。

"PicDownUseTransColor"属性的值必须为 TRUE 才能将颜色设置为"transparent"。

运行系统中可进行读写访问

语法

Object.**TransparentColorPictureOn**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Button
- RoundButton*

*：只读访问权限

Color

可选项。用于指定在“打开”状态下将已分配位图对象的何种颜色设置为“transparent”的值或常量。

注释

可以使用“RGB”函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

TrendActualize

描述

指定是否更新所选趋势。

运行系统中可进行读写访问

语法

Object.**TrendActualize**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- FunctionTrendControl

BOOLEAN

可选项。

TRUE，实时更新所选的趋势。

FALSE，不更新所选的趋势。在将记录的趋势与当前趋势进行比较时，该设置很有用。

参见

FunctionTrendControl (页 351)

TrendAdd

说明

创建新趋势。会自动使用"TrendIndex"引用新创建的趋势。

运行系统中可进行的访问：读和写

语法

Object.**TrendAdd**[=STRING]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

STRING

必需项。用于通过"TrendName"指定新趋势名称的值或常量。

TrendAutoRangeBeginTagName

说明

指定定义趋势的数据范围起始值的变量。 还可使用 "TrendAutoRangeSource (页 1293)"属性指定动态确定趋势的数据范围。

运行系统中的访问权限：读和写

语法

Object.**TrendAutoRangeBeginTagName**[=STRING]

Object

必需项。 "ScreenItem"对象，且具有以下特性：

- OnlineTrendControl

STRING

可选项。 用于指定起始值的变量的值或常量。

参见

OnlineTrendControl (页 418)

TrendAutoRangeBeginValue

说明

指定趋势的数据范围的起始值。 还可使用 “TrendAutoRangeSource (页 1293)”属性指定静态设置趋势的数据范围。

运行系统中的访问权限：读和写

语法

Object.**TrendAutoRangeBeginValue**[=DOUBLE]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- OnlineTrendControl

DOUBLE

可选项。 用于指定趋势的数据范围起始值的值或常量。

参见

OnlineTrendControl (页 418)

TrendAutoRangeEndTagName

说明

指定定义趋势的数据范围结束值的变量。 还可使用 “TrendAutoRangeSource (页 1293)”属性指定动态确定趋势的数据范围。

运行系统中的访问权限：读和写

语法

Object.**TrendAutoRangeEndTagName**[=STRING]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- OnlineTrendControl

STRING

可选项。 用于指定结束值的变量的值或常量。

参见

OnlineTrendControl (页 418)

TrendAutoRangeEndValue

说明

指定趋势的数据范围的结束值。 还可使用 “TrendAutoRangeSource (页 1293)”属性指定静态设置趋势的数据范围。

运行系统中的访问权限：读和写

语法

Object.**TrendAutoRangeEndValue**[=DOUBLE]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- OnlineTrendControl

DOUBLE

可选项。 用于指定趋势的数据范围结束值的值或常量。

参见

OnlineTrendControl (页 418)

TrendAutoRangeSource

说明

指定确定趋势数据的自动数据范围的方法。

运行系统中的访问权限：读和写

语法

Object.**TrendAutoRangeSource**[=AutorangeSourceType]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

AutorangeSourceType

可选项。用于指定数据范围类型的值或常量。

有下列设置可用：

值	名称	说明
x	自动调整	指定将显示的值范围自动调整到当前值。
1	动态范围	指定根据已连接在线变量的值构成显示的值范围的下限和上限。下限和上限的变量映射到"TrendAutoRangeBeginTagName (页 1290)"和"TrendAutoRangeEndTagName (页 1291)"属性中。
2	静态范围	指定通过值范围的下限和上限组态来定义显示的值范围。下限和上限的值映射到"TrendAutoRangeBeginValue (页 1291)"和"TrendAutoRangeEndValue (页 1292)"属性中。

TrendBeginTime

描述

定义用于将数据传送到所选趋势的时间范围的开始时间。

运行系统中的访问权限：读和写

语法

Object.**TrendBeginTime**[=DateTime]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

DateTime

可选项。用于定义所选趋势的数据供应的起始时间。

参见

FunctionTrendControl (页 351)

TrendColor

说明

指定趋势视图中所引用趋势的趋势线的颜色。

运行系统中的访问权限：读和写

语法

Object.**TrendColor**[=Color]

Object

必需项。 具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

Color

可选项。 用于指定对象中所引用趋势的趋势线颜色的值或常量。

注释

使用"RGB"函数来定义 RGB 格式（红、绿、蓝）的颜色。分别为三个 RGB 值输入相应的十进制数值（范围从 0 到 255）。"红色"显示如下：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 vbRed 和 vbGreen。

参见

TrendIndex (页 1299)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendCount

描述

指定所组态趋势的数目。

运行系统中可进行读写访问

语法

Object.**TrendCount**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

Int32

可选项。用于指定所组态趋势数目的值或常量。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendEndTime

描述

定义用于所选趋势数据连接的时间范围的结束时间。

运行系统中的访问权限：读和写

语法

Object.**TrendEndTime**[=DateTime]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

DateTime

可选项。用于指定所选趋势的数据供应的结束时间。

参见

FunctionTrendControl (页 351)

TrendExtendedColorSet

描述

指定是否显示趋势的点和填充颜色。

运行系统中可进行读写访问

语法

Object.**TrendExtendedColorSet**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

可选项。

TRUE，"TrendFillColor (页 1298)"和"TrendPointColor (页 1307)"中的设置有效。

FALSE，点和填充颜色的设置无效。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendFill

描述

指定趋势下方的区域是否显示为已填充。

如果“TrendExtendedColorSet (页 1296)”设置为“FALSE”，则趋势颜色也可作为填充颜色使用。

运行系统中可进行读写访问

语法

Object.**TrendFill**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

可选项。

TRUE，趋势下方的区域以填充方式显示。

FALSE，趋势不以填充方式显示。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendFillColor

说明

定义趋势的填充颜色。

如果属性“TrendFill (页 1297)”和“TrendExtendedColorSet (页 1296)”也设置为“TRUE"，则该设置有效。

运行系统中可进行读写访问

语法

Object.**TrendFillColor**[=Color]

Object

必选。具有以下格式的“ScreenItem”类型的对象：

- FunctionTrendControl
- OnlineTrendControl

Color

可选。用于指定所选趋势填充颜色的值或常量。

注释

可以使用“RGB”函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendIndex

描述

引用一个已组态的趋势。要访问一个趋势的属性，需要设置""TrendIndex""。

介于 0 至 (TrendCount - 1) 之间的值为"TrendIndex"的有效值。"TrendCount"属性指定所组态趋势的数目。

运行系统中可进行读写访问

语法

Object.**TrendIndex**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

Int32

可选项。用于通过索引指定要编辑的趋势的值或常量。

参见

TrendCount (页 1295)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendIndicatorColor

描述

指定趋势视图的颜色。趋势指示器用小箭头表示要监视的测量值的趋势（上升或下降）。要激活趋势指示器，"ShowTrendIndicator"属性值必须为"TRUE"。

运行系统中可进行读写访问

语法

Object.**TrendIndicatorColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar

Color

可选项。用于指定趋势指示器颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Bar (页 285)

TrendLabel

说明

为使用"TrendIndex"引用的趋势定义标签文本。

运行系统中的访问权限：读和写

语法

Object.**TrendLabel**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

STRING

可选项。用于指定引用的趋势标签文本的值或常量。

参见

TrendIndex (页 1299)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendLineStyle

示例

指定用于趋势显示的线类型。

运行系统中可进行读写访问

语法

Object.**TrendLineStyle**[=LineStyle]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

LineStyle

可选项。用于指定用来显示趋势的线类型的值或常量。

值	说明
0	趋势显示为实线。
1	趋势显示为虚线。
2	趋势显示为点线。
3	趋势显示为点划线。
4	趋势显示为双点划线。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendLineType

描述

指定如何显示趋势。

运行系统中可进行读写访问

语法

Object.**TrendLineType**[=TrendLineTypeScada]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

TrendLineTypeScada

可选项。用于指定如何显示趋势的值或常量。

值	说明
0	仅显示趋势点。
1	以线性互连各个点的方式显示趋势。
2	显示步进趋势及其互连点。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendLineWidth

描述

指定所选趋势的线宽（以像素为单位）。

运行系统中的访问权限：读和写

语法

Object.**TrendLineWidth**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

Int32

可选项。用于指定以像素为单位的所选趋势线宽的值或常量。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendLowerLimit

说明

指定在对象中作为趋势显示的变量下限。如果变量低于"TrendLowerLimit"的值，则使用在"TrendLowerLimitColor (页 1304)"中设置的颜色标记受影响的趋势点。

使用"TrendLowerLimitColoring (页 1305)"属性另外指定激活超出下限的显示。

运行系统中可进行读写访问

语法

Object.**TrendLowerLimit**[=DOUBLE]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

DOUBLE

可选项。用于指定趋势视图中显示的变量下限的值或常量。

参见

TrendIndex (页 1299)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendLowerLimitColor

说明

指定用于标记低于"TrendLowerLimit (页 1303)"值的趋势值的颜色。

如果"TrendLowerLimitColoring (页 1305)"属性的值为"TRUE""，则此设置有效。

运行系统中的访问权限：读和写

语法

Object.**TrendLowerLimitColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

Color

可选项。用于指定低于"TrendLowerLimit (页 1303)"值的趋势值颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

TrendLowerLimitColoring

描述

指定是否使用"TrendLowerLimitColor"标识小于"TrendLowerLimit"值的变量值。

运行系统中的访问权限：读和写

语法

Object.**TrendLowerLimitColoring**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

可选项。

如果 "TrendLowerLimitColor"属性有效，则为 TRUE。

如果 "TrendLowerLimitColor"属性无效，则为 FALSE。

TrendMeasurePoints

描述

指定用于显示选定趋势的测量点的数量。

定义从用户归档向趋势提供的值对的数量。

运行系统中的访问权限：读和写

语法

Object.**TrendMeasurePoints**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

Int32

可选项。用于指定所选趋势测量点或值对数的值或常量。

参见

FunctionTrendControl (页 351)

TrendName

描述

指定使用"TrendIndex"引用的趋势的名称。

运行系统中可进行读写访问

语法

Object.**TrendName**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

STRING

可选项。用于指定引用的趋势名称的值或常量。

参见

TrendIndex (页 1299)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendPointColor

说明

指定所引用趋势上点的颜色。

如果“TrendExtendedColorSet (页 1296)”属性的值为“TRUE""，则此设置有效。

运行系统中可进行读写访问

语法

Object.**TrendPointColor**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- FunctionTrendControl
- OnlineTrendControl

Color

可选项。用于指定所引用趋势的点颜色的值或常量。

注释

可以使用“RGB”函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

TrendIndex (页 1299)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendPointStyle

描述

指定如何显示趋势点。

运行系统中可进行读写访问

语法

Object.**TrendPointStyle**[=PointStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

PointStyle

可选项。用于指定趋势中点显示方式的值或常量。

值	说明
0	不显示点。设置点宽度禁用。
1	趋势点以一个像素大小显示。设置点宽度禁用。
2	点显示为正方形。设置点宽度启用。
3	点显示为圆形。设置点宽度启用。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendPointWidth

描述

指定点宽度（以像素为单位）。 只能为“正方形”和“圆形”点定义点宽度。

运行系统中的访问权限：读和写

语法

Object.**TrendPointWidth**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- FunctionTrendControl
- OnlineTrendControl

Int32

可选项。用于指定以像素为单位的所选趋势点宽的值或常量。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendProvider

描述

指定所选趋势的数据源。

运行系统中可进行读写访问

语法

Object.**TrendProvider**[=DataSourceType]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

DataSourceType

可选项。用于指定所选趋势数据源的值或常量。

值	描述	说明
0	无	没有数据源组态为可在运行系统中通过用户自定义函数执行。
1	记录变量	具有过程值归档的归档变量的数据源。
2	HMI 变量	具有 HMI 变量值的数据
3	配方数据	具有配方列的数据

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendProviderCLSID

说明

指定趋势中数据源的 CLSID。

运行系统中可进行读写访问

语法

Object.**TrendProviderCLSID**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

STRING

可选项。用于指定数据源 CLSID 的数值或常量。

值	说明
	没有可在运行系统中通过脚本建立连接的已组态数据源。
{416A09D2-8B5A-11D2-8B81-006097A45D48}	具有过程值归档的归档变量的数据源。
{A3F69593-8AB0-11D2-A440-00A0C9DBB64E}	具有来自变量管理的 在线变量的数据源。
{2DC9B1C8-4FC1-41B1-B354-3E469A13FBFD}	具有用户日志列的数据源。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendRangeType

描述

指定为所选趋势提供数据的时间范围。

只有选择用户归档作为数据源时，才可以定义测量点的数目。

运行系统中可进行读写访问

语法

Object.**TrendRangeType**[=TimeRangeMode]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- FunctionTrendControl

TimeRangeMode

可选项。用于指定为所选趋势提供数据的时间范围的值或常量。

值	描述	说明
0	时间范围	定义数据连接的开始时间和时间范围。
1	开始到结束的时间	定义数据连接的开始时间和结束时间。
2	测量点数量	定义数据连接的开始时间和测量点的数量。

参见

FunctionTrendControl (页 351)

TrendRemove

描述

使用引用的趋势名称可移除该趋势。

运行系统中可进行读写访问

语法

Object.**TrendRemove**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

STRING

可选项。用于指定待移除的引用的趋势名称的值或常量。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendRename

描述

指定使用“TrendIndex”引用的趋势的新名称。

运行系统中可进行读写访问

语法

Object.**TrendRename**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- FunctionTrendControl
- OnlineTrendControl

STRING

可选项。用于指定所选趋势新名称的值或常量。

参见

TrendIndex (页 1299)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendRepos

说明

在对象的趋势窗口中指定使用“TrendIndex”引用的趋势位置。

如果已使用“TrendRepos”更改了元素位置，“TrendRepos”的值则会分配给“TrendIndex”。

运行系统中可进行读写访问

语法

Object.**TrendRepos**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

Int32

可选项。用于指定趋势窗口中引用的趋势位置的值或常量。值范围为 0 到 (TrendCount - 1)。超出该范围的值无效。

0：引用的趋势放置在前方。

参见

TrendIndex (页 1299)

TrendCount (页 1295)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

Trends

说明

在运行系统中无访问权限。

TrendSelectTagName

说明

指定在运行系统中最初是否显示相应对话框以用来选择数值轴数据源的变量名称。

运行系统中的访问权限：读和写

语法

Object.**TrendSelectTagName**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

BOOLEAN

可选项。

TRUE，以在画面中显示相应对话框，从而可选择数值轴数据源的变量名称。

FALSE, ，在画面中不显示相应对话框，从而无法选择数值轴数据源的变量名称。

TrendSelectTagNameX

说明

指定用于选择 x 轴数据源的变量名称的对话框在运行系统中初次显示。

运行系统中可进行读写访问

语法

Object.**TrendSelectTagNameX**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

BOOLEAN

可选项。TRUE，选择 x 轴数据源的变量名称的对话框在屏幕中显示。

参见

TrendIndex (页 1299)

FunctionTrendControl (页 351)

TrendSelectTagNameY

说明

指定用于选择 y 轴数据源的变量名称的对话框在运行系统中初次显示。

运行系统中可进行读写访问

语法

Object.**TrendSelectTagNameY**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

BOOLEAN

可选项。TRUE，选择 Y 轴数据源的变量名称的对话框在屏幕中显示。

参见

TrendIndex (页 1299)

FunctionTrendControl (页 351)

TrendsForPrinting

说明

在运行系统中无访问权限。

TrendTagName

说明

指定为数值轴提供数据的变量的名称。

运行系统中的访问权限：读和写

语法

Object.**TrendTagName**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

STRING

可选项。用于指定数值轴的变量名称的数值或常量。

TrendTagNameX

描述

指定 X 轴连接的 HMI 变量或列名称。 通过选择按钮选择 HMI 变量或列。

运行系统中的访问权限：读和写

语法

Object.**TrendTagNameX**[=STRING]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- FunctionTrendControl

STRING

可选项。用于指定 X 轴的 HMI 变量或列名称的值或常量。

参见

FunctionTrendControl (页 351)

TrendTagNameY

描述

指定 Y 轴连接的 HMI 变量或列名称。通过选择按钮选择 HMI 变量或列。

运行系统中的访问权限：读和写

语法

Object.**TrendTagNameY**[=STRING]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- FunctionTrendControl

STRING

可选项。用于指定 Y 轴的 HMI 变量或列名称的值或常量。

参见

FunctionTrendControl (页 351)

TrendTimeAxis

说明

指定要用于所选趋势的时间轴。

运行系统中可进行读写访问

语法

Object.**TrendTimeAxis**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

STRING

可选项。用于指定所选趋势的时间轴的值或常量。

参见

TimeAxisCount (页 1198)

OnlineTrendControl (页 418)

TrendTimeRangeBase

描述

指定用于计算时间范围的时间单位。

运行系统中可进行读写访问

语法

Object.**TrendTimeRangeBase**[=TagLoggingTimeUnit]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

TagLoggingTimeUnit

可选项。用于指定确定时间范围的时间单位的值或常量。

值	描述
500	500 ms
1000	1 秒
60000	1 分钟
3600000	1 小时
86400000	1 天

参见

FunctionTrendControl (页 351)

TrendTimeRangeFactor

说明

指定确定为其显示引用趋势的时间间隔的因数。

通过将因数与时间单位相乘来计算时间范围。

如果"TrendRangeType (页 1311)"属性的值为"0"，则此设置有效。

运行系统中的访问权限：读和写

语法

Object.**TrendTimeRangeFactor**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

Int32

可选项。用于指定确定所引用趋势时间范围因数的值或常量。

TrendTrendWindow

描述

定义用于显示所选趋势的趋势窗口。

在“趋势窗口”选项卡中定义可用的趋势窗口。

运行系统中的访问权限：读和写

语法

Object.**TrendTrendWindow**[=STRING]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- FunctionTrendControl
- OnlineTrendControl

STRING

可选项。用于指定所选趋势的趋势窗口名称的值或常量。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendUncertainColor

描述

指定不确定状态的值的颜色。

运行系统中可进行读写访问

语法

Object.**TrendUncertainColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

Color

可选项。用于指定所选趋势中不确定状态值的颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendUncertainColoring

描述

指定不确定状态的值将突出显示。

运行系统中可进行读写访问

语法

Object.**TrendUncertainColoring**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

可选项。

TRUE，突出显示不确定状态的值。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendUpperLimit

说明

指定在特定对象中作为趋势显示的变量上限。如果变量超出"TrendUpperLimit"值，则使用"TrendUpperLimitColor (页 1323)"指定的颜色标记相关趋势点。

也可使用超出限制显示激活的 "TrendUpperLimitColoring (页 1324)"来指定颜色。

运行系统中可进行读写访问

语法

Object.**TrendUpperLimit**[=DOUBLE]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

DOUBLE

可选。用于指定所选趋势中上限值的值或常量。

参见

TrendIndex (页 1299)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendUpperLimitColor

说明

指定用于标记高于“TrendUpperLimit (页 1322)”值的趋势值的颜色。

如果“TrendUpperLimitColoring (页 1324)”属性的值为“TRUE””，则该设置有效。

运行系统中可进行读写访问

语法

Object.**TrendUpperLimitColor**[=Color]

Object

必选。具有以下格式的“ScreenItem”类型的对象：

- FunctionTrendControl
- OnlineTrendControl

Color

可选。用于指定高于“TrendUpperLimit (页 1322)”值的趋势值颜色的值或常量。

注释

可以使用“RGB”函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如，“红色”可按如下方式表示：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

TrendIndex (页 1299)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendUpperLimitColoring

描述

指定是否使用系统定义的颜色显示选择框架。

运行系统中可进行读写访问

语法

Object.**TrendUpperLimitColoring**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

可选项。

TRUE，启用"TrendUpperLimitColor"属性。

FALSE，禁用"TrendUpperLimitColor"属性。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendValueAlign

说明

在运行系统中无访问权限。

TrendValueAxis

说明

指定用于所选趋势的数值轴。

运行系统中可进行读写访问

语法

Object.**TrendValueAxis**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

STRING

可选项。用于指定所选趋势的数值轴的值或常量。

参见

ValueAxisCount (页 1384)

OnlineTrendControl (页 418)

TrendValueUnit

说明

指定“显示值”趋势类型的值的单位。 此单位取决于要显示的值，例如"%"或"°C"。

运行系统中的访问权限：读和写

语法

Object.**TrendValueUnit**[=STRING]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- OnlineTrendControl

STRING

可选项。 用于指定数值轴值单位的值或常量。

参见

OnlineTrendControl (页 418)

TrendVisible

说明

指定是否在对象中显示使用“TrendIndex”引用的趋势。

运行系统中可进行读写访问

语法

Object.**TrendVisible**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

可选项。

TRUE，显示引用的趋势。

FALSE，不显示引用的趋势。

参见

TrendIndex (页 1299)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendWindowAdd

描述

创建新的趋势视图。新创建的趋势视图会自动使用“TrendWindowIndex”而引用。

运行系统中可进行读写访问

语法

Object.**TrendWindowAdd**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- FunctionTrendControl
- OnlineTrendControl

STRING

必需项。用于通过“TrendWindowName”指定新趋势视图名称的值或常量。

参见

TrendWindowIndex (页 1333)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendWindowCoarseGrid

描述

指定是否显示主刻度的网格线。

运行系统中可进行读写访问

语法

Object.**TrendWindowCoarseGrid**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

可选项。

TRUE，显示主刻度的网格线。

FALSE，不显示主刻度的网格线。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendWindowCoarseGridColor

说明

指定对象中所引用图表的主网格颜色。

运行系统中的访问权限：读和写

语法

Object.**TrendWindowCoarseGridColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

Color

可选项。用于指定所引用图表的主网格颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

TrendWindowIndex (页 1333)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendWindowCount

说明

指定趋势视图中所组态的趋势图。

运行系统中可进行读写访问

语法

Object.**TrendWindowCount**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

Int32

可选项。用于指定趋势视图中所组态的趋势图数目的值或常量。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendWindowFineGrid

描述

指定是否对次刻度显示网格线。

运行系统中可进行读写访问

语法

Object.**TrendWindowFineGrid**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

可选项。

TRUE，显示辅助刻度的网格线。

FALSE，不显示辅助刻度的网格线。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendWindowFineGridColor

说明

指定对象中引用图的辅助网格的颜色。

运行系统中可进行读写访问

语法

Object.**TrendWindowFineGridColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

Color

可选项。用于指定所引用图表的帮助网格颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

TrendWindowIndex (页 1333)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendWindowForegroundTrendGrid

描述

指定是否在所选趋势窗口中仅显示前景趋势的网格线。

运行系统中可进行读写访问

语法

Object.**TrendWindowForegroundTrendGrid**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

可选项。

TRUE，趋势窗口中显示前景趋势的网格线。

FALSE，趋势窗口中显示所有趋势的网格线。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendWindowGridInTrendColor

描述

指定是否使用趋势颜色显示主刻度的网格线。

运行系统中的访问权限：读和写

语法

Object.**TrendWindowGridInTrendColor**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

可选项。

如果以趋势颜色显示网格线，则为 TRUE。

如果以 “TrendWindowCoarseGridColor (页 1328)”中所设置的颜色显示网格线，则为 FALSE。

参见

FunctionTrendControl (页 351)

TrendWindowHorizontalGrid

描述

指定是否显示水平网格线。

运行系统中的访问权限：读和写

语法

Object.**TrendWindowHorizontalGrid**[=BOOLEAN]

Object

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

TRUE：显示水平网格线。

FALSE：未显示水平网格线。

参见

FunctionTrendControl (页 351)

TrendWindowIndex

描述

引用一个趋势视图。要访问一个趋势视图的属性，需要设置""TrendWindowIndex""。

介于 0 至 (TrendWindowCount - 1) 之间的值为"TrendWindowIndex"的有效值。
"TrendWindowCount"属性指定所组态趋势视图的数目。

运行系统中可进行读写访问

语法

Object.**TrendWindowIndex**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

Int32

可选项。用于通过索引指定要编辑的趋势视图的值或常量。

参见

TrendWindowCount (页 1329)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendWindowName

描述

指定使用"TrendWindowIndex"引用的趋势视图的名称。

运行系统中可进行读写访问

语法

Object.**TrendWindowName**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

STRING

可选项。用于指定引用的趋势视图名称的值或常量。

参见

TrendWindowIndex (页 1333)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendWindowRemove

描述

使用趋势视图名称移除引用的趋势视图。

运行系统中可进行读写访问

语法

Object.**TrendWindowRemove**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

STRING

可选项。用于指定待移除的引用的趋势视图名称的值或常量。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendWindowRename

描述

指定使用"TrendWindowIndex"引用的趋势视图的新名称。

运行系统中可进行读写访问

语法

Object.**TrendWindowRename**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

STRING

可选项。用于指定所选趋势视图新名称的值或常量。

参见

TrendWindowIndex (页 1333)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendWindowRepos

说明

指定使用"TrendWindowIndex"引用的趋势视图的位置。

如果已使用"TrendWindowRepos"更改了趋势视图位置，"TrendWindowRepos"的值则会分配给"TrendWindowIndex"。

运行系统中可进行读写访问

语法

Object.**TrendWindwRepos**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

Int32

可选项。用于指定引用的趋势视图位置的值或常量。值范围为 0 到 (TrendWindowCount - 1)。超出该范围的值无效。

0：引用的趋势视图放置在前方。

参见

TrendWindowIndex (页 1333)

TrendWindowCount (页 1329)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendWindowRulerColor

说明

指定标尺的颜色。

如果“TrendWindowRulerStyle (页 1339)”属性的值为“1”，则此设置有效。

运行系统中的访问权限：读和写

语法

Object.**TrendWindowRulerColor**[=Color]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- FunctionTrendControl
- OnlineTrendControl

Color

可选项。用于指定标尺颜色的值或常量。

注释

可以使用“RGB”函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如：“红色”表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendWindowRulerLayer

描述

指定趋势视图中标尺的显示层。

运行系统中可进行读写访问

语法

Object.**TrendWindowRulerLayer**[=RulerLayer]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

RulerLayer

可选项。用于指定趋势视图中标尺显示层的值或常量。

值	说明
0	标尺显示在网格下的层上。
1	标尺位于趋势之上、网格下。
2	标尺位于趋势之上。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendWindowRulerStyle

描述

指定标尺的外观。

运行系统中可进行读写访问

语法

Object.**TrendWindowRulerStyle**[=RulerStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

RulerStyle

可选项。用于指定标尺显示方式的值或常量。

值	说明
0	标尺显示为单条黑线。
1	以组态的颜色和宽度显示标尺。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendWindowRulerWidth

说明

指定标尺的宽度（以像素为单位）。

如果"TrendWindowRulerStyle (页 1339)"属性的值为"1"，则此设置有效。

运行系统中的访问权限：读和写

语法

Object.**TrendWindowRulerWidth**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

Int32

可选项。用于指定标尺宽度（像素）的值或常量。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendWindows

说明

在运行系统中无访问权限。

TrendWindowSpacePortion

说明

指定对象图表区域中所引用图表的范围大小。

运行系统中的访问权限：读和写

语法

Object.**TrendWindowSpacePortion**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

Int32

可选项。用于指定对象图表区域中所引用图表范围大小的值或常量。

注释

从范围组成部分的总数，可以计算出每个范围的大小。例如，您组态了总共 3 个图表，如果每个图表的范围大小的值为"1"，则三个图表将具有相等大小。若要增加范围组成部分之间的相对大小，可增加一个或多个图表的范围大小。

参见

TrendWindowIndex (页 1333)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendWindowStatisticRulerColor

说明

指定统计数据标尺的颜色。 还可使用 "TrendWindowStatisticRulerStyle (页 1342)"属性指定以图形方式组态统计数据标尺。

运行系统中的访问权限：读和写

语法

Object.**TrendWindowStatisticRulerColor**[=Color]

Object

必需项。 "ScreenItem"对象，且具有以下特性：

- OnlineTrendControl

Color

可选项。 用于指定统计数据标尺颜色的值或常量。

注释

使用"RGB"函数来定义 RGB 格式（红、绿、蓝）的颜色。 为此，可分别为三个 RGB 值指定相应的十进制值（值范围为 0 至 255）。 "红色"显示如下：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

OnlineTrendControl (页 418)

TrendWindowStatisticRulerStyle

说明

定义用于指定趋势窗口中统计数据区域的线条的图形化组态。

运行系统中可进行读写访问

语法

Object.**TrendWindowStatisticRulerStyle**[=RulerStyle]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

RulerStyle

可选项。用于指定是否可以图形方式组态统计数据标尺的值或常量。

有下列设置可用：

值	名称	说明
0	简单	标尺显示为单条黑线。
1	图形	以组态的颜色和线宽显示标尺。

参见

OnlineTrendControl (页 418)

TrendWindowStatisticRulerWidth

说明

指定统计数据标尺的线宽。 还可使用 “TrendWindowStatisticRulerStyle (页 1342)”属性指定以图形方式组态统计数据标尺。

运行系统中的访问权限：读和写

语法

Object.**TrendWindowStatisticRulerWidth**[=Int32]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- OnlineTrendControl

Int32

可选项。 用于指定统计数据标尺线宽的值或常量。

参见

OnlineTrendControl (页 418)

TrendWindowVerticalGrid

描述

指定是否显示垂直网格线。

运行系统中可进行读写访问

语法

Object.**TrendWindowVerticalGrid**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

可选项。

TRUE，显示垂直网格线。

FALSE，不显示垂直网格线。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendWindowVisible

说明

指定是否显示使用"TrendWindowIndex"引用的趋势视图。

运行系统中的访问权限：读和写

语法

Object.**TrendWindowVisible**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

可选项。

如果显示引用的趋势视图，则为 TRUE。

如果未显示引用的趋势视图，则为 FALSE。

参见

TrendWindowIndex (页 1333)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendXAxis

说明

定义要用于所引用趋势的 X 轴。

运行系统中的访问权限： 读和写

语法

Object.**TrendXAxis**[=STRING]

Object

必需项。 具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

STRING

可选项。 用于指定用于所引用趋势的 X 轴名称的值或常量。

参见

TrendIndex (页 1299)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

TrendYAxis

说明

指定哪些 Y 轴将用于所引用趋势。

运行系统中的访问权限： 读和写

语法

Object.**TrendYAxis**[=STRING]

Object

必需项。 具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

STRING

可选项。 用于指定用于所引用趋势的 Y 轴名称的值或常量。

参见

TrendIndex (页 1299)

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

1.5.5.14 属性 U-W

Unit

描述

指定测量单位。

运行系统中的访问权限：读和写

语法

Object.**Unit**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- IOField

STRING

可选项。用于指定测量单位的值或常量。

参见

Bar (页 285)

IOField (页 382)

UnitColor

描述

指定测量单位的文本颜色。

运行系统中可进行读写访问

语法

Object.**UnitColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Gauge

Color

可选项。用于指定测量单位文本颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Gauge (页 366)

UnitFont

说明

指定测量单位的字体。

运行系统中的访问权限：读和写

语法

Object.**UnitFont**[=Font]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- Gauge

Font

可选项。 用于指定字体的值或常量。

参见

Gauge (页 366)

UnitText

描述

指定测量单位文本。

运行系统中可进行读写访问

语法

Object.**UnitText**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Gauge

STRING

可选项。用于指定测量单位文本的值或常量。

注释

输入要显示的文本，例如：所显示值的物理单位。

参见

Gauge (页 366)

UnitTop

描述

指定测量单位距对象上端的距离。文字的位置只能与刻度盘垂直方向的直径对齐。属性值是指对象的高度，可以测量从对象的上端到文字下端的距离。

运行系统中可进行读写访问

语法

Object.**UnitTop**[=DOUBLE]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Gauge

DOUBLE

可选项。指定测量单位距对象上端距离的值或常量。

值	说明
0	文字的下端与所选对象的上端对齐。文本位于所选对象范围之外时，文本将不可见。
1	文字的下端与所选对象的下端对齐。

参见

Gauge (页 366)

UnitViewColumnOrder

说明

在运行系统中无访问权限。

UpdateButtonVisible

说明

在运行系统中无访问权限。

UpperLimit

描述

指定输入值的上限。

运行系统中的访问权限：读和写

语法

Object.**UpperLimit**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- IOField

DOUBLE

可选项。用于指定输入值上限的值或常量。

参见

IOField (页 382)

UseAutoScaling

说明

在运行系统中无访问权限。

UseButtonFirstGradient

说明

在运行系统中无访问权限。

UseButtonSecondGradient

说明

在运行系统中无访问权限。

UseColumnBackColor

说明

指定列背景颜色的设置。

运行系统中可进行读写访问

语法

Object.**UseColumnBackColor**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

BOOLEAN

可选项。

TRUE，使用 "TableColor (页 1171)"中组态的背景色。

FALSE，使用表格视图列组态的背景色。

参见

OnlineTableControl (页 402)

UseColumnForeColor

说明

指定列字体颜色的设置。

运行系统中可进行读写访问

语法

Object.**UseColumnForeColor**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

BOOLEAN

可选项。

TRUE，使用由“TableForeColor (页 1173)”指定的字体颜色。

FALSE,，使用表格视图中为列指定的字体颜色。

参见

OnlineTableControl (页 402)

UseCurserKeyScroll

说明

在运行系统中无访问权限。

参见

OnlineTableControl (页 402)

Used

描述

返回使用的磁盘空间大小。

运行系统中的访问权限：读取

语法

Object.**Used**[=DOUBLE]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- DiscSpaceView

DOUBLE

可选项。用于返回已用磁盘空间大小的值或常量。

参见

DiskSpaceView (页 337)

UseDesignColorSchema

说明

指定在当前设计的全局颜色方案中定义的颜色是否将用于此对象。

运行系统中可进行的访问：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**UseDesignColorSchema**[=BOOLEAN]

Object

必选。具有以下格式的 "ScreenItem" 类型的对象：

- Bar
- Button
- CheckBox
- Circle
- CircleSegment
- CircularArc
- Clock
- ComboBox
- Connector
- Ellipse
- EllipseSegment
- EllipticalArc
- Gauge
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit
- OptionGroup
- Polygon
- Polyline
- Rectangle
- RoundButton*
- Slider*
- SymbolicIOField*
- TextField
- TubeArcObject

- TubeDoubleTeeObject
- TubeTeeObject
- Tubepolyline
- WindowSlider*

*：只读访问权限

在运行系统中您没有以下格式的访问权限：

- AlarmView
- DateTimeField
- RecipeView
- StatusForce
- Switch
- SysDiagControl
- TrendView
- UserView

BOOLEAN

可选。

如果使用全局颜色方案中为此对象类型定义的颜色显示此对象，则为 TRUE。

如果按照此对象中设置的颜色显示此对象，则为 FALSE。

UseDesignShadowSettings

描述

指定显示的对象是否带有全局阴影。在当前设计中指定了全局阴影。

运行系统中的访问权限：读取

语法

Object.**UseDesignShadowSettings**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar
- Button
- CheckBox
- CircleSegment
- CircularArc
- Clock
- ComboBox
- Connector
- EllipseSegment
- EllipticalArc
- Gauge
- GraphicIOField
- GraphicView
- IOField
- Line
- ListBox
- MultiLineEdit
- OptionGroup
- Polyline
- RoundButton
- Slider
- SymbolicIOField
- TextField
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject

- Tubepolyline
- WindowSlider

在运行系统中您没有以下格式的访问权限：

- CircleSegment

BOOLEAN

可选项。

如果使用为此对象类型指定的全局阴影显示此对象，则为 TRUE。

如果以无阴影的形式显示该对象，则为 FALSE。

参见

Circle (页 312)

CircleSegment (页 316)

CircularArc (页 320)

Ellipse (页 340)

EllipseSegment (页 344)

EllipticalArc (页 348)

GraphicView (页 375)

Line (页 388)

Polygon (页 444)

Polyline (页 448)

Rectangle (页 467)

TextField (页 527)

TubeArcObject (页 554)

TubeDoubleTeeObject (页 557)

TubePolyline (页 560)

TubeTeeObject (页 563)

Bar (页 285)

Button (页 296)

CheckBox (页 307)

Clock (页 323)

ComboBox (页 326)

Gauge (页 366)

GraphicIOField (页 371)

IOField (页 382)

Listbox (页 392)

MultiLineEdit (页 399)

OptionGroup (页 435)

RoundButton (页 471)

Slider (页 485)

SymbolicIOField (页 504)

WindowSlider (页 586)

UsedPercent

描述

以百分比形式指定已用磁盘空间的测量值。该值可以在运行系统中查询。该值无法预先定义。

运行系统中的访问权限：读取

语法

Object.**UsedPercent**[=Int32]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- DiscSpaceView

Int32

可选项。用于返回百分比形式的已用磁盘空间测量值的值或常量。

参见

DiskSpaceView (页 337)

UseExponentialFormat

描述

指定是否以指数形式显示数值（例如：“1.00e+000”）。

运行系统中的访问权限：读和写

语法

Object.**UseExponentialFormat**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

BOOLEAN

可选。若以指数形式显示数值（例如：“1.00e+000”），则为 TRUE。

参见

Bar (页 285)

UseFirstGradient

说明

在运行系统中无访问权限。

UseFlashTransparentColor

说明

指定是否将闪烁图形位图对象的颜色设置为""transparent”。

运行系统中的访问权限：读和写

语法

Object.**UseFlashTransparentColor**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- "GraphicIOField"

BOOLEAN

可选。如果将闪烁图形位图对象的颜色设置为"transparent"，则为 TRUE。

参见

GraphicIOField (页 371)

UseMessageColor

描述

指定是否显示消息类别的商定颜色。

运行系统中可进行读写访问

语法

Object.**UseMessageColor** [=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl

BOOLEAN

可选项。

TRUE，显示颜色。

FALSE，采用"布局"(Layout) 选项卡中为表格内容所指定的颜色设置。

参见

AlarmControl (页 255)

UserData（专业版）

说明

指定在执行用户自定义菜单条目或工具栏按钮时传送至 VB 脚本的值。

使用“菜单和工具栏”(Menus and toolbars) 编辑器中的“数据”(Data) 域可将参数传送至过程。

运行系统中可进行的访问：读取和写入

语法

Object.**UserData**[=String]

Object

要求"Item"类型对象。

String

可选项。在执行用户自定义菜单条目或工具栏按钮时传送至 VB 脚本的值或常量。

示例

以下示例显示了单击用户自定义菜单项时执行的脚本"ChangeScreen"。使用“数据”(Data) 字段将画面名称传送到脚本。

```
Sub ChangeScreen (ByVal Item)
    Dim objScreen, strScreenName
    ' "UserData" contains the screen name specified
    ' in editor menus and toolbars.
    strScreenName = Item.Userdata
    HMIRuntime.BaseScreenName = strScreenName
End Sub
```

UserName

说明

返回触发了报警对象的用户的名称。

UseScadaRendererStyle

说明

在运行系统中无访问权限。

UseSecondGradient

说明

在运行系统中无访问权限。

UseSelectedTitleColor

描述

指定是否将选择颜色用于选定表格单元格的标题。

运行系统中可进行读写访问

语法

Object.**UseSelectedTitleColor**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，使用突出显示颜色。在运行系统中激活“背景”和“字体”设置。

FALSE，不使用突出显示颜色。在运行系统中禁用“背景”和“字体”设置。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

UseSourceBackColors

说明

指定使用数据源的背景色。

运行系统中的访问权限：读和写

语法

Object.**UseSourceBackColors**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- TrendRulerControl

BOOLEAN

可选项。

TRUE，使用已连接趋势或表格视图的背景颜色。

FALSE，使用在值表中组态的背景颜色。

参见

TrendRulerControl (页 532)

UseSourceForeColors

说明

指定使用数据源的字体景色。

运行系统中的访问权限：读和写

语法

Object.**UseSourceForeColors**[=BOOLEAN]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- TrendRulerControl

BOOLEAN

可选项。

TRUE，使用已连接趋势或表格视图的字体颜色。

FALSE，使用在值表中组态的字体颜色。

参见

TrendRulerControl (页 532)

UseSystemScrollbarWidth

说明

在运行系统中无访问权限。

UseTableColor2

描述

指定是否用第二种行颜色来表现表格。

运行系统中可进行读写访问

语法

Object.**UseTableColor2**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，交替使用 "TableColor (页 1171)"和 "TableColor2 (页 1172)"指定的背景色。

FALSE，所有行均使用 "TableColor (页 1171)"指定的背景色。

参见

TableColor2 (页 1172)

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

UseTableHeaderFirstGradient

说明

在运行系统中无访问权限。

UseTableHeaderSecondGradient

说明

在运行系统中无访问权限。

UseTagLimitColors

说明

指定超出为对象组态的上限和下限时，是否以颜色突出显示。

运行系统中可进行读写访问

语法

Object.**UseTagLimitColors**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- IOField

BOOLEAN

可选项。"TRUE"，以颜色突出显示限值。

参见

IOField (页 382)

UseTransparentColor

描述

指定"TransparentColor"属性定义的颜色是否为透明。

运行系统中的访问权限：读和写

语法

Object.**UseTransparentColor**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- GraphicIOField
- GraphicView

BOOLEAN

可选。如果指定的颜色显示为透明，则为 TRUE。

参见

GraphicIOField (页 371)

GraphicView (页 375)

UseTransparentColorDeactivatedPicture

描述

指定是否使用由属性"TransparentColorDeactivatedPicture"为状态“禁用”定义的透明色。

运行系统中的访问权限：读取

语法

Object.**UseTransparentColorDeactivatedPicture**[=BOOLEAN]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- RoundButton

BOOLEAN

可选项。如果使用针对“已禁用"(disabled) 状态通过属性 "TransparentColorDeactivatedPicture"定义的透明色，则为 TRUE。

参见

RoundButton (页 471)

UseTransparentColorPictureOff

描述

指定是否使用由属性"TransparentColorPictureOff"针对“关闭”状态定义的透明色。

运行系统中的访问权限：读和写

语法

Object.**UseTransparentColorPictureOff**[=BOOLEAN]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- Button
- RoundButton*

*：只读访问权限

BOOLEAN

可选项。如果使用针对“关闭”(off) 状态通过属性“TransparentColorPictureOff”定义的透明色，则为 TRUE。

UseTransparentColorPictureOn

描述

指定是否使用由属性“TransparentColorPictureOn”针对“打开”状态定义的透明色。

运行系统中的访问权限：读和写

语法

Object.**UseTransparentColorPictureOn**[=BOOLEAN]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- Button
- RoundButton*

*：只读访问权限

BOOLEAN

可选项。如果使用针对“开启”(on) 状态通过属性“TransparentColorPictureOn”定义的透明色，则为 TRUE。

UseTrendNameAsLabel

描述

指定使用“名称”(Name) 还是“标签”(Label) 属性作为运行系统中趋势的标识。

运行系统中可进行读写访问

语法

Object.**UseTrendNameAsLabel**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- FunctionTrendControl
- OnlineTrendControl

BOOLEAN

可选项。

TRUE，使用在“属性 > 属性 > 趋势 > 名称”(Properties > Properties > Trends > Name) 下组态的名称。

FALSE，使用在“属性 > 属性 > 趋势 > 标签”(Properties > Properties > Trends > Label) 下组态的名称。

参见

FunctionTrendControl (页 351)

OnlineTrendControl (页 418)

UseTwoHandOperation

说明

在运行系统中无访问权限。

UseUdp

描述

在运行系统中无访问权限。

UV_ColumnWidth_AKZ

描述

在运行系统中无访问权限。

UV_ColumnWidth_Descriptor

描述

在运行系统中无访问权限。

UV_ColumnWidth_InstallationDate

描述

在运行系统中无访问权限。

UV_ColumnWidth_LADDR

描述

在运行系统中无访问权限。

UV_ColumnWidth_Name

描述

在运行系统中无访问权限。

UV_ColumnWidth_OKZ

描述

在运行系统中无访问权限。

UV_ColumnWidth_OperationState

描述

在运行系统中无访问权限。

UV_ColumnWidth_OrderID

描述

在运行系统中无访问权限。

UV_ColumnWidth_ProfileID

描述

在运行系统中无访问权限。

UV_ColumnWidth_Rack

描述

在运行系统中无访问权限。

UV_ColumnWidth_Slot

描述

在运行系统中无访问权限。

UV_ColumnWidth_SoftwareRevision

描述

在运行系统中无访问权限。

UV_ColumnWidth_SpecificProfileData

描述

在运行系统中无访问权限。

UV_ColumnWidth_State

描述

在运行系统中无访问权限。

UV_ColumnWidth_Station

描述

在运行系统中无访问权限。

UV_ColumnWidth_SubAddress

描述

在运行系统中无访问权限。

UV_ColumnWidth_SubSlot

描述

在运行系统中无访问权限。

UV_ColumnWidth_SubSystem

描述

在运行系统中无访问权限。

UV_ColumnWidth_Type

描述

在运行系统中无访问权限。

UV_ShowItem_AKZ

描述

在运行系统中无访问权限。

UV_ShowItem_Descriptor

描述

在运行系统中无访问权限。

UV_ShowItem_InstallationDate

描述

在运行系统中无访问权限。

UV_ShowItem_LADDR

描述

在运行系统中无访问权限。

UV_ShowItem_Name

描述

在运行系统中无访问权限。

UV_ShowItem_OKZ

描述

在运行系统中无访问权限。

UV_ShowItem_OperationState

描述

在运行系统中无访问权限。

UV_ShowItem_OrderID

描述

在运行系统中无访问权限。

UV_ShowItem_ProfileID

描述

在运行系统中无访问权限。

UV_ShowItem_Rack

描述

在运行系统中无访问权限。

UV_ShowItem_Slot

描述

在运行系统中无访问权限。

UV_ShowItem_SoftwareRevision

描述

在运行系统中无访问权限。

UV_ShowItem_SpecificProfileData

描述

在运行系统中无访问权限。

UV_ShowItem_State

描述

在运行系统中无访问权限。

UV_ShowItem_Station

描述

在运行系统中无访问权限。

UV_ShowItem_SubAddress

描述

在运行系统中无访问权限。

UV_ShowItem_SubSlot

描述

在运行系统中无访问权限。

UV_ShowItem_SubSystem

描述

在运行系统中无访问权限。

UV_ShowItem_Type

描述

在运行系统中无访问权限。

值

描述

为使用的对象指定值，或返回该值。

运行中可进行的访问： 读和写

语法

Object.**Value**[=VARIANT]

Object

必需。 “Tag”、“DataItem”或“ScreenItem”类型的对象，且具有“Gauge”格式。

VARIANT

可选项，指定的值取决于使用的对象：

- Tag.Value：返回最后一次读取访问的变量值，或指定将来的变量值。 使用“Read”方法从“Value”属性读取变量值。 通过“Write”方法将新变量值赋给“Value”属性。
- Dataset.Value：指定数值或返回值的复制值或对象引用。 返回对象引用时，确保对象引用具有多线程的特性。
- ScreenItem("Gauge_1").Value：指定指针所指的值。 通过属性“ValueMin”和“ValueMax”设置值的数值范围。

示例

下列示例为在“Tag1”变量中写入新值：

```
'VBS94
Dim objTag
Set objTag = HMIRuntime.Tags("Tag1")
objTag.Value = 50
objTag.Write
```

该示例为如何向变量列表添加值以及作为跟踪输出。 之后，更改数值，再次输出，然后删除。 在多个不同的操作中执行上述步骤才有意义：

```
'VBS198
HMIRuntime.DataSet.Add "motor1", 23
HMIRuntime.Trace "motor1: " & HMIRuntime.DataSet("motor1").Value & vbNewLine
HMIRuntime.DataSet("motor1").Value = 55
HMIRuntime.Trace "motor1: " & HMIRuntime.DataSet("motor1").Value & vbNewLine
HMIRuntime.DataSet.Remove("motor1")
```

参见

DataItem (页 218)

变量 (页 244)

Gauge (页 366)

ValueAxes

说明

在运行系统中无访问权限。

ValueAxis1AutoRange

说明

在运行系统中无访问权限。

ValueAxis1Begin

说明

在运行系统中无访问权限。

ValueAxis1End

说明

在运行系统中无访问权限。

ValueAxis1LabelLength

说明

在运行系统中无访问权限。

ValueAxis1Style

说明

在运行系统中无访问权限。

ValueAxis2AutoRange

说明

在运行系统中无访问权限。

ValueAxis2Begin

说明

在运行系统中无访问权限。

ValueAxis2End

说明

在运行系统中无访问权限。

ValueAxis2LabelLength

说明

在运行系统中无访问权限。

ValueAxis2Style

说明

在运行系统中无访问权限。

ValueAxisAdd

说明

创建新的数值轴。新创建的数值轴会自动使用“ValueAxisIndex”而引用。

运行系统中可进行读写访问

语法

Object.**ValueAxisAdd**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

STRING

必需项。用于通过"ValueAxisName"指定新数值轴名称的值或常数。

参见

ValueAxisIndex (页 1386)

OnlineTrendControl (页 418)

ValueAxisAlignment

说明

指定使用"ValueAxisIndex"引用的数值轴对齐方式。

运行系统中可进行读写访问

语法

Object.**ValueAxisAlignment**[=AxisAlignment]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

AxisAlignment

可选项。用于指定数值轴对齐的值或常量。

值	名称	说明
0	左对齐	所选数值轴显示在趋势左侧。
1	右对齐	所选数值轴显示在趋势右侧。

参见

OnlineTrendControl (页 418)

ValueAxisAutoPrecisions

说明

指定是否自动确定使用"ValueAxisIndex"引用的数值轴的小数位数。

运行系统中可进行读写访问

语法

Object.**ValueAxisAutoPrecisions**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

BOOLEAN

可选项。

TRUE，自动确定小数位数。

FALSE，使用"ValueAxisPrecisions"值。

参见

ValueAxisPrecisions (页 1388)

ValueAxisIndex (页 1386)

OnlineTrendControl (页 418)

ValueAxisAutoRange

描述

指定是否自动计算使用"ValueAxisIndex"引用的数值轴的取值范围。

运行系统中可进行读写访问

语法

Object.**ValueAxisAutoRange**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

BOOLEAN

可选项。

TRUE，自动计算取值范围。

FALSE，通过"ValueAxisBeginValue"和"ValueAxisEndValue"指定取值范围。

参见

ValueAxisBeginValue (页 1382)

ValueAxisEndValue (页 1384)

ValueAxisIndex (页 1386)

OnlineTrendControl (页 418)

ValueAxisBeginValue

说明

指定使用"ValueAxisIndex"引用数值轴的取值范围下限。

也可将"ValueAxisAutoRange"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**ValueAxisBeginValue**[=DOUBLE]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

DOUBLE

可选项。用于指定引用的数值轴取值范围下限的值或常量。

参见

ValueAxisIndex (页 1386)

ValueAxisAutoRange (页 1381)

OnlineTrendControl (页 418)

ValueAxisColor

说明

指定使用"ValueAxisIndex"引用的数值轴的颜色。仅在 "ValueAxisInTrendColor (页 1386)"属性设为"FALSE"时才评估该属性。

运行系统中可进行读写访问

语法

Object.**ValueAxisColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

Color

可选项。用于指定数值轴颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

OnlineTrendControl (页 418)

ValueAxisCount

说明

指定所组态的数值轴数目。

运行系统中可进行读写访问

语法

Object.**ValueAxisCount**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

Int32

可选项。用于指定可组态的数值轴数量的值或常量。

参见

OnlineTrendControl (页 418)

ValueAxisEndValue

说明

指定使用"ValueAxisIndex"引用数值轴的取值范围上限。

也可将"ValueAxisAutoRange"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**ValueAxisEndValue**[=DOUBLE]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

DOUBLE

可选项。用于指定引用的数值轴取值范围上限的值或常量。

参见

ValueAxisIndex (页 1386)

ValueAxisAutoRange (页 1381)

OnlineTrendControl (页 418)

ValueAxisExponentialFormat

说明

指定是否以指数计数法显示使用"ValueAxisIndex"引用的数值轴的值。

运行系统中可进行读写访问

语法

Object.**ValueAxisExponentialFormat**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

BOOLEAN

可选项。

TRUE，以指数计数法显示值。

FALSE，不以指数计数法显示值。

参见

ValueAxisIndex (页 1386)

OnlineTrendControl (页 418)

ValueAxisIndex

说明

引用数值轴。要访问数值轴的属性，需要设置""ValueAxisIndex""。

介于 0 至 (ValueAxisCount - 1) 之间的值为"ValueAxisIndex"的有效值。"ValueAxisCount"属性指定可组态数值轴的数目。

运行系统中可进行读写访问

语法

Object.**ValueAxisIndex**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

Int32

可选项。用于通过索引指定要编辑的数值轴的值或常量。

参见

ValueAxisCount (页 1384)

OnlineTrendControl (页 418)

ValueAxisInTrendColor

说明

指定使用"ValueAxisIndex"引用的轴的颜色是否与趋势颜色相对应。

运行系统中可进行读写访问

语法

Object.**ValueAxisInTrendColor**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

BOOLEAN

可选项。

TRUE，以趋势颜色显示引用的轴。"ValueAxisColor"值无效。

FALSE，以"ValueAxisColor"指定的颜色显示引用的轴。

参见

ValueAxisIndex (页 1386)

OnlineTrendControl (页 418)

ValueAxisLabel

描述

为使用"ValueAxisIndex"引用的数值轴定义标签文本。

运行系统中可进行读写访问

语法

Object.ValueAxisLabel[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

STRING

可选项。用于指定引用的数值轴标签文本的值或常量。

参见

ValueAxisIndex (页 1386)

OnlineTrendControl (页 418)

ValueAxisName

说明

指定使用"ValueAxisIndex"引用的数值轴的名称。

运行系统中可进行读写访问

语法

Object.**ValueAxisName**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

STRING

可选项。用于指定引用的数值轴名称的值或常量。

参见

ValueAxisIndex (页 1386)

OnlineTrendControl (页 418)

ValueAxisPrecisions

说明

为使用"ValueAxisIndex"引用的数值轴标签指定小数位数。

运行系统中可进行读写访问

语法

Object.**ValueAxisPrecisions**[=Int16]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

Int16

可选项。用于指定小数位数的值或常量。

参见

OnlineTrendControl (页 418)

ValueAxisRemove

说明

使用引用的数值轴名称可移除该数值轴。

运行系统中可进行读写访问

语法

Object.**ValueAxisRemove**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

STRING

可选项。用于指定待移除的引用的数值轴名称的值或常量。

参见

OnlineTrendControl (页 418)

ValueAxisRename

说明

指定使用"ValueAxisIndex"引用的数值轴的新名称。

运行系统中可进行读写访问

语法

Object.**ValueAxisRename**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

STRING

可选项。用于指定数值轴新名称的值或常量。

参见

ValueAxisIndex (页 1386)

OnlineTrendControl (页 418)

ValueAxisRepos

说明

指定使用"ValueAxisIndex"引用的数值轴的位置。

如果已使用"ValueAxisRepos"更改了数值轴位置，"ValueAxisRepos"的值则会分配给"ValueAxisIndex"。

运行系统中可进行读写访问

语法

Object.**ValueAxisRepos**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

Int32

可选项。用于指定引用的数值轴位置的值或常量。值范围为 0 到 (ValueAxisCount - 1)。超出该范围的值无效。

0：引用的时间轴放置于外部。

参见

ValueAxisIndex (页 1386)

ValueAxisCount (页 1384)

OnlineTrendControl (页 418)

ValueAxisScalingType

描述

指定使用"ValueAxisIndex"引用的数值轴比例缩放的类型。

运行系统中可进行读写访问

语法

Object.ValueAxisScalingType[=AxisScalingType]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

AxisScalingType

可选项。用于指定引用的数值轴比例缩放的类型的值或常量。

值	VB 常量	名称
0	hmiBarScalingLinear	线性
1	hmiBarScalingLogarithmic	对数
2	hmiBarScalingNegativeLogarithmic	负对数

参见

ValueAxisIndex (页 1386)

OnlineTrendControl (页 418)

ValueAxisTrendWindow

说明

指定使用"ValueAxisIndex"引用的轴在其中显示的趋势图。

运行系统中可进行读写访问

语法

Object.**ValueAxisTrendWindow**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

STRING

可选项。用于指定趋势图名称的值或常量。

参见

TrendWindowCount (页 1329)

ValueAxisIndex (页 1386)

OnlineTrendControl (页 418)

ValueAxisVisible

说明

指定是否在对象中显示使用"ValueAxisIndex"引用的数值轴。

运行系统中可进行读写访问

语法

Object.**ValueAxisVisible**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTrendControl

BOOLEAN

可选项。

TRUE，显示引用的数值轴。

FALSE，不显示引用的数值轴。

参见

ValueAxisIndex (页 1386)

OnlineTrendControl (页 418)

ValueCaption

说明

在运行系统中无访问权限。

ValueColumnAdd

说明

创建新的数值列。新创建的数值列会自动使用“ValueColumnIndex”而引用。

运行系统中可进行读写访问

语法

Object.**ValueColumnAdd**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- OnlineTableControl

STRING

必需项。用于通过“ValueColumnName”指定新数值列名称的值或常数。

参见

ValueColumnIndex (页 1400)

OnlineTableControl (页 402)

ValueColumnAlignment

描述

指定使用"ValueColumnIndex"引用的数值列中的文本如何对齐。

运行系统中可进行读写访问

语法

Object.ValueColumnAlignment[=HorizontalAlignment]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

HorizontalAlignment

可选项。用于指定引用的数值列中的文本如何对齐的值或常量。

值	VB 常量	说明
0	hmiAlignmentLeft	文本左对齐。
1	hmiAlignmentCentered	文本居中。
2	hmiAlignmentRight	文本右对齐。

参见

OnlineTableControl (页 402)

ValueColumnAutoPrecisions

说明

指定是否自动确定使用"ValueColumnIndex"引用的数值轴的小数位数。

运行系统中可进行读写访问

语法

Object.**ValueColumnAutoPrecisions**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

BOOLEAN

可选项。

TRUE，自动确定小数位数。

FALSE，使用"ValueColumnPrecisions"值。

参见

ValueColumnPrecisions (页 1402)

ValueColumnIndex (页 1400)

OnlineTableControl (页 402)

ValueColumnBackColor

说明

指定使用"ValueColumnIndex"引用的数值列的背景色。

也可将 "UseColumnBackColor (页 1351)"设置为"TRUE"。

运行系统中可进行读写访问

语法

Object.**ValueColumnBackColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

Color

可选项。用于指定背景色的数值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

OnlineTableControl (页 402)

ValueColumnCaption

说明

指定使用"ValueColumnIndex"引用的数值列的名称。

运行系统中可进行读写访问

语法

Object.**ValueColumnCaption**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

STRING

可选项。用于指定数值列名称的数值或常量。

参见

OnlineTableControl (页 402)

ValueColumnCount

说明

指定所组态的数值列数目。

运行系统中可进行读写访问

语法

Object.**ValueColumnCount**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

Int32

可选项。用于指定已组态的数值列数量的值或常量。

参见

OnlineTableControl (页 402)

ValueColumnExponentialFormat

说明

指定是否以指数计数法显示使用"ValueColumnIndex"引用的数值列的值。

运行系统中可进行读写访问

语法

Object.**ValueColumnExponentialFormat**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

BOOLEAN

可选项。

TRUE，以指数计数法显示值。

FALSE，不以指数计数法显示值。

参见

ValueColumnIndex (页 1400)

OnlineTableControl (页 402)

ValueColumnForeColor

说明

指定使用"ValueColumnIndex"引用的数值列的字体颜色。

也可将 "UseColumnForeColor (页 1352)"设置为"TRUE"。

运行系统中可进行读写访问

语法

Object.**ValueColumnForeColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

Color

可选项。用于指定字体颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

OnlineTableControl (页 402)

ValueColumnHideText

说明

指定是否隐藏使用"ValueColumnIndex"引用的数值列的文本。

运行系统中可进行读写访问

语法

Object.**ValueColumnHideText**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

BOOLEAN

可选项。

TRUE，以隐藏数值列文本。

FALSE，以显示数值列文本。

参见

OnlineTableControl (页 402)

ValueColumnHideTitleText

说明

指定是否隐藏使用"ValueColumnIndex"引用的数值列标题的文本。

运行系统中可进行读写访问

语法

Object.**ValueColumnHideTitleText**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

BOOLEAN

可选项。

TRUE，以隐藏数值列标题文本。

FALSE，以显示数值列标题文本。

参见

OnlineTableControl (页 402)

ValueColumnIndex

说明

引用数值列。要访问列的属性，需要设置""ValueColumnIndex""。

介于 0 至 (ValueColumnCount - 1) 之间的值为"ValueColumnIndex"的有效值。"ValueColumnCount"属性指定可组态数值列的数目。

运行系统中可进行读写访问

语法

Object.**ValueColumnIndex**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

Int32

可选项。用于通过索引指定要编辑的数值列的值或常量。

参见

ValueColumnCount (页 1397)

OnlineTableControl (页 402)

ValueColumnLength

说明

指定使用"ValueColumnIndex"引用的数值列的宽度（以字符为单位）。

运行系统中可进行读写访问

语法

Object.**ValueColumnLength**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

Int32

可选项。用于指定所显示字符数的值或常量。

参见

OnlineTableControl (页 402)

ValueColumnName

说明

指定使用"ValueColumnIndex"引用的数值列的名称。

运行系统中可进行读写访问

语法

Object.**ValueColumnName**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

STRING

可选项。用于指定引用的数值列名称的值或常量。

参见

ValueColumnIndex (页 1400)

OnlineTableControl (页 402)

ValueColumnPrecisions

说明

指定使用"ValueColumnIndex"引用的数值列中数值的小数位数。

运行系统中可进行读写访问

语法

Object.**ValueColumnPrecisions**[=Int16]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

Int16

可选项。用于指定小数位数的值或常量。

参见

OnlineTableControl (页 402)

ValueColumnProvider

说明

指定使用"ValueColumnIndex"引用的数值列数据源。

运行系统中可进行读写访问

语法

Object.**ValueColumnProvider**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

STRING

可选项。用于指定数值列数据源的值或常数。

值	说明
记录变量	具有过程值归档的归档变量的数据源。
变量	具有来自变量管理的 在线变量的数据源。

参见

OnlineTableControl (页 402)

ValueColumnProviderCLSID

说明

指定使用"ValueColumnIndex"引用的数值列的数据源 CLSID。

运行系统中可进行读写访问

语法

Object.**ValueColumnProviderCLSID**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

STRING

可选项。用于指定数据源 CLSID 的数值或常量。

值	说明
{416A09D2-8B5A-11D2-8B81-006097A45D48}	具有过程值归档的归档变量的数据源。
{A3F69593-8AB0-11D2-A440-00A0C9DBB64E}	具有来自变量管理的 在线变量的数据源。

参见

OnlineTableControl (页 402)

ValueColumnRemove

说明

使用引用的数值列名称可移除该数值列。

运行系统中可进行读写访问

语法

Object.**ValueColumnRemove**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

STRING

可选项。用于指定待移除的引用的数值列名称的值或常量。

参见

OnlineTableControl (页 402)

ValueColumnRename

说明

指定使用"ValueColumnIndex"引用的数值列的新名称。

运行系统中可进行读写访问

语法

Object.**ValueColumnRename**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

STRING

可选项。用于指定数值列新名称的值或常数。

参见

ValueColumnIndex (页 1400)

OnlineTableControl (页 402)

ValueColumnRepos

说明

指定使用"ValueColumnIndex"引用的数值列的位置。

如果已使用"ValueColumnRepos"更改了数值列位置，"ValueColumnRepos"的值则会分配给"ValueColumnIndex"。

运行系统中可进行读写访问

语法

Object.**ValueColumnRepos**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

Int32

可选项。用于指定引用的数值列位置的值或常量。值范围为 0 到 (ValueColumnCount - 1)。超出该范围的值无效。

0：引用的数值列放置在左侧。

参见

ValueColumnIndex (页 1400)

ValueColumnCount (页 1397)

OnlineTableControl (页 402)

ValueColumns

说明

在运行系统中无访问权限。

ValueColumnSelectTagName

说明

指定用于选择使用"ValueColumnIndex"引用的数值列的数据源变量名称的对话框在运行系统中初次显示。

运行系统中可进行读写访问

语法

Object.**ValueColumnSelectTagName**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

BOOLEAN

可选项。

TRUE，以在画面中显示相应对话框，从而可选择数值列数据源的变量名称。

FALSE, ，在画面中不显示相应对话框，从而无法选择数值列数据源的变量名称。

参见

OnlineTableControl (页 402)

ValueColumnShowIcon

说明

指定是否在使用"ValueColumnIndex"引用的数值列中显示图标。

运行系统中可进行读写访问

语法

Object.**ValueColumnShowIcon**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

BOOLEAN

可选项。

TRUE，以显示图标。

FALSE，不显示图标。

参见

OnlineTableControl (页 402)

ValueColumnShowTitleIcon

说明

指定是否在使用"ValueColumnIndex"引用的数值列的标题中显示图标。

运行系统中可进行读写访问

语法

Object.**ValueColumnShowTitleIcon**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

BOOLEAN

可选项。

TRUE，以在当前列的标题中显示图标。

FALSE，在当前列的标题中不显示图标。

参见

OnlineTableControl (页 402)

ValueColumnSort

说明

指定使用"ValueColumnIndex"引用的数值列的排序类型。

运行系统中可进行读写访问

语法

Object.**ValueColumnSort**[=SortMode]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

SortMode

可选项。用于指定当前数值列排序类型的值或常量。

值	名称	说明
0	无排序	不对列中的值进行排序。
1	升序	从最顶行开始，按照从最小值到最大值的顺序对数值列中的值进行排序。
2	降序	从最底行开始，按照从最大值到最小值的顺序对数值列中的值进行排序。

参见

OnlineTableControl (页 402)

ValueColumnSortIndex

说明

指定排序顺序。

运行系统中的访问权限：读和写

语法

Object.**ValueColumnSortIndex**[=Int32]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- OnlineTableControl

Int32

可选项。 用于指定排序顺序的值或常量。

参见

OnlineTableControl (页 402)

ValueColumnTagName

说明

指定其数值显示在使用“ValueColumnIndex”引用的数值列中的变量的名称。

运行系统中可进行读写访问

语法

Object.**ValueColumnTagName**[=STRING]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- OnlineTableControl

STRING

可选项。用于指定要在当前列中显示其数值的变量名称的值或常量。

参见

OnlineTableControl (页 402)

ValueColumnTimeColumn

说明

指定相关时间列。

运行系统中的访问权限：读和写

语法

Object.**ValueColumnTimeColumn**[=STRING]

Object

必需项。 “ScreenItem”对象，且具有以下特性：

- OnlineTableControl

STRING

可选项。 用于指定相关时间列的值或常量。

参见

OnlineTableControl (页 402)

ValueColumnVisible

说明

指定是否在对象中显示使用"ValueColumnIndex"引用的数值列。

运行系统中可进行读写访问

语法

Object.**ValueColumnVisible**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- OnlineTableControl

BOOLEAN

可选项。

TRUE，显示引用的数值列。

FALSE，不显示引用数值列。

参见

ValueColumnIndex (页 1400)

OnlineTableControl (页 402)

ValueColumnWidth

说明

在运行系统中无访问权限。

ValueTableHeight

说明

在运行系统中无访问权限。

ValueTableLeft

说明

在运行系统中无访问权限。

ValueTableTop

说明

在运行系统中无访问权限。

ValueTableWidth

说明

在运行系统中无访问权限。

ValueY1HlpLine

说明

在运行系统中无访问权限。

ValueY2HlpLine

说明

在运行系统中无访问权限。

VerticalAlignment

描述

指定所选对象中的文本垂直对齐方式。

运行系统中的访问权限：读和写

语法

Object.**VerticalAlignment**[=VerticalAlignment]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Button
- CheckBox
- DateTimeField
- IOField
- OptionGroup
- RoundButton *
- Switch *
- SymbolicIOField **
- TextField

* 只读访问权限

** RT Professional，只读访问权限

VerticalAlignment

可选项。用于指定垂直对齐的数值。

值	VB 常量	说明
0	hmiAlignmentTop	文本显示在顶部。
1	hmiAlignmentMiddle	文本显示居中。
2	hmiAlignmentBottom	文本显示于底部。

参见

Button (页 296)

DateTimeField (页 334)

IOField (页 382)

OptionGroup (页 435)

RoundButton (页 471)

Switch (页 499)

SymbolicIOField (页 504)

TextField (页 527)

CheckBox (页 307)

VerticalGridLines

描述

指定是否显示垂直分隔线。

运行系统中可进行读写访问

语法

Object.**VerticalGridLines**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- AlarmControl
- OnlineTableControl
- TrendRulerControl
- UserArchiveControl

BOOLEAN

可选项。

TRUE，显示垂直分隔线。

FALSE，不显示垂直分隔线。

参见

AlarmControl (页 255)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

VerticalPictureAlignment

说明

在运行系统中无访问权限。

VerticalScrollBarEnabled

说明

在运行系统中无访问权限。

VerticalScrollbarPosition

说明

指定滑块在对象垂直滚动条中的位置。 通过垂直移动滚动条中的滑块可在画面窗口中显示画面。

如果要将画面显示为剪切状，其中滚动条位于画面的左上边缘，请使用“LeftOffset (页 926)”和“LeftOffset (页 926)”属性来指定该剪切状画面。

运行系统中的访问权限： 读和写

语法

Object.**VerticalScrollBarPosition**[=Int32]

Object

必需项。 具有以下格式的“ScreenItem”类型的对象：

- Screenwindow

Int32

可选项。 用于指定滑块在对象滚动条中垂直位置的值或常量。

参见

TopOffset (页 1282)

ScreenWindow (页 479)

VerticalScrolling

说明

在运行系统中无访问权限。

VerticalScrollingEnabled

说明

在运行系统中无访问权限。

ViewOnly

描述

指定 Sm@rtClient 显示用于远程监视还是远程维护。

远程维护意味着可以在监视设备上更改设置。

远程监视意味着无法更改监视设备的设置。

运行系统中的访问权限：读和写

语法

Object.**ViewOnly**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- SmartClientView

BOOLEAN

可选。若 Sm@rtClient 显示仅用于远程监视，则为 TRUE。

参见

SmartClientView (页 490)

ViewType

说明

在运行系统中无访问权限。

ViewTypeForSaveStream

说明

运行系统中的访问权限：

- RT Advanced：读取
- RT Professional：无访问权

语法

Object.ViewTimeForSaveStream[=[!]: AlarmViewBasicMode | Int16 | PasswordViewType]

Object

必选项。具有以下格式的"ScreenItem"类型的对象：

- RecipeView

在运行系统中您没有以下格式的访问权限：

- AlarmView
- UserView

[!]: AlarmViewBasicMode | Int16 | PasswordViewType:

可选项。tbd tbd tbd tbd are.

Visible

描述

指定是否显示所选对象。

运行系统中的访问权限：读和写

语法

Object.**Visible**[=BOOLEAN]

Object

必需项。具有以下格式的 "ScreenItem" 类型的对象：

- AlarmControl
- AlarmView
- ApplicationWindow
- Bar
- BatteryView
- Button
- ChannelDiagnose
- CheckBox
- CircleSegment
- CircularArc
- ComboBox
- Connector
- DateTimeField
- DiscSpaceView
- EllipseSegment
- EllipticalArc
- FunctionTrendControl
- Gauge
- GraphicIOField

- GraphicView
- HTMLBrowser
- IOField
- ListBox
- MediaPlayer
- MultiLineEdit
- OnlineTableControl
- OnlineTrendControl
- OptionGroup
- PLCCodeViewer
- ProDiagOverview
- ProtectedAreaNameView
- RangeLabelView
- RangeQualityView
- RecipeView
- RoundButton
- S7GraphOverview
- ScreenWindow
- SmartClientView
- StatusForce
- Switch
- SymbolLibrary
- SymbolicIOField
- SysDiagControl
- TrendRulerControl
- TrendView
- TubeArcObject
- TubeDoubleTeeObject
- TubeTeeObject

- Tubepolyline
- UserArchiveControl
- WLanQualityView
- WindowSlider
- ZoneLabelView
- ZoneQualityView

在运行系统中您没有以下格式的访问权限：

- PdfView

BOOLEAN

可选。如果显示对象，则为 TRUE。

参见

AlarmControl (页 255)

AlarmView (页 274)

ApplicationWindow (页 282)

Bar (页 285)

BatteryView (页 294)

Button (页 296)

ChannelDiagnose (页 305)

CheckBox (页 307)

Circle (页 312)

CircleSegment (页 316)

CircularArc (页 320)

Clock (页 323)

ComboBox (页 326)

DateTimeField (页 334)

DiskSpaceView (页 337)

Ellipse (页 340)

EllipseSegment (页 344)

SysDiagControl (页 514)

TextField (页 527)

TrendRulerControl (页 532)

TrendView (页 546)

TubeArcObject (页 554)

TubeDoubleTeeObject (页 557)

TubePolyline (页 560)

TubeTeeObject (页 563)

用户归档控件 (页 566)

UserView (页 581)

WindowSlider (页 586)

ZoneLabelView (页 593)

ZoneQualityView (页 595)

S7GraphOverview (页 476)

WLanQualityView (页 591)

VisibleItems

说明

在运行系统中无访问权限。

Warning

描述

指定所用存储空间的百分比限值（超出此值将生成警告）。

运行系统中的访问权限：读取

语法

Object.**Warning**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- DiskSpaceView

Int32

可选项。用于指定所用存储空间百分比限值（超出此值将生成警告）的值或常量。

参见

DiskSpaceView (页 337)

WarningColor

描述

指定达到警告范围时存储空间显示栏的显示颜色。

运行系统中可进行读访问

语法

Object.**WarningColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- DiskSpaceView

Color

可选项。用于指定在超过警告范围时存储空间显示栏的显示颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

DiskSpaceView (页 337)

WarningLowerLimit

描述

指定"WarningLowerLimit"的下限。

"WarningLowerLimitEnable"属性值必须为"TRUE"才能监控限值。

运行系统中的访问权限：读和写

语法

Object.**WarningLowerLimit**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

DOUBLE

可选项。用于指定"WarningLowerLimit"下限的值或常量。

注释

通过属性"WarningLowerLimitColor"和"WarningLowerLimitRelative"指定以下值：

- 达到限值后的显示
- 估算的类型

参见

Bar (页 285)

WarningLowerLimitColor

描述

指定"WarningLowerLimit"下限的颜色。

若滚动条颜色在达到限值后立即更改，则"WarningLowerLimitEnable"属性值必须为 TRUE。

运行系统中可进行读写访问

语法

Object.**WarningLowerLimitColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar

Color

可选项。用于指定"WarningLowerLimit"下限颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

以下值通过"WarningUpperLimit"、"WarningUpperLimitColor"和"WarningUpperLimitRelative"属性定义：

- 限值
- 达到限值后的显示
- 估算的类型

参见

Bar (页 285)

WarningLowerLimitEnabled

描述

指定是否监视"WarningLowerLimit"极限。

运行系统中的访问权限：读和写

语法

Object.**WarningLowerLimitEnabled**[=BOOLEAN]

Object

必需项。 "ScreenItem" 对象，且具有以下格式：

- Bar

BOOLEAN

可选。如果监视“WarningLowerLimit”限值，则为 TRUE。

注释

通过属性“WarningLowerLimit”、“WarningLowerLimitColor”和“WarningLowerLimitRelative”定义以下值：

- 限值
- 达到限值后的显示
- 估算的类型

参见

Bar (页 285)

WarningLowerLimitRelative

描述

指定“WarningLowerLimit”下限是以百分比还是绝对值形式进行估算。

运行系统中可进行读写访问

语法

Object.**WarningLowerLimitRelative**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- Bar

BOOLEAN

可选项。

TRUE，将"WarningLowerLimit"下限以百分比形式估算。

FALSE，将"WarningLowerLimit"下限以绝对值形式估算。

参见

Bar (页 285)

WarningRangeColor

描述

指定"Gauge"对象警告刻度范围的颜色。

"WarningRangeVisible"属性的值必须为 TRUE，以显示警告范围。

运行系统中可进行读写访问

语法

Object.**WarningRangeColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Gauge

Color

可选项。用于指定警告范围颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Gauge (页 366)

WarningRangeStart

描述

指定“Gauge”对象警告范围的起始刻度值。

“WarningRangeVisible”属性的值必须为 TRUE，以显示警告范围。

运行系统中的访问权限：读和写

语法

Object.**WarningRangeStart**[=SINGLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Gauge

SINGLE

可选项。其中包含警告范围起始刻度值的值或常量。

注释

取值范围为“Warning”到“Danger”。 若“Danger”未设定范围，则“Warning”的范围将一直到刻度终止。

参见

Gauge (页 366)

WarningRangeVisible

描述

指定是否显示“Gauge”对象的警告刻度范围。

运行系统中的访问权限：读和写

语法

Object.**WarningRangeVisible**[=BOOLEAN]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Gauge

BOOLEAN

可选。如果要显示警告刻度范围，则为 TRUE。

注释

指定“WarningRangeColor”属性的警告范围的颜色。

指定“WarningRangeStart”属性的警告范围的起始值。

参见

Gauge (页 366)

WarningUpperLimit

描述

指定警告上限。

“WarningUpperLimitEnabled””属性值必须为 TRUE 才能监控限值。

运行系统中的访问权限：读和写

语法

Object.**WarningUpperLimit**[=DOUBLE]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

DOUBLE

可选。用于指定警告上限的值或常量。

注释

"WarningUpperLimitColor"定义达到限值时的显示。

"WarningUpperLimitRelative"指定估算类型。

参见

Bar (页 285)

WarningUpperLimitColor

描述

指定警告上限值的颜色。

若滚动条颜色在达到限值后立即更改，则 "WarningUpperLimitEnabled" 属性值必须为 TRUE。

运行系统中的访问权限

- RT Advanced：读取
- RT Professional：读和写

语法

Object.**WarningUpperLimitColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar

Color

可选项。用于指定上限颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

Bar (页 285)

WarningUpperLimitEnabled

描述

指定是否监视上限。

运行系统中的访问权限：

- RT Advanced：读取
- RT Professional：读和写

语法

Object.**WarningUpperLimitEnabled**[=BOOLEAN]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- Bar

BOOLEAN

可选项。如果上限受监视，则为 TRUE。

WarningUpperLimitRelative

描述

指定"WarningUpperLimit"上限是以百分比还是绝对值进行估算。

运行系统中可进行读写访问

语法

Object.**WarningUpperLimitRelative**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Bar

BOOLEAN

可选项。TRUE，将"WarningUpperLimit"下限以百分数形式估算。

参见

Bar (页 285)

Width

描述

指定对象的宽度（以像素为单位）。

运行系统中的访问权限：读和写

在"Runtime Advanced"和"Panel Runtime"中，用户具备以下格式的只读权限：

- AlarmView
- Bar
- BatteryView
- Button
- CameraControl
- Clock
- DateTimeField
- FunctionTrendControl
- Gauge
- GraphicIOField
- GraphicView
- HTMLBrowser
- IOField
- Line
- MediaPlayer
- PDFView
- PLCCodeViewer

- Polygon
- Polyline
- ProDiagOverview
- ProtectedAreaNameView
- RangeLabelView
- RangeQualityView
- RecipeView
- S7GraphOverview
- Slider
- SmartClientView
- StatusForce
- Switch
- SymbolicIOField
- SymbolLibrary
- SysDiagControl
- TextField
- TrendView
- UserView
- WLanQualityView
- ZoneLabelView
- ZoneQualityView

语法

Object.**Width**[=Int32]

Object

必需项。“ScreenItem”类型的对象。该属性是 ScreenItem 对象的标准属性，因此适用于所有格式。

Int32

可选项。用于指定以像素为单位的对象宽度的值或常量。

参见

AlarmControl (页 255)

AlarmView (页 274)

ApplicationWindow (页 282)

Bar (页 285)

BatteryView (页 294)

Button (页 296)

ChannelDiagnose (页 305)

CheckBox (页 307)

Circle (页 312)

CircleSegment (页 316)

CircularArc (页 320)

Clock (页 323)

ComboBox (页 326)

DateTimeField (页 334)

DiskSpaceView (页 337)

Ellipse (页 340)

EllipseSegment (页 344)

EllipticalArc (页 348)

FunctionTrendControl (页 351)

Gauge (页 366)

GraphicIOField (页 371)

GraphicView (页 375)

HTMLBrowser (页 379)

IOField (页 382)

Line (页 388)

Listbox (页 392)

MediaPlayer (页 396)

MultiLineEdit (页 399)

WindowSlider (页 586)

WLanQualityView (页 591)

ZoneLabelView (页 593)

ZoneQualityView (页 595)

S7GraphOverview (页 476)

WindowCloseEnabled

描述

指示是否可在运行系统中关闭窗口。

运行系统中的访问权限：读和写

语法

Object.**WindowCloseEnabled**[=BOOLEAN]

Object

必选项。"ScreenItem" 对象，且具有以下格式：

- ApplicationWindow
- ScreenWindow

BOOLEAN

可选项。如果可在运行系统中关闭窗口，则为 TRUE。

参见

ApplicationWindow (页 282)

ScreenWindow (页 479)

WindowMaximizeEnabled

描述

返回关于运行系统中是否可以最大化对象的信息。

运行系统中可进行读写访问

语法

Object.**WindowMaximizeEnabled**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- ApplicationWindow
- Screenwindow

BOOLEAN

可选项。

TRUE，对象可以在运行系统中最大化。

FALSE，对象不可以在运行系统中最大化。

参见

ApplicationWindow (页 282)

ScreenWindow (页 479)

WindowMovingEnabled

描述

返回关于运行系统中是否可以移动对象的信息。

运行系统中的访问权限：读和写

语法

Object.**WindowMovingEnabled**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- ApplicationWindow
- Screenwindow

BOOLEAN

可选项。

TRUE，对象可以在运行系统中移动。

FALSE，对象不可以在运行系统中移动。

参见

ApplicationWindow (页 282)

ScreenWindow (页 479)

WindowOnTop

描述

返回关于对象是否始终位于运行系统的前景中的信息。

运行系统中的访问权限：读和写

语法

Object.**WindowOnTop**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- ApplicationWindow
- Screenwindow

BOOLEAN

可选项。

TRUE，对象始终位于运行系统的前景中。

FALSE，对象并非始终位于运行系统的前景中。

参见

ApplicationWindow (页 282)

ScreenWindow (页 479)

WindowsContents

说明

返回打印作业或脚本诊断的内容

运行系统中可进行读写访问

语法

Object.**WindowsContents**[=WindowContent]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- ApplicationWindow

WindowContent

可选项。用于返回对象中显示内容的值或常量。

名称	描述
功能	显示全局脚本的应用程序。此处使用的模板在"Template (页 1184)"属性中定义。 • GSC 诊断 显示诊断系统的结果。 • GSC 运行系统 显示运行系统中关于特征的分析结果。
打印作业	显示报表。 此处使用的模板在"Template (页 1184)"属性中指定：

参见

ApplicationWindow (页 282)

WindowSizingEnabled

描述

指示是否可更改大小。

运行系统中可进行读写访问

语法

Object.**WindowSizingEnabled**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- ApplicationWindow
- ScreenWindow

BOOLEAN

可选项。TRUE, 可更改大小。

参见

ApplicationWindow (页 282)

ScreenWindow (页 479)

WindowsStyle

描述

指定是否以普通 Windows 样式显示对象。

运行系统中的访问权限：

- RT Advanced：无访问权
- RT Professional：读和写

语法

Object.**WindowsStyle**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- Button
- WindowSlider

在运行系统中您没有以下格式的访问权限：

- Switch

BOOLEAN

可选。如果以普通 Windows 样式显示对象，则为 TRUE。

参见

Button (页 296)

WindowSlider (页 586)

Switch (页 499)

WindowStartupPosition

说明

组态独立的画面窗口时，返回画面窗口的位置和模式。将 “IndipendentWindow (页 888)”设为“TRUE”。

运行系统中的访问权限：读和写

语法

Object.**WindowsStartupPosition**[=PositionMode]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- ScreenWindow

PositionMode

可选项。用于返回画面窗口位置和模式的值或常量。

名称	说明
最大化 (Maximized)	将画面窗口调整到显示器大小。
按照组态 (As configured)	画面窗口以组态的大小显示在组态的位置。
居中 (Centered)	画面窗口以组态的大小显示在居中位置。

参见

ScreenWindow (页 479)

1.5.5.15 属性 X-Z

XAxes

说明

在运行系统中无访问权限。

XAxisAdd

说明

创建新的 X 轴。会使用"XAxisIndex"自动引用新创建的 X 轴。

运行系统中的访问权限：读和写

语法

Object.**XAxisAdd**[=STRING]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

STRING

必需项。用于通过"XAxisName"指定新 X 轴名称的值或常量。

参见

XAxisIndex (页 1448)

FunctionTrendControl (页 351)

XAxisAlignment

说明

指定使用"XAxisIndex"引用的 X 轴的对齐方式。

运行系统中可进行读写访问

语法

Object.XAxisAlignment[=AxisAlignment]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

AxisAlignment

可选项。用于指定关于趋势窗口 X 轴对齐方式的值或常量：

值	名称
0	左侧
1	右侧

参见

FunctionTrendControl (页 351)

XAxisAutoPrecisions

说明

指定是否自动确定使用"XAxisIndex"引用的 X 轴的小数位数。

运行系统中可进行读写访问

语法

Object.**XAxisAutoPrecisions**[=BOOLEAN]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

BOOLEAN

可选项。

TRUE，自动确定小数位数。

FALSE，使用"XAxisPrecisions"值。

参见

XAxisIndex (页 1448)

FunctionTrendControl (页 351)

XAxisAutoRange

描述

指定是否自动计算使用"XAxisIndex"引用的 X 轴的取值范围。

运行系统中可进行读写访问

语法

Object.**XAxisAutoRange**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

BOOLEAN

可选项。

TRUE，自动计算取值范围。

FALSE，通过"XAxisBeginValue"和"XAxisEndValue"指定取值范围。

参见

XAxisIndex (页 1448)

XAxisEndValue (页 1447)

XAxisBeginValue (页 1444)

FunctionTrendControl (页 351)

XAxisBeginValue

描述

指定使用"XAxisIndex"引用的 X 轴的取值范围下限。

也可将"XAxisAutoRange"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**XAxisBeginValue**[=DOUBLE]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

DOUBLE

可选项。用于指定引用的 X 轴取值范围下限的值或常量。

参见

XAxisAutoRange (页 1444)

XAxisIndex (页 1448)

FunctionTrendControl (页 351)

XAxisColor

描述

指定使用"XAxisIndex"引用的 x 轴的颜色。

也可将"XAxisInTrendColor"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**XAxisColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

Color

可选项。用于指定引用的轴颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

FunctionTrendControl (页 351)

XAxisCount

描述

指定所组态的 x 轴的数目。

运行系统中的访问权限：读和写

语法

Object.**XAxisCount**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

Int32

可选项。用于指定已组态的 X 轴数量的值或常量。

参见

FunctionTrendControl (页 351)

XAxisEndValue

描述

指定使用"XAxisIndex"引用的 x 轴的取值范围上限。

也可将"XAxisAutoRange"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**XAxisEndValue**[=DOUBLE]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

DOUBLE

可选项。用于指定引用的 X 轴取值范围上限的值或常量。

参见

XAxisAutoRange (页 1444)

XAxisIndex (页 1448)

FunctionTrendControl (页 351)

XAxisExponentialFormat

描述

指定是否以指数计数法显示使用"XAxisIndex"引用的 X 轴的值。

运行系统中可进行读写访问

语法

Object.**XAxisExponentialFormat**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

BOOLEAN

可选项。

TRUE，以指数计数法显示值。

FALSE，不以指数计数法显示值。

参见

XAxisIndex (页 1448)

FunctionTrendControl (页 351)

XAxisIndex

说明

引用 x 轴。要访问 X 轴的属性，需要设置 "YAxisIndex"。

介于 0 至 (XAxisCount - 1) 之间的值为"XAxisIndex"的有效值。"XAxisCount"属性指定已组态 X 轴的数目。

运行系统中可进行读写访问

语法

Object.**XAxisIndex**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

Int32

可选项。用于通过索引指定要编辑的 X 轴的值或常量。

参见

XAxisCount (页 1446)

FunctionTrendControl (页 351)

XAxisInTrendColor

说明

指定使用"XAxisIndex"引用的轴的颜色是否与趋势颜色相对应。

运行系统中可进行读写访问

语法

Object.**XAxisInTrendColor**[=BOOLEAN]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

BOOLEAN

可选项。

TRUE，以趋势颜色显示引用的轴。"XAxisColor"值无效。

FALSE，以"XAxisColor"指定的颜色显示引用的轴。

参见

XAxisIndex (页 1448)

FunctionTrendControl (页 351)

XAxisLabel

说明

指定使用"XAxisIndex"引用的 X 轴的标签文本。

运行系统中可进行读写访问

语法

Object.**XAxisLabel**[=STRING]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

STRING

可选。用于指定引用的 X 轴标签文本的值或常量。

参见

XAxisIndex (页 1448)

FunctionTrendControl (页 351)

XAxisName

描述

指定使用"XAxisIndex"引用的轴的名称。

运行系统中可进行读写访问

语法

Object.**XAxisName**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

STRING

可选项。用于指定引用的 X 轴名称的值或常量。

参见

XAxisIndex (页 1448)

FunctionTrendControl (页 351)

XAxisPrecisions

描述

指定使用"XAxisIndex"引用的 X 轴的值显示的小数位数。

也可将"XAxisAutoPrecisions"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**XAxisPrecisions**[=Int16]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

Int16

可选项。用于指定小数位数的值或常量。

参见

FunctionTrendControl (页 351)

XAxisRemove

描述

使用引用的 X 轴名称可移除该 X 轴。

运行系统中可进行读写访问

语法

Object.**XAxisRemove**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

STRING

可选项。用于指定待移除的引用的 X 轴名称的值或常量。

参见

FunctionTrendControl (页 351)

XAxisRename

说明

指定使用"XAxisIndex"引用的 X 轴的新名称。

运行系统中可进行读写访问

语法

Object.**XAxisRename**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

STRING

可选项。用于指定 X 轴新名称的值或常量。

参见

XAxisIndex (页 1448)

FunctionTrendControl (页 351)

XAxisRepos

说明

对于多个 X 轴，指定使用"XAxisIndex"引用的 X 轴的位置。

如果已使用"XAxisRepos"更改了 X 轴的位置，"XAxisRepos"的值则会分配给"XAxisIndex"。

运行系统中可进行读写访问

语法

Object.**XAxisRepos**[=Int32]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

Int32

可选。用于指定引用的 X 轴位置的值或常量。值范围为 0 到 (XAxisCount - 1)。超出该范围的值无效。

0：引用的时间轴放置于外部。

参见

XAxisIndex (页 1448)

XAxisCount (页 1446)

FunctionTrendControl (页 351)

XAxisScalingType

说明

指定使用"XAxisIndex"引用的 X 轴标定类型。

运行系统中可进行读写访问

语法

Object.**XAxisScalingType**[=AxisScalingType]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

AxisScalingType

可选项。用于指定所引用 X 轴标定类型的值或常量。

值	VB 常量	名称
0	hmiBarScalingLinear	线性
1	hmiBarScalingLogarithmic	对数
2	hmiBarScalingNegativeLogarithmic	负对数

参见

XAxisIndex (页 1448)

FunctionTrendControl (页 351)

XAxisTrendWindow

描述

指定使用"XAxisIndex"引用的轴在其中显示的趋势图。

运行系统中可进行读写访问

语法

Object.**XAxisTrendWindow**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

STRING

可选项。用于指定趋势图名称的值或常量。

参见

XAxisIndex (页 1448)

FunctionTrendControl (页 351)

XAxisVisible

说明

指定是否在对象中显示使用“XAxisIndex”引用的 X 轴。

运行系统中可进行读写访问

语法

Object.**XAxisVisible**[=BOOLEAN]

Object

必需项。具有以下格式的“ScreenItem”类型的对象：

- FunctionTrendControl

BOOLEAN

可选项。

TRUE，以显示所引用的 X 轴。

FALSE，不显示所引用的 X 轴。

参见

XAxisIndex (页 1448)

FunctionTrendControl (页 351)

YAxes

说明

在运行系统中无访问权限。

YAxisAdd

说明

创建新的 Y 轴。将使用“YAxisIndex”自动引用新创建的 Y 轴。

运行系统中可进行的访问：读和写

语法

Object.**YAxisAdd**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

STRING

必需项。使用"YAxisName"指定新的 Y 轴名称的值或常量。

参见

YAxisIndex (页 1461)

FunctionTrendControl (页 351)

YAxisAlignment

说明

指定使用"YAxisIndex"引用的 Y 轴的对齐方式。

运行系统中可进行读写访问

语法

Object.YAxisAlignment[=AxisAlignment]

Object

必选。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

AxisAlignment

可选项。用于指定对于趋势窗口的 Y 轴对齐方式的值或常量：

值	名称
0	左对齐
1	右对齐

参见

FunctionTrendControl (页 351)

YAxisAutoPrecisions

描述

指定是否自动指定使用"YAxisIndex"引用的 Y 轴的小数位数。

运行系统中可进行读写访问

语法

Object.**YAxisAutoPrecisions**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

BOOLEAN

可选项。

TRUE，自动确定小数位数。

FALSE，使用"YAxisPrecisions"值。

参见

FunctionTrendControl (页 351)

YAxisAutoRange

说明

指定是否自动计算使用"YAxisIndex"引用的 Y 轴的取值范围。

运行系统中可进行读写访问

语法

Object.**YAxisAutoRange**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

BOOLEAN

可选项。

TRUE，自动计算取值范围。

FALSE，通过"YAxisBeginValue"和"YAxisEndValue"指定取值范围。

参见

FunctionTrendControl (页 351)

YAxisBeginValue

说明

指定使用 YAxisIndex 引用的 Y 轴的取值范围下限。

另外将"YAxisAutoRange"设置为"FALSE"。

运行系统中可进行的访问： 读和写

语法

Object.**YAxisBeginValue**[=DOUBLE]

Object

必需项。 具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

DOUBLE

可选项。 用于指定所引用 Y 轴取值范围下限的值或常量。

参见

FunctionTrendControl (页 351)

YAxisColor

描述

指定使用"YAxisIndex"引用的 Y 轴的颜色。

也可将"YAxisInTrendColor"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**YAxisColor**[=Color]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

Color

可选项。用于指定引用的轴颜色的值或常量。

注释

可以使用"RGB"函数指定 RGB 格式（红、绿、蓝）的颜色。为此，可分别为三个 RGB 值输入相应的十进制值（范围为 0 至 255）。例如："红色"表示为：

RGB (255, 0, 0)

还可以使用 VBS 颜色常量，如 `vbRed` 和 `vbGreen`。

参见

FunctionTrendControl (页 351)

YAxisCount

描述

指定已组态的 Y 轴的数目。

运行系统中可进行读写访问

语法

Object.**YAxisCount**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

Int32

可选项。用于指定已组态的 Y 轴数的值或常量。

参见

FunctionTrendControl (页 351)

YAxisEndValue

描述

指定使用"YAxisIndex"引用 Y 轴的取值范围上限。

也可将"YAxisAutoRange"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**YAxisEndValue**[=DOUBLE]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

DOUBLE

可选项。用于指定引用的 Y 轴取值范围上限的值或常量。

参见

FunctionTrendControl (页 351)

YAxisExponentialFormat

描述

指定是否以指数计数法显示使用"YAxisIndex"引用的 Y 轴的值。

运行系统中可进行读写访问

语法

Object.**YAxisExponentialFormat**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

BOOLEAN

可选项。

TRUE，以指数计数法显示值。

FALSE，不以指数计数法显示值。

参见

FunctionTrendControl (页 351)

YAxisIndex

描述

引用 Y 轴。要访问 Y 轴的属性，需要设置""YAxisIndex""。

介于 0 至 (YAxisCount - 1) 之间的值为"YAxisIndex"的有效值。"YAxisCount"属性指定已组态 Y 轴的数目。

运行系统中可进行读写访问

语法

Object.**YAxisIndex**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

Int32

可选项。用于通过索引指定要编辑的 Y 轴的值或常量。

参见

FunctionTrendControl (页 351)

YAxisInTrendColor

描述

指定使用"YAxisIndex"引用的轴的颜色是否与趋势颜色相对应。

运行系统中可进行读写访问

语法

Object.**YAxisInTrendColor**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

BOOLEAN

可选项。

TRUE，以趋势颜色显示引用的轴。"YAxisColor"值无效。

FALSE，以"YAxisColor"指定的颜色显示引用的轴。

参见

YAxisIndex (页 1461)

FunctionTrendControl (页 351)

YAxisLabel

描述

指定使用"YAxisIndex"引用的 Y 轴的标签文本。

运行系统中可进行读写访问

语法

Object.**YAxisLabel**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

STRING

可选项。用于指定引用的 Y 轴标签文本的值或常量。

参见

FunctionTrendControl (页 351)

YAxisName

描述

指定使用"YAxisIndex"引用的 Y 轴的名称。

运行系统中可进行读写访问

语法

Object.**YAxisName**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

STRING

可选项。用于指定引用的 Y 轴名称的值或常量。

参见

FunctionTrendControl (页 351)

YAxisPrecisions

描述

指定使用"YAxisIndex"引用的 Y 轴的值显示的小数位数。

也可将"YAxisAutoPrecisions"设置为"FALSE"。

运行系统中可进行读写访问

语法

Object.**YAxisPrecisions**[=Int16]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

Int16

可选项。用于指定小数位数的值或常量。

参见

FunctionTrendControl (页 351)

YAxisRemove

描述

使用引用的 Y 轴名称可移除该 Y 轴。

运行系统中可进行读写访问

语法

Object.**YAxisRemove**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

STRING

可选项。用于指定待移除的引用的 Y 轴名称的值或常量。

参见

FunctionTrendControl (页 351)

YAxisRename

描述

指定使用"YAxisIndex"引用的 Y 轴的新名称。

运行系统中可进行读写访问

语法

Object.**YAxisRename**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

STRING

可选项。用于指定所选 Y 轴新名称的值或常量。

参见

FunctionTrendControl (页 351)

YAxisRepos

说明

对于多个 Y 轴，指定使用"YAxisIndex"引用的 Y 轴的位置。

如果已使用"YAxisRepos"更改 Y 轴的位置，则"YAxisIndex"将接收"YAxisRepos"的值。

运行系统中的访问权限

- RT Advanced：读取
- RT Professional：读和写

语法

Object.**YAxisRepos**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

Int32

可选项。用于指定所引用 Y 轴位置的值或常量。值范围为 0 到 (YAxisCount - 1)。超出该范围的值无效。

0：引用的 Y 轴放置于外部。

YAxisScalingType

说明

指定使用"YAxisIndex"引用的 Y 轴标定类型。

运行系统中的访问权限

- RT Advanced：读取
- RT Professional：读和写

语法

Object.**YAxisScalingType**[=AxisScalingType]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

AxisScalingType

可选项。用于指定所引用 Y 轴标定类型的值或常量。

值	VB 常量	名称
0	hmiBarScalingLinear	线性
1	hmiBarScalingLogarithmic	对数
2	hmiBarScalingNegativeLogarithmic	负对数

YAxisTrendWindow

描述

指定使用"YAxisIndex"引用的轴在其中显示的趋势图。

运行系统中的访问权限

- RT Advanced：读取
- RT Professional：读和写

语法

Object.**YAxisTrendWindow**[=STRING]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

STRING

可选项。用于指定趋势图名称的值或常量。

YAxisVisible

说明

指定是否在对象中显示使用"YAxisIndex"引用的 Y 轴。

运行系统中的访问权限

- RT Advanced：读取
- RT Professional：读和写

语法

Object.**YAxisVisible**[=BOOLEAN]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- FunctionTrendControl

BOOLEAN

可选项。

TRUE，以显示所引用的 Y 轴。

FALSE，不显示所引用的 Y 轴。

ZeroPoint

描述

以棒图高度百分比的形式指定零点位置。 零点位置还可位于显示区域的范围之外。

"ScalingType"属性必须设为"Auto"。

"ShowScale"属性必须设为"TRUE"。

运行系统中的访问权限：读和写

语法

Object.**ZeroPoint**[=Int32]

Object

必需项。"ScreenItem" 对象，且具有以下格式：

- Bar

Int32

可选项。用于以棒图高度百分比形式指定零点位置的值或常量。

参见

Bar (页 285)

ZoomFactor

说明

指定画面或画面窗口的缩放因子。

运行系统中可进行读写访问

语法

Object.**ZoomFactor**[=Int32]

Object

必需项。具有以下格式的"ScreenItem"类型的对象：

- ScreenWindow

Int32

可选项。用于指定显示嵌入式图形的缩放因子的值或常量。

参见

ScreenWindow (页 479)

1.5.6 方法

1.5.6.1 方法 A-G

Activate

描述

它仅对下列可操作的画面对象使用 Activate 方法时才有意义。在不能操作的画面对象(例如矩形)处输出出错消息。

- IOField
- Switch
- SymbolLibrary
- TrendView
- FunctionTrendControl
- HTMLBrowser
- Slider
- GraphicIOField
- SymbolicIOField
- Button
- AlarmControl
- UserView
- UserArchiveControl
- SmartClientView
- StatusForce

在 Runtime Advanced 和面板中，如果将相应方法应用于某个画面对象，则会激活永久区域。使用"BaseScreenName"属性激活尚未选择的画面。

语法

```
Expression.Activate()
```

表达式

必需项。输出，返回“Screen”或“ScreenItem”类型的对象。

参数

--

示例

```
Dim objO
Set objO = HMIRuntime.Screens("Screen_1").ScreenItems("IO-Field_1")
objO.Activate
```

参见

ScreenItem (页 234)

Screen (页 231)

ChannelDiagnose (页 305)

CheckBox (页 307)

Circle (页 312)

CircleSegment (页 316)

CircularArc (页 320)

Clock (页 323)

Connector (页 330)

DateTimeField (页 334)

DiskSpaceView (页 337)

Ellipse (页 340)

EllipseSegment (页 344)

EllipticalArc (页 348)

Gauge (页 366)

GraphicIOField (页 371)

GraphicView (页 375)

HTMLBrowser (页 379)

IOField (页 382)

Rectangle (页 467)

ScriptDiagnostics (页 482)

Switch (页 499)

SymbolicIOField (页 504)

SymbolLibrary (页 511)

TextField (页 527)

TrendView (页 546)

TubeArcObject (页 554)

TubeDoubleTeeObject (页 557)

TubePolyline (页 560)

TubeTeeObject (页 563)

UserView (页 581)

WindowSlider (页 586)

StatusForce (页 493)

SmartClientView (页 490)

Slider (页 485)

ScreenWindow (页 479)

RoundButton (页 471)

Polyline (页 448)

Polygon (页 444)

OptionGroup (页 435)

MultiLineEdit (页 399)

MediaPlayer (页 396)

Listbox (页 392)

Line (页 388)

Bar (页 285)

Button (页 296)

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

FunctionTrendControl (页 351)

AlarmView (页 274)

AlarmControl (页 255)

ActivateDynamic

描述

在运行期间，动态激活触发器和属性的指定周期。它需要在属性中具有一个 VB 脚本并需要一个设置为“按需”的触发器。每次激活触发器时，都可以使用不同的激活周期。

语法

```
Expression.ActivateDynamic (ByVAl Property name As String, ByVal
Cycle name As Cycle)
```

Expression

必需项。返回“ScreenItem”类型对象的表达式。

参数

参数	描述
Property name	与触发器相关的属性。使用属性的 VBS 名称。
Cycle name	激活周期的名称，例如“CycleTime1s”。

参见

ChannelDiagnose (页 305)

CheckBox (页 307)

Circle (页 312)

CircleSegment (页 316)

CircularArc (页 320)

Clock (页 323)

Connector (页 330)

DiskSpaceView (页 337)

Ellipse (页 340)

EllipseSegment (页 344)

EllipticalArc (页 348)

Gauge (页 366)

GraphicIOField (页 371)

GraphicView (页 375)

HTMLBrowser (页 379)

IOField (页 382)

Rectangle (页 467)

ScriptDiagnostics (页 482)

SymbolicIOField (页 504)

SymbolLibrary (页 511)

TextField (页 527)

TubeArcObject (页 554)

TubeDoubleTeeObject (页 557)

TubePolyline (页 560)

TubeTeeObject (页 563)

UserView (页 581)

WindowSlider (页 586)

Slider (页 485)

ScreenWindow (页 479)

RoundButton (页 471)

Polyline (页 448)

Polygon (页 444)

OptionGroup (页 435)

MultiLineEdit (页 399)

MediaPlayer (页 396)

Listbox (页 392)

Line (页 388)

Bar (页 285)

Button (页 296)

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

FunctionTrendControl (页 351)

AlarmControl (页 255)

Add

TagSet 对象说明

向列表中添加变量。 通过名称或引用可将变量添加到变量对象中。

语法

```
Expression.Add [Tag]
```

表达式

必需。 返回"TagSet"类型对象的表达式。

参数

VARIANT

参数	说明
变量	WinCC 变量的名称或对添加到列表中的变量对象的引用。

示例：

在下列示例中，生成 TagSet 对象以及添加变量。

```
'VBS170
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
```

按如下步骤也可以添加变量对象。

```
'VBS171
Dim Tag
Set Tag = HMIRuntime.Tags("Motor2")
Dim group2
Set group2 = HMIRuntime.Tags.CreateTagSet
group2.Add Tag
```

DataSet 对象说明

向列表添加值或对象引用。

说明

Data Set 对象不支持任何类。
DataSet 列表不能包含 Screen、Screens、ScreenItem、ScreenItems、Tag 和 TagSet 类型对象。
对于对象引用，必须确定对象可以多次读取。

语法

```
Expression.Add [vtName], [vtUserData]
```

表达式

必需。 返回"DataSet"类型对象的表达式。

参数

VARIANT

参数	说明
vtName	按名称将数值或变量添加到列表中。
vtUserData	添加到列表中的值。

示例：

在该示例中，值包含在 DataSet 列表中。

```
'VBS172
HMIRuntime.DataSet.Add "Motor1",23
```

参见

DataSet（列表） (页 221)

OnlineTrendControl (页 418)

AttachDB 方法

描述

执行控件的“连接备份”键功能。

语法

```
Expression.AttachDB()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

FunctionTrendControl (页 351)

AlarmControl (页 255)

CalculateStatistic

描述

执行 f(t) 趋势视图和表格视图的"计算统计数据"键功能。

语法

```
Expression.CalculateStatistic()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

CopyRows

描述

执行控件的“复制行”键功能。

语法

```
Expression.CopyRows()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

用户归档控件 (页 566)

AlarmControl (页 255)

Create

描述

创建新 Alarm 对象。

语法

```
Expression.Create (VARIANT vtApplication)
```

Expression

必需项。用于返回 “报警”(Alarm) 类型对象的表达式。

参数

VARIANT

参数	描述
vtApplication	报警对象的名称（可选）

参见

OnlineTrendControl (页 418)

CreateTagSet

描述

创建新 TagSet 对象。 此对象可用于优化的多变量访问。

语法

```
Expression.CreateTagSet()
```

表达式

必需。用于返回"TagSet"类型对象的表达式。

参数

VARIANT

示例：

以下示例说明了如何创建 TagSet 对象。

```
'VBS168
'Build a Reference to the TagSet Object
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
```

参见

OnlineTrendControl (页 418)

CutRows

描述

执行配方视图的“剪切行”键功能。

语法

```
Expression.CutRows()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

OnlineTrendControl (页 418)

用户归档控件 (页 566)

DeactivateDynamic

描述

在运行期间，取消激活用于指定属性的“ActivateDynamic”方法的触发器。

语法

```
Expression.DeactivateDynamic(ByVal Property name As String)
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

参数	描述
Property name	属性，已将其触发器禁用。使用属性的 VBS 名称。

参见

ChannelDiagnose (页 305)

CheckBox (页 307)

Circle (页 312)

CircleSegment (页 316)

CircularArc (页 320)

Clock (页 323)

Connector (页 330)

DiskSpaceView (页 337)

Ellipse (页 340)

EllipseSegment (页 344)

EllipticalArc (页 348)

Gauge (页 366)

GraphicIOField (页 371)

GraphicView (页 375)

HTMLBrowser (页 379)

IOField (页 382)

Rectangle (页 467)

ScriptDiagnostics (页 482)

SymbolicIOField (页 504)

SymbolLibrary (页 511)

TextField (页 527)

TubeArcObject (页 554)

TubeDoubleTeeObject (页 557)

TubePolyline (页 560)

TubeTeeObject (页 563)

UserView (页 581)

WindowSlider (页 586)

Slider (页 485)

ScreenWindow (页 479)

RoundButton (页 471)

Polyline (页 448)

Polygon (页 444)

OptionGroup (页 435)

MultiLineEdit (页 399)

MediaPlayer (页 396)

Listbox (页 392)

Line (页 388)

Bar (页 285)

Button (页 296)

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

FunctionTrendControl (页 351)

AlarmControl (页 255)

DeleteRows

描述

执行配方视图的“删除行”键功能。

语法

```
Expression.DeleteRows()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

OnlineTrendControl (页 418)

用户归档控件 (页 566)

DetachDB

描述

执行控件的“断开备份”键功能。

语法

```
Expression.DetachDB()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

FunctionTrendControl (页 351)

AlarmControl (页 255)

Edit

描述

执行表格视图的“编辑”键功能。

语法

```
Expression.Edit()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

AlarmControl (页 255)

导出

描述

执行控件的“导出日志”或“导出数据”键功能。

语法

```
Expression.Export()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

FunctionTrendControl (页 351)

AlarmControl (页 255)

GetColumn

描述

以“ICCAxUAColumn”类型的形式返回配方视图中具有指定名称或索引的列对象。

语法

```
Ausdruck.GetColumn(ByVal vIndex As Variant)
```

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

VARIANT

参数	描述
vIndex	配方视图中列的索引或名称

示例

```
'VBS312
Dim ctrl
Dim objColumn
Set ctrl = ScreenItems("RecipeControl")
Set objColumn = ctrl.GetColumn("Field1")
objColumn.Length = 30
Set objColumn = ctrl.GetColumn(3)
objColumn.Align = 2
```

说明

当使用“Get...”方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于“Column”列表，应该写为“objColumn.Align”而不是“objColumn.ColumnAlign”。

参见

OnlineTrendControl (页 418)

用户归档控件 (页 566)

GetColumnCollection

描述

以“ICCAxCollection”类型的形式返回配方视图中所有列对象的列表。

语法

```
Ausdruck.GetColumnCollection()
```

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

--

ICCAxCollection 的特性和函数

下列属性可用于 ICCAxCollection：

- Count
- Item

下列函数可用于 ICCAxCollection：

- AddItem(vName) As Object
- RemoveItem(vIndex)

示例

```
'VBS313
Dim ctrl
Dim coll
Dim field
Set ctrl = ScreenItems("RecipeControl")
Set coll = ctrl.GetColumnCollection
HMIRuntime.Trace "Number of fields:" & coll.Count & vbCrLf
For Each field In coll
 HMIRuntime.Trace field.Name & vbCrLf
 HMIRuntime.Trace field.Type & vbCrLf
 HMIRuntime.Trace field.Length & vbCrLf
 HMIRuntime.Trace field.Caption & vbCrLf
Next
```

参见

OnlineTrendControl (页 418)

用户归档控件 (页 566)

GetHistlistColumnCollection

描述

以“ICCAxCollection”类型的形式返回消息视图统计列表中所有列对象的列表。

语法

```
Expression.GetHitlisteColumnCollection()
```

表达式

必需。 返回"ScreenItem"类型对象的表达式。

参数

--

ICCAxCollection 的特性和函数

下列属性可用于 ICCAxCollection：

- Count
- Item

下列函数可用于 ICCAxCollection：

- AddItem(vName) As Object
- RemoveItem(vIndex)

示例

```
'VBS315
Dim ctrl
Dim coll
Dim hitlistcol
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetHitlistColumnCollection
HMIRuntime.Trace "Number of hitlist columns:" & coll.Count & vbCrLf
For Each hitlistcol In coll
 HMIRuntime.Trace hitlistcol.Index & vbCrLf
 HMIRuntime.Trace hitlistcol.Name & vbCrLf
 HMIRuntime.Trace hitlistcol.Sort & vbCrLf
 HMIRuntime.Trace hitlistcol.SortIndex & vbCrLf
Next
```

参见

OnlineTrendControl (页 418)

AlarmControl (页 255)

GetHitlistColumn

描述

以“ICCAxMessageColumn”类型的形式返回消息视图统计列表中具有指定名称或索引的列对象。

语法

```
Expression.GetHitlistColumn(ByVal vIndex As Variant)
```

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

VARIANT

参数	描述
vIndex	命中列表列的索引或名称。

示例

```
'VBS314
Dim ctrl
Dim objHitlistColumn
Set ctrl = ScreenItems("AlarmControl")
Set objHitlistColumn = ctrl.GetHitlistColumn("Date")
objHitlistColumn.Sort = 2
Set objHitlistColumn = ctrl.GetHitlistColumn("AverageComeGo")
objHitlistColumn.Visible = FALSE
```

说明

当使用“Get...”方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于“HitlistColumn”列表，应该写为“objHitlistColumn.Visible”而不是“objHitlistColumn.HitlistColumnVisible”。

参见

OnlineTrendControl (页 418)

AlarmControl (页 255)

GetMessageBlock

描述

以“ICCAxMessageBlock”类型的形式返回消息视图中具有指定名称或索引的消息块。

语法

```
Expression.GetMessageBlock(ByVal vIndex As Variant)
```

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

VARIANT

参数	描述
vIndex	消息块的索引或名称。

示例

```
'VBS316
Dim ctrl
Dim objMsgBlock
Set ctrl = ScreenItems("AlarmControl")
Set objMsgBlock = ctrl.GetMessageBlock("Date")
objMsgBlock.Align = 2
Set objMsgBlock = ctrl.GetMessageBlock("Number")
objMsgBlock.LeadingZeros = 4
```

说明

当使用"Get..."方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于"MessageBlock"列表，应该写为"objMsgBlock.Align"而不是"objMsgBlock.MessageBlockAlign"。

参见

OnlineTrendControl (页 418)

AlarmControl (页 255)

GetMessageBlockCollection

描述

以"ICCAxCollection"类型的形式返回消息视图中所有消息块对象的列表。

语法

```
Expression.GetMessageBlockCollection()
```

表达式

必需。 返回"ScreenItem"类型对象的表达式。

参数

- -

ICCAxCollection 的特性和函数

下列属性可用于 ICCAxCollection：

- Count
- Item

下列函数可用于 ICCAxCollection：

- AddItem(vName) As Object
- RemoveItem(vIndex)

示例

```
'VBS317
Dim ctrl
Dim coll
Dim msgblock
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetMessageBlockCollection
For Each msgblock In coll
 msgblock.Align = 1
 msgblock.Length = 12
 msgblock.Selected = TRUE
Next
```

说明

当使用“Get...”方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于“MessageBlock”列表，应该写为“msgblock.Align”而不是“msgblock.MessageBlockAlign”。

参见

OnlineTrendControl (页 418)

AlarmControl (页 255)

GetMessageColumn

描述

以“ICCAxMessageColumn”类型的形式返回消息视图中具有指定名称或索引的列对象。

语法

```
Expression.GetMessageColumn(ByVal vIndex As Variant)
```

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

VARIANT

参数	描述
vIndex	消息列表中列的索引或名称。

示例

```
'VBS318
Dim ctrl
Dim objMessColumn
Set ctrl = ScreenItems("AlarmControl")
Set objMessColumn = ctrl.GetMessageColumn("Date")
objMessColumn.Visible = FALSE
Set objMessColumn = ctrl.GetMessageColumn("Number")
objMessColumn.Sort = 1
```

说明

当使用“Get...”方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于“MessageColumn”列表，应该写为“objMessColumn.Visible”而不是“objMessColumn.MessageColumnVisible”。

参见

OnlineTrendControl (页 418)

AlarmControl (页 255)

GetMessageColumnCollection

描述

以“ICCAxCollection”类型的形式返回消息视图中所有列对象的列表。

语法

```
Expression.GetMessageColumnCollection()
```

表达式

必需。 返回"ScreenItem"类型对象的表达式。

参数

--

ICCAxCollection 的特性和函数

下列属性可用于 ICCAxCollection：

- Count
- Item

下列函数可用于 ICCAxCollection：

- AddItem(vName) As Object
- RemoveItem(vIndex)

示例

```
'VBS319
Dim ctrl
Dim coll
Dim msgcol
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetMessageColumnCollection
HMIRuntime.Trace "Number of message columns:" & coll.Count & vbCrLf
For Each msgcol In coll
 HMIRuntime.Trace msgcol.Index & vbCrLf
 HMIRuntime.Trace msgcol.Name & vbCrLf
 HMIRuntime.Trace msgcol.Sort & vbCrLf
 HMIRuntime.Trace msgcol.SortIndex & vbCrLf
Next
```

参见

OnlineTrendControl (页 418)

AlarmControl (页 255)

GetOperatorMessage

描述

以“ICCAxOperatorMessage”类型的形式返回消息视图中具有指定名称或索引的操作员消息对象。

语法

```
Expression.GetOperatorMessage(ByVal vIndex As Variant)
```

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

VARIANT

参数	描述
vIndex	操作员消息的索引或名称

示例

```
'VBS320
Dim ctrl
Dim objOpMess
Set ctrl = ScreenItems("AlarmControl")
Set objOpMess = ctrl.GetOperatorMessage(0)
objOpMess.Source1 = "Number"
objOpMess.SourceType1 = 1
```

说明

当使用“Get...”方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于“OperatorMessage”列表，应该写为“objOpMess.Source1”而不是“objOpMess.OperatorMessageSource1”。

参见

OnlineTrendControl (页 418)

AlarmControl (页 255)

GetOperatorMessageCollection

描述

以"ICCAxCollection"类型的形式返回消息视图中所有操作员消息对象的列表。

语法

```
Expression.GetOperatorMessageCollection()
```

表达式

必需。 返回"ScreenItem"类型对象的表达式。

参数

--

ICCAxCollection 的特性和函数

下列属性可用于 ICCAxCollection：

- Count
- Item

下列函数可用于 ICCAxCollection：

- AddItem(vName) As Object
- RemoveItem(vIndex)

示例

```
'VBS321
Dim ctrl
Dim coll
Dim opmsg
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetOperatorMessageCollection
For Each opmsg In coll
 HMIRuntime.Trace opmsg.Index & vbCrLf
 HMIRuntime.Trace opmsg.Name & vbCrLf
 HMIRuntime.Trace opmsg.Number & vbCrLf
 HMIRuntime.Trace opmsg.Selected & vbCrLf
Next
```

参见

OnlineTrendControl (页 418)

AlarmControl (页 255)

GetRow

描述

以“ICCAxDataRow”类型的形式返回基于表格的控件中的行编号所定义的行对象。

语法

表达式.GetRow(ByVal IRow As Long)

表达式

必选。返回“ScreenItem”类型对象的表达式。

参数

Long

参数	说明
IRow	控件的所需行数。

示例

```
'VBS356
Dim coll
Dim ctrl
Dim lIndex
Dim lCellIndex
Set ctrl = ScreenItems("UAControl")
Set coll = ctrl.GetRowCollection
'enumerate and trace out row numbers
For lIndex = 1 To coll.Count
 HMIRuntime.trace "Row: " & (ctrl.GetRow(lIndex).RowNumber) & "  "
 'enumerate and trace out column titles and cell texts
 For lCellIndex = 1 To ctrl.GetRow(lIndex).CellCount
  HMIRuntime.trace ctrl.GetRow(0).CellText(lCellIndex) & "  "
  HMIRuntime.trace ctrl.GetRow(lIndex).CellText(lCellIndex) & "  "
 Next
 HMIRuntime.trace vbNewLine
Next
```

说明

当使用"Get..."方法通过控件对象列表而不是控件对象访问属性时，必须删除属性的列表名称前缀。

例如，对于"Row"列表，应该写为"objRow.CellCount"而不是"objRow.RowCellCount"。

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

AlarmControl (页 255)

GetRowCollection

描述

以类型"ICCAxDataRowCollection"形式返回基于表控件的所有行对象的列表。

语法

```
Expression.GetRowCollection()
```

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

--

ICCAxDataRowCollection 的属性

ICCAxDataRowCollection 涉及运行系统数据。 数据为只读数据。 无法添加和编辑数据。

下列属性可用于 ICCAxDataRowCollection：

- Count - 确定集合中的行数。
- Item - 通过行编号访问集合内的各个行。 编号的范围是 1 到 Count。 将返回 Row 对象。

示例

```
'VBS357
Dim ctrl
Dim coll
Dim lIndex
Dim lCellIndex
Set ctrl = ScreenItems("AlarmControl")
Set coll = ctrl.GetRowCollection
HMIRuntime.Trace "Number of message rows:" & coll.Count & vbCrLf
'列举并跟踪行号
For lIndex = 1 To coll.Count
 HMIRuntime.Trace "Row: " & (ctrl.GetRow(lIndex).RowNumber) & "  "
 '列举并跟踪列标题和单元格文本
 For lCellIndex = 1 To ctrl.GetRow(lIndex).CellCount
  HMIRuntime.Trace ctrl.GetMessageColumn(lCellIndex -1).Name & " "
  HMIRuntime.Trace ctrl.GetRow(lIndex).CellText(lCellIndex) & "  "
 Next
 HMIRuntime.Trace vbNewLine
Next
```

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

AlarmControl (页 255)

GetRulerBlock

描述

以"ICCAxRulerBlock"类型的形式返回评估表中具有指定名称或索引的块对象。

语法

```
Expression.GetRulerBlock(ByVal vIndex As Variant)
```

表达式

必需。 返回"ScreenItem"类型对象的表达式。

参数

VARIANT

参数	描述
vIndex	评估表中块的索引或名称

示例

```
'VBS322
Dim ctrl
Dim objRulerBlock
Set ctrl = ScreenItems("RulerControl")
Set objRulerBlock = ctrl.GetRulerBlock(0)
objRulerBlock.Caption = "RulerBlock1"
Set objRulerBlock = ctrl.GetRulerBlock("Name")
objRulerBlock.Length = 10
```

说明

当使用"Get..."方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于"RulerBlock"列表，应该写为"objRulerBlock.Caption"而不是"objRulerBlock.BlockCaption"。

参见

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

GetRulerBlockCollection

描述

以"ICCAxCollection"类型的形式返回评估表中所有块对象的列表。

语法

```
Expression.GetRulerBlockCollection()
```

表达式

必需。 返回"ScreenItem"类型对象的表达式。

参数

- -

ICCAxCollection 的特性和函数

下列属性可用于 ICCAxCollection：

- Count
- Item

下列函数可用于 ICCAxCollection：

- AddItem(vName) As Object
- RemoveItem(vIndex)

示例

```
'VBS323
Dim ctrl
Dim coll
Dim rulerblock
Set ctrl = ScreenItems("RulerControl")
Set coll = ctrl.GetRulerBlockCollection
For Each rulerblock In coll
 rulerblock.Align = 1
 rulerblock.Length = 12
Next
```

说明

当使用“Get...”方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于“RulerBlock”列表，应该写为“rulerblock.Align”而不是“rulerblock.RulerBlockAlign”。

参见

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

GetRulerColumn

描述

以“ICCAxRulerColumn”类型的形式返回评估表中具有指定名称或索引的列对象。

语法

```
Expression.GetRulerColumn(ByVal vIndex As Variant)
```

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

VARIANT

参数	描述
vIndex	评估表中列的索引或名称

示例

```
'VBS324
Dim ctrl
Dim objRulercol
Set ctrl = ScreenItems("RulerControl")
Set objRulercol = ctrl.GetRulerColumn("Name")
objRulercol.Sort = 0
Set objRulercol = ctrl.GetRulerColumn("ValueY")
objRulercol.Visible = FALSE
```

说明

当使用“Get...”方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于“RulerColumn”列表，应该写为“objRulercol.Visible”而不是“objRulercol.ColumnVisible”。

参见

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

GetRulerColumnCollection

描述

以“ICCAxCollection”类型的形式返回评估表中所有列对象的列表。

语法

```
Expression.GetRulerColumnCollection()
```

表达式

必需。 返回"ScreenItem"类型对象的表达式。

参数

--

ICCAxCollection 的特性和函数

下列属性可用于 ICCAxCollection：

- Count
- Item

下列函数可用于 ICCAxCollection：

- AddItem(vName) As Object
- RemoveItem(vIndex)

示例

```
'VBS325
Dim ctrl
Dim coll
Dim rulercol
Set ctrl = ScreenItems("RulerControl")
Set coll = ctrl.GetRulerColumnCollection
HMIRuntime.Trace "Number of ruler columns:" & coll.Count & vbCrLf
For Each rulercol In coll
 HMIRuntime.Trace rulercol.Index & vbCrLf
 HMIRuntime.Trace rulercol.Name & vbCrLf
 HMIRuntime.Trace rulercol.Sort & vbCrLf
 HMIRuntime.Trace rulercol.SortIndex & vbCrLf
Next
```

参见

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

GetRulerData

说明

返回标尺位置处被调用趋势的值。

语法

```
表达式.GetRulerData(ByVal RulerIndex As Long, pvValue As Variant,
Optional pvTimeStamp As Variant, Optional pvFlags As Varian) Long
```

表达式

必需。 返回“Trend”类型对象的表达式。

参数

参数	说明
RulerIndex	0 = 标尺
pvValue	X 轴的值
pvTimeStamp	时间或 Y 轴的值
pvFlags	Qualitycode

示例

```
'VBS326
Dim ctrl
Dim objTrend
Dim objIOField1
Dim objIOField2
Dim rulvalue
Dim rultime
Set ctrl = ScreenItems( "Control1" )
Set objTrend = ctrl.GetTrend( "Trend 1" )
Set objIOField1 = ScreenItems( "I/O Field1" )
Set objIOField2 = ScreenItems( "I/O Field2" )
objTrend.GetRulerData 0, rulvalue, rultime
objIOField1.OutputValue = rulvalue
objIOField2.OutputValue = rultime
```

参见

OnlineTrendControl (页 418)

GetSelectedRow

描述

以"ICCAxDataRow"类型的形式返回基于表格的控件的所选行对象。

语法

```
Expression.GetSelectedRow()
```

表达式

必需。返回"ScreenItem"类型对象的表达式。

参数

--

示例

```
'VBS358
Dim ctrl
 Dim lCellIndex
 Dim lCellCount
 Dim headingRow
 Dim selectedRow
 Set ctrl = ScreenItems("TableControl")
 Set headingRow = ctrl.GetRow(0)
 Set selectedRow = ctrl.GetSelectedRow
 lCellCount = headingRow.CellCount
 '列举并跟踪列标题和单元格文本
 For lCellIndex = 1 To lCellCount
  HMIRuntime.trace headingRow.CellText(lCellIndex) & ":  "
  HMIRuntime.trace selectedRow.CellText(lCellIndex)
    HMIRuntime.trace vbNewLine
 Next
```

说明

当使用“Get...”方法通过控件对象列表而不是控件对象访问属性时，必须删除属性的列表名称前缀。

例如，对于“Row”列表，应该写为“objRow.CellCount”而不是“objRow.RowCellCount”。

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

AlarmControl (页 255)

GetSelectedRows

描述

以“ICCAxDataRow”类型的形式返回用于多项选择的基于表格的控件的所选行对象。

语法

```
Expression.GetSelectedRows()
```

表达式

必需。返回“ScreenItem”类型对象的表达式。

参数

- -

示例

```
'VBS359
Dim ctrl
 Dim lCellIndex
 Dim lCellCount
 Dim lRowIndex
 Dim lRowCount
 Dim headingRow
 Dim selectedRow
 Dim selectedRows
 Set ctrl = ScreenItems("TableControl")
 Set headingRow = ctrl.GetRow(0)
 Set selectedRows = ctrl.GetSelectedRows
 lCellCount = headingRow.CellCount
 lRowCount = selectedRows.Count
 'enumerate selected rows
 For lRowIndex = 1 To lRowCount
  Set selectedRow = selectedRows(lRowIndex)
  HMIRuntime.Trace "Row number: " & CStr(lRowIndex) & vbNewLine
  '列举并跟踪列标题和单元格文本
  For lCellIndex = 1 To lCellCount
   HMIRuntime.trace headingRow.CellText(lCellIndex) & ":  "
   HMIRuntime.trace selectedRow.CellText(lCellIndex)
     HMIRuntime.trace vbNewLine
  Next
 Next
```

说明

当使用“Get...”方法通过控件对象列表而不是控件对象访问属性时，必须删除属性的列表名称前缀。

例如，对于“Row”列表，应该写为“objRow.CellCount”而不是“objRow.RowCellCount”。

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

AlarmControl (页 255)

GetStatisticAreaColumn

描述

以“ICCAxRulerColumn”类型的形式返回评估窗口的统计区域窗口中具有指定名称或索引的列对象。

语法

```
Ausdruck.GetStatisticAreaColumn(ByVal vIndex As Variant)
```

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

VARIANT

参数	描述
vIndex	统计区域窗口的列的索引或名称。

示例

```
'VBS327
Dim ctrl
Dim objStatAreaCol
Set ctrl = ScreenItems("RulerControl")
Set objStatAreaCol = ctrl.GetStatisticAreaColumn("DatasourceY")
objStatAreaCol.Visible = FALSE
Set objStatAreaCol = ctrl.GetStatisticAreaColumn("ValueY(LL)")
objStatAreaCol.Sort = 1
```

说明

当使用“Get...”方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于“StatisticAreaColumn”列表，应该写为“objStatAreaCol.Visible”而不是“objStatAreaCol.ColumnVisible”。

参见

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

GetStatisticAreaColumnCollection

描述

以“ICCAxCollection”类型的形式返回评估表的统计区域窗口中所有列对象的列表。

语法

```
Ausdruck.GetStatisticAreaColumnCollection()
```

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

- -

ICCAxCollection 的特性和函数

下列属性可用于 ICCAxCollection：

- Count
- Item

下列函数可用于 ICCAxCollection：

- AddItem(vName) As Object
- RemoveItem(vIndex)

示例

```
'VBS328
Dim ctrl
Dim coll
Dim statcol
Set ctrl = ScreenItems("RulerControl")
Set coll = ctrl.GetStatisticAreaColumnCollection
HMIRuntime.Trace "Number of statistic Area columns:" & coll.Count & vbCrLf
For Each statcol In coll
 HMIRuntime.Trace statcol.Index & vbCrLf
 HMIRuntime.Trace statcol.Name & vbCrLf
 HMIRuntime.Trace statcol.Sort & vbCrLf
 HMIRuntime.Trace statcol.SortIndex & vbCrLf
Next
```

参见

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

GetStatisticResultColumn

描述

以“ICCAxRulerColumn”类型的形式返回评估窗口中具有指定名称或索引的统计窗口的列对象。

语法

```
Ausdruck.GetStatisticResultColumn(ByVal vIndex As Variant)
```

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

VARIANT

参数	描述
vIndex	统计窗口的列的索引或名称。

示例

```
'VBS329
Dim ctrl
Dim objStatResCol
Set ctrl = ScreenItems("RulerControl")
Set objStatResCol = ctrl.GetStatisticResultColumn("MaxValue")
objStatResCol.Visible = FALSE
Set objStatResCol = ctrl.GetStatisticResultColumn("Average")
objStatResCol.Sort = 2
```

说明

当使用"Get..."方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于"StatisticResultColumn"列表，应该写为"objStatResCol.Visible"而不是"objStatResCol.ColumnVisible"。

参见

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

GetStatisticResultColumnCollection

描述

以"ICCAxCollection"类型的形式返回评估表的统计窗口中所有列对象的列表。

语法

```
Ausdruck.GetStatisticResultColumnCollection()
```

表达式

必需。 返回"ScreenItem"类型对象的表达式。

参数

--

ICCAxCollection 的特性和函数

下列属性可用于 ICCAxCollection：

- Count
- Item

下列函数可用于 ICCAxCollection：

- AddItem(vName) As Object
- RemoveItem(vIndex)

示例

```
'VBS330
Dim ctrl
Dim coll
Dim statcol
Set ctrl = ScreenItems("RulerControl")
Set coll = ctrl.GetStatisticResultColumnCollection
HMIRuntime.Trace "Number of statistic result columns:" & coll.Count & vbCrLf
For Each statcol In coll
 HMIRuntime.Trace statcol.Index & vbCrLf
 HMIRuntime.Trace statcol.Name & vbCrLf
 HMIRuntime.Trace statcol.Sort & vbCrLf
 HMIRuntime.Trace statcol.SortIndex & vbCrLf
Next
```

参见

OnlineTrendControl (页 418)

TrendRulerControl (页 532)

GetStatusbarElement

描述

以类型"ICCAxStatusbarElement"形式返回指定为名称或索引的控制状态栏的元素。

语法

```
Ausdruck.GetStatusbarElement(ByVal vIndex As Variant)
```

表达式

必选。返回“ScreenItem”类型对象的表达式。

参数

VARIANT

参数	描述
vIndex	状态栏元素的索引或名称。

示例

```
'VBS331
Dim ctrl
Dim objStatusBar
Set ctrl = ScreenItems( "Control1" )
Set objStatusBar = ctrl.GetStatusbarElement(1)
objStatusBar.Visible = FALSE
Set objStatusBar = ctrl.GetStatusbarElement(3)
objStatusBar.Width = 10
```

说明

当使用“Get...”方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于“StatusbarElement”列表，应该写为“objStatusBar.Visible”而不是“objStatusBar.StatusbarElementVisible”。

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

FunctionTrendControl (页 351)

AlarmControl (页 255)

GetStatusbarElementCollection

描述

以类型“ICCAxCollection”形式返回控件的所有状态栏元素的列表。

语法

```
Ausdruck.GetStatusbarElementCollection()
```

表达式

必选。返回“ScreenItem”类型对象的表达式。

参数

- -

ICCAxCollection 的特性和函数

下列属性可用于 ICCAxCollection：

- Count
- Item

下列函数可用于 ICCAxCollection：

- AddItem(vName) As Object
- RemoveItem(vIndex)

示例

```
'VBS332
Dim ctrl
Dim coll
Dim statelement
Set ctrl = ScreenItems.Item("Control1")
Set coll = ctrl.GetStatusbarElementCollection
HMIRuntime.Trace "Number of statusbar elements:" & coll.Count & vbCrLf
For Each statelement In coll
 HMIRuntime.Trace statelement.Name & vbCrLf
 HMIRuntime.Trace statelement.Width & vbCrLf
 HMIRuntime.Trace statelement.Text & vbCrLf
Next
```

说明

当使用"Get..."方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于"StatusbarElement"列表，应该写为"statelement.Name"而不是"statelement.StatusbarElementName"。

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

FunctionTrendControl (页 351)

AlarmControl (页 255)

GetTimeAxis

说明

以"ICCAxTimeAxis"类型的形式返回 f(t) 趋势视图中具有指定名称或索引的时间对象。

语法

表达式.GetTimeAxis(ByVal vIndex As Variant)

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

VARIANT

参数	说明
vIndex	时间轴的索引或名称。

示例

```
'VBS333
Dim ctrl
Dim objTimeAxis
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTimeAxis = ctrl.GetTimeAxis(1)
objTimeAxis.Visible = FALSE
Set objTimeAxis = ctrl.GetTimeAxis("axis 2")
objTimeAxis.Label = "Time axis 2"
objTimeAxis.DateFormat = "dd.MM.yy"
objTimeAxis.TimeFormat = "HH:mm:ss.ms"
objTimeAxis.RangeType = 2
'The format to be used for date and time entries depends on the
'regional settings and language options in the operating system.
objTimeAxis.BeginTime = "06.04.2010 9:33:18"
'objTimeAxis.BeginTime = "04/06/2010 9:33:18"
objTimeAxis.MeasurePoints = 100
```

说明

当使用“Get...”方法通过控件对象列表而不是控件对象访问属性时，必须删除属性的列表名称前缀。

例如，对于“TimeAxis”列表，应该写为“objTimeAx.Visible”而不是“objTimeAx.TimeAxisVisible”。

参见

OnlineTrendControl (页 418)

GetTimeAxisCollection

说明

以“ICCAxCollection”类型的形式返回 f(t) 趋势视图中所有时间对象的列表。

语法

表达式.GetTimeAxisCollection()

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

- -

ICCAxCollection 的特性和函数

下列属性可用于 ICCAxCollection：

- Count
- Item

下列函数可用于 ICCAxCollection：

- AddItem(vName) As Object
- RemoveItem(vIndex)

示例

```
'VBS334
Dim ctrl
Dim objTrendWnd
Dim objTimeAxis1
Dim objTimeAxis2
Dim objTrend
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindowCollection.AddItem("myWindow")
Set objTimeAxis1 = ctrl.GetTimeAxisCollection.AddItem("TimeAxis2010")
Set objTimeAxis2 = ctrl.GetTimeAxisCollection.AddItem("TimeAxis2011")
objTimeAxis1.TrendWindow = objTrendWnd.Name
objTimeAxis1.Label = "2010"
objTimeAxis1.RangeType = 1
'The format to be used for date and time entries depends on the
'regional settings and language options in the operating system.
objTimeAxis1.BeginTime = "01.01.2010 0:00:00"
'objTimeAxis1.BeginTime = "01/01/2010 0:00:00"
objTimeAxis1.EndTime = "31.12.2010 11:59:59"
'objTimeAxis1.EndTime = "12/31/2010 11:59:59"
objTimeAxis2.TrendWindow = objTrendWnd.Name
objTimeAxis2.Label = "2011"
objTimeAxis2.RangeType = 1
objTimeAxis2.BeginTime = "01.01.2011 0:00:00"
'objTimeAxis2.BeginTime = "01/01/2011 0:00:00"
objTimeAxis2.EndTime = "31.12.2011 11:59:59"
'objTimeAxis2.EndTime = "12/31/2011 11:59:59"
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend1")
objTrend.TrendWindow = objTrendWnd.Name
objTrend.TimeAxis = objTimeAxis1.Name
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend2")
objTrend.TrendWindow = objTrendWnd.Name
objTrend.TimeAxis = objTimeAxis2.Name
```

说明

当使用“Get...”方法通过控件对象列表而不是控件对象访问属性时，必须删除属性的列表名称前缀。

例如，对于“TimeAxis”列表，应该写为“objTimeAxis1.Label”而不是“objTimeAxis1.TimeAxisLabel”。

参见

OnlineTrendControl (页 418)

GetTimeColumn

描述

以“ICCAxTimeColumn”类型的形式返回表视图中具有指定名称或索引的时间列对象。

语法

```
Ausdruck.GetTimeColumn(ByVal vIndex As Variant)
```

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

VARIANT

参数	描述
vIndex	时间列的索引或名称。

示例

```
'VBS335
Dim ctrl
Dim objTimeCol
Set ctrl = ScreenItems("TableControl")
Set objTimeCol = ctrl.GetTimeColumn("Timecolumn1")
objTimeCol.ShowDate = FALSE
Set objTimeCol = ctrl.GetTimeColumn("Timecolumn2")
objTimeCol.Visible = FALSE
```

说明

当使用“Get...”方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于“TimeColumn”列表，应该写为“objTimeColumn.ShowDate”而不是“objTimeColumn.TimeColumnShowDate”。

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

GetTimeColumnCollection

说明

以“ICCAxCollection”类型的形式返回表视图中所有时间列对象的列表。

语法

表达式.GetTimeColumnCollection()

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

- -

ICCAxCollection 的特性和函数

下列属性可用于 ICCAxCollection：

- Count
- Item

下列函数可用于 ICCAxCollection：

- AddItem(vName) As Object
- RemoveItem(vIndex)

示例

```
'VBS336
Dim ctrl
Dim objTimeCol1
Dim objTimeCol2
Dim coll
Dim timecol
Set ctrl = ScreenItems("TableControl")
Set objTimeCol1 = ctrl.GetTimeColumnCollection.AddItem("TimeColumn2010")
Set objTimeCol2 = ctrl.GetTimeColumnCollection.AddItem("TimeColumn2011")
objTimeCol1.Caption = "2010"
objTimeCol1.RangeType = 1
'The format to be used for date and time entries depends on the
'regional settings and language options in the operating system.
objTimeCol1.BeginTime = "01.01.2010 0:00:00"
'objTimeCol1.BeginTime = "01/01/2010 0:00:00"
objTimeCol1.EndTime = "31.12.2010 11:59:59"
'objTimeCol1.EndTime = "12/31/2010 11:59:59"
objTimeCol2.Caption = "2011"
objTimeCol2.RangeType = 0
objTimeCol2.BeginTime = "01.01.2011 0:00:00"
'objTimeCol2.BeginTime = "01/01/2011 0:00:00"
objTimeCol2.TimeRangeFactor = 1
objTimeCol2.TimeRangeBase = 3600000
Set coll = ctrl.GetTimeColumnCollection
For Each timecol In coll
 timecol.Align = 1
 timecol.Length = 12
 timecol.BackColor = RGB(240,240,0)
 timecol.ForeColor = RGB(130,160,255)
Next
```

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

GetToolbarButton

描述

以“ICCAxToolbarButton”类型的形式返回控件工具栏上按名称或索引指定的按钮功能。

语法

```
Ausdruck.GetToolbarButton(ByVal vIndex As Variant)
```

表达式

必需。返回“ScreenItem”类型对象的表达式。

参数

VARIANT

参数	描述
vIndex	工具栏按钮功能的索引或名称。

示例

```
'VBS337
Dim ctrl
Set ctrl = ScreenItems( "Control1" )
Dim toolbu
Set toolbu = ctrl.GetToolbarButton ("ShowHelp")
HMIRuntime.Trace "Name: " & toolbu.Name & vbCrLf
HMIRuntime.Trace "Index: " & toolbu.Index & vbCrLf
HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
```

说明

当使用“Get...”方法通过控件对象列表而不是控件对象访问属性时，必须删除属性的列表名称前缀。

例如，对于“ToolbarButton”列表，应该写为“toolbu.Index”而不是“toolbu.ToolbarButtonIndex”。

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

FunctionTrendControl (页 351)

AlarmControl (页 255)

GetToolbarButtonCollection

描述

以"ICCAxCollection"类型的形式返回控件工具栏的所有按钮功能的列表。

语法

```
Ausdruck.GetToolbarButtonCollection()
```

表达式

必需。返回"ScreenItem"类型对象的表达式。

参数

- -

ICCAxCollection 的特性和函数

下列属性可用于 ICCAxCollection：

- 计数
- Item

下列方法可用于 ICCAxCollection：

- AddItem(vName) As Object
- RemoveItem(vIndex)

示例

```
'VBS338
Dim ctrl
Dim coll
Dim toolbu
Set ctrl = ScreenItems( "Control1" )
Set coll = ctrl.GetToolbarButtonCollection
HMIRuntime.Trace "Number of toolbar buttons:" & coll.Count & vbCrLf
For Each toolbu In coll
 HMIRuntime.Trace toolbu.Name & vbCrLf
 HMIRuntime.Trace "Hotkey: " & toolbu.HotKey & vbCrLf
 HMIRuntime.Trace "Authorization: " & toolbu.PasswordLevel & vbCrLf
Next
```

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

FunctionTrendControl (页 351)

AlarmControl (页 255)

GetTrend

描述

以“ICCAxTrend”或“ICCAxFunctionTrend”类型的形式返回按名称或索引指定的 f(t) 或 f(x) 趋势视图。

语法

```
Ausdruck.GetTrend(ByVal vIndex As Variant)
```

表达式

必需。返回“ScreenItem”类型对象的表达式。

参数

VARIANT

参数	描述
vIndex	曲线的索引或名称。

运行系统专业版的示例

```
'VBS339
Dim ctrl
Dim objTrend
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrend = ctrl.GetTrend( "Trend 1" )
objTrend.PointStyle = 1
objTrend.LineWidth = 4
Set objTrend = ctrl.GetTrend(2)
objTrend.Provider = 1
objTrend.TagName = "Archive\ArchiveTag2"
```

说明

当使用“Get...”方法通过控件对象列表而不是控件对象访问属性时，必须删除属性的列表名称前缀。

例如，对于“Trend”列表，应该写为“objTrend.PointStyle”而不是“objTrend.TrendPointStyle”。

参见

OnlineTrendControl (页 418)

FunctionTrendControl (页 351)

GetTrendCollection

描述

以“ICCAxCollection”类型的形式返回 f(t) 或 f(x) 趋势视图的所有趋势的列表。

语法

```
Ausdruck.GetTrendCollection()
```

表达式

必需。返回"ScreenItem"类型对象的表达式。

参数

--

ICCAxCollection 的特性和函数

下列属性可用于 ICCAxCollection：

- 计数
- Item

下列函数可用于 ICCAxCollection：

- AddItem(vName) As Object
- RemoveItem(vIndex)

运行系统专业版的示例

```
'VBS340
Dim ctrl
Dim objTrendWnd
Dim objTimeAxis
Dim objValAxis
Dim objTrend
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindowCollection.AddItem("myWindow")
Set objTimeAxis = ctrl.GetTimeAxisCollection.AddItem("myTimeAxis")
Set objValAxis = ctrl.GetValueAxisCollection.AddItem("myValueAxis")
objTimeAxis.TrendWindow = objTrendWnd.Name
objValAxis.TrendWindow = objTrendWnd.Name
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend1")
objTrend.Provider = 1
objTrend.TagName = "Archive\ArchiveTag1"
objTrend.TrendWindow = objTrendWnd.Name
objTrend.TimeAxis = objTimeAxis.Name
objTrend.ValueAxis = objValAxis.Name
```

说明

当使用“Get...”方法通过控件对象列表而不是控件对象访问属性时，必须删除属性的列表名称前缀。

例如，对于“Trend”列表，应该写为“objTrend.TagName”而不是“objTrend.TrendTagName”。

参见

OnlineTrendControl (页 418)

FunctionTrendControl (页 351)

GetTrendWindow

描述

以“ICCAxTrendWindow”类型的形式返回按名称或索引指定的 f(t) 趋势视图或 f(x) 趋势视图的趋势视图对象。

语法

```
Ausdruck.GetTrendWindow(ByVal vIndex As Variant)
```

表达式

必选。返回“ScreenItem”类型对象的表达式。

参数

VARIANT

参数	描述
vIndex	曲线窗口的索引或名称。

运行系统专业版的示例

```
'VBS341
Dim ctrl
Dim objTrendWnd
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindow(1)
objTrendWnd.Visible = FALSE
Set objTrendWnd = ctrl.GetTrendWindow("trend window 2")
objTrendWnd.VerticalGrid = TRUE
objTrendWnd.FineGrid = TRUE
```

说明

当使用“Get...”方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于“TrendWindow”列表，应该写为“objTrendWnd.Visible”而不是“objTrendWnd.TrendWindowVisible”。

参见

OnlineTrendControl (页 418)

FunctionTrendControl (页 351)

GetTrendWindowCollection

描述

以“ICCAxCollection”类型的形式返回 f(t) 趋势显示或 f(x) 函数显示的所有趋势窗口对象的列表。

语法

```
Ausdruck.GetTrendWindowCollection()
```

表达式

必选项。 返回“ScreenItem”类型对象的表达式。

参数

--

ICCAxCollection 的特性和函数

下列属性可用于 ICCAxCollection：

- Count
- Item

下列函数可用于 ICCAxCollection：

- AddItem(vName) As Object
- RemoveItem(vIndex)

运行系统专业版的示例

```
'VBS342
Dim ctrl
Dim objTrendWnd
Dim objTimeAxis
Dim objValAxis
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindowCollection.AddItem("myWindow")
Set objTimeAxis = ctrl.GetTimeAxisCollection.AddItem("myTimeAxis")
Set objValAxis = ctrl.GetValueAxisCollection.AddItem("myValueAxis")
objTimeAxis.TrendWindow = objTrendWnd.Name
objValAxis.TrendWindow = objTrendWnd.Name
```

参见

OnlineTrendControl (页 418)

FunctionTrendControl (页 351)

GetValueAxis

描述

以“ICCAxValueAxis”类型的形式返回 f(t) 趋势视图中具有指定名称或索引的数值轴对象。

语法

```
Ausdruck.GetValueAxis(ByVal vIndex As Variant)
```

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

VARIANT

参数	描述
vIndex	数值轴的索引或名称。

示例

```
'VBS343
Dim ctrl
Dim objValAxis
Set ctrl = ScreenItems("OnlineTrendControl")
Set objValAxis = ctrl.GetValueAxis(1)
objValAxis.Visible = FALSE
Set objValAxis = ctrl.GetValueAxis("axis 2")
objValAxis.Label = "Value axis 2"
objValAxis.ScalingType = 0
objValAxis.Precisions = 2
objValAxis.AutoRange = TRUE
```

说明

当使用“Get...”方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于“ValueAxis”列表，应该写为“objValueAx.Visible”而不是“objValueAx.ValueAxisVisible”。

参见

OnlineTrendControl (页 418)

GetValueAxisCollection

描述

以“ICCAxCollection”类型的形式返回 f(t) 趋势视图中所有数值轴对象的列表。

语法

```
Ausdruck.GetValueAxisCollection()
```

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

- -

ICCAxCollection 的特性和函数

下列属性可用于 ICCAxCollection：

- Count
- Item

下列函数可用于 ICCAxCollection：

- AddItem(vName) As Object
- RemoveItem(vIndex)

示例

```
'VBS344
Dim ctrl
Dim objTrendWnd
Dim objValAxis1
Dim objValAxis2
Dim objTrend
Set ctrl = ScreenItems("OnlineTrendControl")
Set objTrendWnd = ctrl.GetTrendWindowCollection.AddItem("myWindow")
Set objValAxis1 = ctrl.GetValueAxisCollection.AddItem("myValueAxis1")
Set objValAxis2 = ctrl.GetValueAxisCollection.AddItem("myValueAxis2")
objValAxis1.TrendWindow = objTrendWnd.Name
objValAxis1.Label = "Value1"
objValAxis2.TrendWindow = objTrendWnd.Name
objValAxis2.inTrendColor = TRUE
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend1")
objTrend.TrendWindow = objTrendWnd.Name
objTrend.ValueAxis = objValAxis1.Name
Set objTrend = ctrl.GetTrendCollection.AddItem("myTrend2")
objTrend.TrendWindow = objTrendWnd.Name
objTrend.ValueAxis = objValAxis2.Name
```

说明

当使用“Get...”方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于“ValueAxis”列表，应该写为“objValueAxis1.Label”而不是“objValueAxis1.ValueAxisLabel”。

参见

OnlineTrendControl (页 418)

GetValueColumn

说明

以“ICCAxValueColumn”类型的形式返回表格视图中具有指定名称或索引的数值列对象。

语法

表达式.GetValueColumn(ByVal vIndex As Variant)

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

VARIANT

参数	描述
vIndex	f(t) 趋势视图中数值列的索引或名称。

示例

```
'VBS345
Dim ctrl
Dim objValueColumn
Set ctrl = ScreenItems("TableControl")
Set objValueColumn = ctrl.GetValueColumn("Valuecolumn1")
objValueColumn.Precisions = 4
Set objValueColumn = ctrl.GetValueColumn(2)
objValueColumn.ExponentialFormat = TRUE
```

说明

当使用“Get...”方法通过控件对象列表而不是控件对象访问属性时，必须删除属性的列表名称前缀。

例如，对于“ValueColumn”列表，应该写为“objValueColumn.Precisions”而不是“objValueColumn.ValueColumnPrecisions”。

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

GetValueColumnCollection

描述

以“ICCAxCollection”类型的形式返回 f(t) 趋势视图中所有数值列对象的列表。

语法

```
Ausdruck.GetValueColulmnCollection()
```

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

- -

ICCAxCollection 的特性和函数

下列属性可用于 ICCAxCollection：

- Count
- Item

下列函数可用于 ICCAxCollection：

- AddItem(vName) As Object
- RemoveItem(vIndex)

示例

```
'VBS346
Dim ctrl
Dim objValCol1
Dim objValCol2
Dim coll
Dim valcol
Set ctrl = ScreenItems("TableControl")
Set objValCol1 = ctrl.GetValueColumnCollection.AddItem("ValueColumn1")
Set objValCol2 = ctrl.GetValueColumnCollection.AddItem("ValueColumn2")
objValCol1.Caption = "Value Archive"
objValCol1.Provider = 1
objValCol1.TagName = "ProcessValueArchive\arch1"
objValCol1.TimeColumn = "TimeColumn1"
objValCol2.Caption = "Value Tag"
objValCol2.Provider = 2
objValCol2.TagName = "tagxx"
objValCol2.TimeColumn = "TimeColumn2"
Set coll = ctrl.GetValueColumnCollection
For Each valcol In coll
 valcol.Align = 2
 valcol.Length = 10
 valcol.AutoPrecisions = TRUE
Next
```

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

GetXAxis

描述

以“ICCAxValueAxis”类型的形式返回按名称或索引指定的 f(x) 趋势视图的 X 轴对象。

语法

```
Ausdruck.GetXAxis(ByVal vIndex As Variant)
```

表达式

必需。 返回“ScreenItem”类型对象的表达式。

参数

VARIANT

参数	描述
vIndex	X 轴的索引或名称。

示例

```
'VBS347
Dim ctrl
Dim objXAx
Set ctrl = ScreenItems("FunctionTrendControl")
Set objXAx = ctrl.GetXAxis(1)
objXAx.Visible = FALSE
Set objXAx = ctrl.GetXAxis("axis 2")
objXAx.Label = "X axis 2"
objXAx.ScalingType = 0
objXAx.Precisions = 2
objXAx.Color = RGB(109,109,109)
```

说明

当使用"Get..."方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于"XAxis"列表，应该写为"objXAx.Visible"而不是"objXAx.XAxisVisible"。

参见

OnlineTrendControl (页 418)

FunctionTrendControl (页 351)

GetXAxisCollection

描述

以"ICCAxCollection"类型的形式返回 f(x) 趋势视图的所有 X 轴对象的列表。

语法

```
Ausdruck.GetXAxisCollection()
```

表达式

必需。 返回"ScreenItem"类型对象的表达式。

参数

--

ICCAxCollection 的特性和函数

下列属性可用于 ICCAxCollection：

- Count
- Item

下列函数可用于 ICCAxCollection：

- AddItem(vName) As Object
- RemoveItem(vIndex)

示例

```
'VBS348
Dim ctrl
Dim objXAxis1
Dim objXAxis2
Dim coll
Dim axes
Set ctrl = ScreenItems("FunctionTrendControl")
Set objXAxis1 = ctrl.GetXAxisCollection.AddItem("myXAxis1")
objXAxis1.Label = "temperature"
Set objXAxis2 = ctrl.GetXAxisCollection.AddItem("myXAxis2")
objXAxis2.Label = "pressure"
Set coll = ctrl.GetXAxisCollection
HMIRuntime.Trace "Number of XAxis:" & coll.Count & vbCrLf
For Each axes In coll
 HMIRuntime.Trace axes.Name & vbCrLf
 HMIRuntime.Trace axes.Label & vbCrLf
Next
```

说明

当使用"Get..."方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于"XAxis"列表，应该写为"objXAxis1.Label"而不是"objXAxis1.XAxisLabel"。

参见

OnlineTrendControl (页 418)

FunctionTrendControl (页 351)

GetYAxis

描述

以"ICCAxValueAxis"类型的形式返回按名称或索引指定的 f(x) 趋势视图的 Y 轴对象。

语法

```
Ausdruck.GetYAxis(ByVal vIndex As Variant)
```

表达式

必需。 返回"ScreenItem"类型对象的表达式。

参数

VARIANT

参数	描述
vIndex	Y 轴的索引或名称。

示例

```
'VBS349
Dim ctrl
Dim objYAx
Set ctrl = ScreenItems("FunctionTrendControl")
Set objYAx = ctrl.GetYAxis(1)
objYAx.Visible = FALSE
Set objYAx = ctrl.GetYAxis("axis 2")
objYAx.Label = "Y axis 2"
objYAx.Align = 0
objYAx.Precisions = 3
objYAx.EndValue = 90.000
objYAx.BeginValue = 10.000
```

说明

当使用"Get..."方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于"YAxis"列表，应该写为"objYAx.Visible"而不是"objYAx.YAxisVisible"。

参见

OnlineTrendControl (页 418)

FunctionTrendControl (页 351)

GetYAxisCollection

描述

以"ICCAxCollection"类型的形式返回 f(x) 趋势视图的所有 Y 轴对象的列表。

语法

```
Ausdruck.GetYAxisCollection()
```

表达式

必需。 返回"ScreenItem"类型对象的表达式。

参数

- -

ICCAxCollection 的特性和函数

下列属性可用于 ICCAxCollection：

- Count
- Item

下列函数可用于 ICCAxCollection：

- AddItem(vName) As Object
- RemoveItem(vIndex)

示例

```
'VBS350
Dim ctrl
Dim objYAxis1
Dim objYAxis2
Dim coll
Dim axes
Set ctrl = ScreenItems("FunctionTrendControl")
Set objYAxis1 = ctrl.GetXAxisCollection.AddItem("myYAxis1")
objYAxis1.Label = "temperature"
Set objYAxis2 = ctrl.GetXAxisCollection.AddItem("myYAxis2")
objYAxis2.Label = "pressure"
Set coll = ctrl.GetYAxisCollection
HMIRuntime.Trace "Number of YAxis:" & coll.Count & vbCrLf
For Each axes In coll
 HMIRuntime.Trace axes.Name & vbCrLf
 HMIRuntime.Trace axes.Label & vbCrLf
Next
```

说明

当使用“Get...”方法通过 Control 对象列表而不是 Control 对象访问属性时，必须删去属性的列表名称前缀。

例如，对于“YAxis”列表，应该写为“objYAxis1.Label”而不是“objYAxis1.YAxisLabel”。

参见

OnlineTrendControl (页 418)

FunctionTrendControl (页 351)

1.5.6.2 方法 H-R

HideAlarm

描述

执行报警视图的 “隐藏报警”(Hide alarm) 按钮功能。

语法

```
Expression.HideAlarm()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

- -

参见

AlarmControl (页 255)

InsertData

描述

将数据添加到被调用的趋势。

语法

```
Expression.InsertData(dblAxisX As Variant, dblAxisY As Variant)
```

表达式

必需。 返回“Trend”类型对象的表达式。

参数

参数	描述
dblAxisX	X 轴的值
dblAxisY	Y 轴的值

示例

```
'VBS300
Dim lngFactor
Dim dblAxisX
Dim dblAxisY
Dim objTrendControl
Dim objTrend
Set objTrendControl = ScreenItems("Control1")
Set objTrend = objTrendControl.GetTrend("Trend 1")
For lngFactor = -100 To 100
dblAxisX = CDbl(lngFactor * 0.02)
dblAxisY = CDbl(dblAxisX * dblAxisX + 2 * dblAxisX + 1)
objTrend.InsertData dblAxisX, dblAxisY
Next
```

Item

描述

返回列表中的一个元素。

语法

Expression.Item(Index)

表达式

必需。 返回列表的表达式。

参数

Index

列表元素的名称或索引号：

- ScreenItems 列表：使用对象名称，如“HmiRuntime.Screens(1).ScreenItems("Circle")”或索引号。
- Screens 列表： 使用名称或索引号。
- SmartTags 列表： 只能将变量名用作 SmartTags 列表中的索引。 没法计数所有变量。

如果传送的数值与列表中的元素不相符，将发生错误。 返回值为“Nothing”。

```
On Error Resume Next
Dim screen
Set screen = HmiRuntime.Screens("Screen_1")
If (screen is Nothing)
then...
Else...
End If
```

要获得优化自动完成的支持，最佳做法是使用将画面名称和对象名称组合的寻址方式，例如 `"HmiRuntime.Screens("Screen").ScreenItems("Circle")"`。

示例

对于列表，item 方法是缺省的方法。 因此，下列两个实例的结果相同：

```
'VBS_Example_Item
HMIRuntime.Screens.Item(1)
HMIRuntime.Screens(1)
```

两条指令均引用相应的基本画面。

参见

ScreenItems（列表） (页 236)

ScreenItem (页 234)

LockAlarm

描述

执行报警视图的 “禁用报警”(Disable alarm) 按钮功能。

语法

```
Expression.LockAlarm()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

AlarmControl (页 255)

LoopInAlarm

描述

执行报警视图的 “报警回路”(Loop in alarm) 按钮功能。

语法

```
Expression.LoopInAlarm()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

- -

参见

AlarmControl (页 255)

MoveAxis

描述

执行 f(t) 和 f(x) 趋势视图的“移动轴区域”(Move axes area) 按钮功能。

语法

```
Expression.MoveAxis()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

- -

参见

OnlineTrendControl (页 418)

FunctionTrendControl (页 351)

MoveToFirst

描述

执行控件的“第一行”(First line) 按钮功能。

语法

```
Expression.MoveToFirst()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

用户归档控件 (页 566)

MoveToFirstLine

描述

执行报警视图的 "第一个报警"(First alarm) 按钮功能。

语法

```
Expression.MoveToFirstLine()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

AlarmControl (页 255)

MoveToFirstPage

描述

执行报警视图的 "第一页"(First page) 按钮功能。

语法

```
Expression.MoveToFirstPage()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

AlarmControl (页 255)

MoveToLast

描述

执行控件的 "最后一条数据记录"(Last data record) 按钮功能。

语法

```
Expression.MoveToLast()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

用户归档控件 (页 566)

MoveToLastLine

描述

执行报警视图的 “最后一个报警”(Last alarm) 按钮功能。

语法

```
Expression.MoveToLastLine()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

AlarmControl (页 255)

MoveToLastPage

描述

执行报警视图的 “最后一页”(Last page) 按钮功能。

语法

```
Expression.MoveToLastPage()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

AlarmControl (页 255)

MoveToNext

描述

执行控件的 “下一条数据记录”(Next data record) 按钮功能。

语法

```
Expression.MoveToNext()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

用户归档控件 (页 566)

MoveToNextLine

描述

执行报警视图的 “下一个报警”(Next alarm) 按钮功能。

语法

```
Expression.MoveToNextLine()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

AlarmControl (页 255)

MoveToNextPage

描述

执行报警视图的 “下一页”(Next page) 按钮功能。

语法

```
Expression.MoveToNextPage()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

AlarmControl (页 255)

MoveToPrevious

描述

执行控件的 “前一条数据记录”(Previous data record) 按钮功能。

语法

```
Expression.MoveToPrevious()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

用户归档控件 (页 566)

MoveToPreviousLine

描述

执行报警视图的 “上一个报警”(Previous alarm) 按钮功能。

语法

```
Expression.MoveToPreviousLine()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

AlarmControl (页 255)

MoveToPreviousPage

描述

执行报警视图的 “上一页”(Previous page) 按钮功能。

语法

```
Expression.MoveToPreviousPage()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

- -

参见

AlarmControl (页 255)

NextColumn

描述

执行表视图的 “下一列”(Next column) 按钮功能。

语法

```
Expression.NextColumn()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

- -

参见

OnlineTableControl (页 402)

NextTrend

描述

执行 f(t) 和 f(x) 趋势视图的 “下一个趋势”(Next trend) 按钮功能。

语法

```
Expression.NextTrend()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

OnlineTrendControl (页 418)

FunctionTrendControl (页 351)

OneToOneView

描述

执行 f(t) 和 f(x) 趋势视图的 “原始视图”(Original view) 按钮功能。

语法

```
Expression.OneToOneView()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

OnlineTrendControl (页 418)

FunctionTrendControl (页 351)

PasteRows

描述

执行配方视图的 “插入行”(Insert rows) 按钮功能。

语法

```
Expression.PasteRows()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

用户归档控件 (页 566)

PreviousColumn

描述

执行表视图的 "前一列"(Previous column) 按钮功能。

语法

```
Expression.PreviousColumn()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

OnlineTableControl (页 402)

PreviousTrend

描述

执行 f(t) 和 f(x) 趋势视图的 "上一个趋势"(Previous trend) 按钮功能。

语法

```
Expression.PreviousTrend()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

OnlineTrendControl (页 418)

FunctionTrendControl (页 351)

打印

描述

执行控件的 "打印"(Print) 按钮功能。

语法

```
Expression.Print()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

FunctionTrendControl (页 351)

AlarmControl (页 255)

QuitHorn

描述

执行报警视图的 “报警器确认”(Alarm annunciator acknowledgment) 按钮功能。

语法

```
Expression.QuitHorn()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

AlarmControl (页 255)

QuitSelected

描述

执行报警视图的 “单个确认”(Single acknowledgment) 按钮功能。

语法

```
Expression.QuitSelected()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

AlarmControl (页 255)

QuitVisible

描述

执行报警视图的“组确认”(Group acknowledgment) 按钮功能。

语法

```
Expression.QuitVisible()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

AlarmControl (页 255)

Read

Tag 对象说明

调用变量（tag 对象）之后可以立即读取其状态。 同时，变量对象包含数值读数。 读取变量时，确定变量的值、质量代码和时间戳。 “LastError”属性可用于确定是否调用成功。

“Name”和“TagPrefix”属性不会因此而改变。

若成功读取变量的值，则为变量对象的属性赋予以下值：

属性	分配
Value	变量值
Name	变量名称（不变）
QualityCode	质量等级
Timestamp	当前变量的时间戳
LastError	0
ErrorDescription	" "

若未成功读取变量的值，则为变量对象的属性赋予以下值：

属性	分配
Value	VT_Empty
Name	变量名称（不变）
QualityCode	严重超出范围
Timestamp	0
LastError	读取操作错误代码
ErrorDescription	关于 LastError 的错误描述

语法

```
Expression.Read([Readmode])
```

表达式

必需。 返回变量对象的表达式。 “Read”方法的返回值是读取的变量值。

参数

选项“Readmode”参数用于区分两种读取类型：

参数	描述
0	从过程映像（缓存）中读取变量值。 0 为默认值。
1	变量值可以直接从 AS 或通道（直接型）读取。

若忽略“Readmode”参数，默认情况下从过程映像中读取数值。 “Read”方法的返回值是读取作为 VARIANT 的变量值。

从过程映像读取

从过程映像读取时，对变量进行记录，从此刻起 PLC 循环轮询变量。 登录周期取决于组态的触发器。 通过 WinCC 从变量映像读取值。 对于“关闭画面”，再次结束变量操作。 调用具有以下特性：

- 通过 WinCC 从变量映像读取值。
- 与直接读取相比，调用速度更快（首次调用除外： 通常，首次调用时间较长，因为必须读取和记录 PLC 的值。）
- 调用的时间并不取决于总线负荷或 AS。

变量触发器的操作特性

变量触发器中包含的所有变量已知为“开放的画面”，注册有定义的监控时间。 因为立即请求所有的变量，所以通过通道可以实现可能的最佳优化目标。 若在操作期间触发器中的变量通过 Read 请求，则变量值已存在，且直接传递给调用。 若请求的变量未包含于触发器中，则情况与标准触发器相同。

循环触发器的操作特性

首次调用时，变量注册为循环周期的一半时间。 以后每次调用时，该值均存在。

事件驱动操作的特性

首次调用时，变量在“upon change”模式中注册。 在“upon change”模式中注册的过程变量对应于循环周期为 1s 的循环读取作业。

若事件（例如：鼠标单击）异步请求数值，则变量传递给变量映像。 从该时间点起，向 AS 循环请求变量，因此将增加基本负荷。 为了避免增加基本负荷，还可以同步读取数值。 同步调用将导致通讯负荷的一次性增加，但不会将变量传递给变量映像。

直接读取

对于直接读取的情况，将返回当前的值。 变量不会循环注册，仅一次性向 AS 请求数值。 直接读取具有以下特性：

- 从 AS 中显式读取数值。
- 与从过程映像读取相比，调用时间更长。
- 在其它情况下，调用的时间取决于总线负荷和 AS。

示例

直接从 AS 或通道读取变量

```
'VBS100
Dim objTag
Dim vntValue
Set objTag = HMIRuntime.Tags("Tagname")
vntValue = objTag.Read(1)     '直接读取
MsgBox vntValue
```

从过程映像读取变量

```
'VBS101
Dim objTag
Dim vntValue
Set objTag = HMIRuntime.Tags("Tagname")
vntValue = objTag.Read     '从缓存读取
MsgBox vntValue
```

TagSet 对象的描述

通过 TagSet 对象可选择在一次调用中读取多个变量。

相应的功能与变量对象的功能大致相同。 因此，下面仅介绍偏差。

表达式

必需。 返回"TagSet"类型对象的表达式。

从过程映像读取

TagSet 对象的特点是：在一次读取命令中可以请求多个变量。 变量在过程映像中注册为组，从而提高了进程的性能。

直接读取

由于一次调用可以处理多条读取命令，因此与单独的调用相比性能提高。

示例

以下示例显示如何在 TagSet 列表中包含变量、如何导入变量值以及之后如何读取。

```
'VBS174
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
group.Read
HMIRuntime.Trace "Motor1: " & group("Motor1").Value & vbNewLine
HMIRuntime.Trace "Motor2: " & group("Motor2").Value & vbNewLine
```

若可选参数“Readmode”设置为 1，则不会注册过程变量，但会从 AS 或通道直接读取。

```
group.Read 1
```

ReadTags

描述

执行配方视图的 “读取变量”(Read tags) 按钮功能。

语法

```
Expression.ReadTags()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

用户归档控件 (页 566)

Refresh

描述

重新绘制所有显示的画面。

语法

```
Expression.Refresh
```

表达式

必需。 返回“Screens”或“Screen”类型对象的表达式。

参数

--

示例

第一个示例为强制重新绘制所有显示画面：

```
'VBS149
HMIRuntime.Screens.Refresh
```

第二个示例为强制立即重新绘制基本画面：

```
'VBS150
HMIRuntime.Screens(1).Refresh
```

参见

Screen (页 231)

HMIRuntime（页 224）

Remove

TagSet 对象的描述

从 TagSet 列表中删除变量。 可按照名称或对变量对象的引用删除变量。

语法

```
Expression.Remove [Tag]
```

表达式

必选。用于返回"TagSet"类型对象的表达式。

参数

VARIANT

参数	描述
Tag	要从列表中删除的 WinCC 变量的名称或对某个变量对象的引用。

示例

以下示例显示了如何在 TagSet 列表中包含多个变量，以及如何重新删除变量。

```
'VBS175
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
group.Remove "Motor1"
```

DataSet 对象的描述

从列表中删除在参数"Name"中指定的元素。

语法

```
Expression.Remove [Name]
```

表达式

必选。用于返回"DataSet"类型对象的表达式。

参数

VARIANT

参数	描述
Name	要从列表中删除的对象的名称。

示例

以下示例显示如何从列表中删除对象"motor1"。

```
'VBS166
HMIRuntime.DataSet.Remove("motor1")
```

对象 Logging、AlarmLogs、DataLogs 的描述

此方法会从运行系统项目中删除先前转入的日志段。

使用"Remove"方法删除的日志段将从项目的"Common logging"文件夹中移除。

根据日志数据的不同，调用可能需要稍长的一段时间。 这可能会阻碍后续脚本的处理。 如果在"全局脚本"动作中启动调用（如通过触发变量启动动作），就可以避免画面中的动作受阻。

断开和清除日志时，会产生 CPU 负载。 这将影响性能。

说明

目前，只能在服务器上调用"Remove"方法。 不过，有一个示例显示了如何利用客户端从服务器启动此方法。

语法

对象 Logging、AlarmLogs

```
Expression.Remove [TimeFrom] [TimeTo] [TimeOut] [ServerPrefix]
```

表达式

必选。用于返回"Logging"或"AlarmLogs"类型对象的表达式。

对象 DataLogs

```
Expression.Remove [TimeFrom] [TimeTo] [TimeOut] [Type]
[ServerPrefix]
```

表达式

必选。用于返回"DataLogs"类型对象的表达式。

参数

TimeFrom

开始清除日志的时间点。
指定时间信息时，也可使用简略形式。"时间格式"部分对此进行了说明。

TimeTo

停止清除日志段的时间。
指定时间信息时，也可使用简略形式。"时间格式"部分对此进行了说明。

Timeout

以毫秒表示的超时。
如果输入"-1"作为值，等待时间将无穷长。 如果输入值"0"，将没有等待时间。

类型

日志的类型。
此参数（可选）只能用于删除变量记录的日志段。
可输入以下值：

分配的值	类型	描述
1	hmiDataLogFast	快速变量记录数据
2	hmiDataLogSlow	慢速变量记录数据
3	hmiDataLogAll	快速和慢速变量记录数据

ServerPrefix

保留供以后的版本使用。

返回值

如果删除日志段时出现错误，此方法将返回错误报警。 更多信息，请参见"数据库区的错误报警"主题。

时间格式

下面定义了用于指定时间信息的格式： YYYY-MM-DD hh:mm:ss，其中 YYYY 表示年，MM 表示月，DD 表示日，hh 表示时，mm 表示分，ss 表示秒。 例如，2004 年 7 月 26 日 11 时 2 分 1 秒将显示为：2004-07-26 11:02:01。

对于参数“TimeFrom”和“TimeTo”，日期和时间语句也可以采用简略形式。 在这种情况下，不必填写所有的格式域。 简略形式表示日期与时间信息可以缺少一个或多个参数，并且以秒值开始。 例如，可以采用“YYYY-MM”或“YYYY-MM-DD hh”格式指定时间。 使用语句 "TimeFrom" = "2004-09" 和 "TimeTo" = "2004-10-04" 时，将会交换 2004 年 9 月一直到 10 月 4 日（包括 10 月 4 日）之间的所有日志段。

示例

在以下示例中，特定时间段内事后转入（再次转入）的日志段会被移除，并将返回值作为跟踪值输出。

```
'VBS182
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.Remove("2004-08-22","2004-09-22",-1) &
vbNewLine
```

在以下示例中，事后转入（再次转入）的所有日志段都会被移除，并将返回值作为跟踪值输出。

```
'VBS183
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.Remove("","",-1) & vbNewLine
```

参见

Logging (页 229)

DataSet（列表） (页 221)

DataLogs（列表） (页 219)

AlarmLogs（列表） (页 216)

RemoveAll

TagSet 对象说明

从 TagSet 列表删除所有变量。

语法

```
Expression.RemoveAll
```

表达式

必需。 返回"TagSet"类型对象的表达式。

参数

--

示例：

以下示例显示如何在 TagSet 列表中包含多个变量以及如何删除所有变量。

```
'VBS176
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Motor1"
group.Add "Motor2"
group.RemoveAll
```

DataSet 对象说明

从 DataSet 列表中删除所有值或对象引用。

语法

```
Expression.RemoveAll
```

表达式

必需。 返回"DataSet"类型对象的表达式。

参数

--

示例：

示例为如何从列表中删除所有对象。

```
'VBS167
HMIRuntime.DataSet.RemoveAll
```

参见

DataSet（列表） (页 221)

Restore

对象 Logging、AlarmLogs、DataLogs 的描述

此方法会向运行系统项目中添加换出的日志段。

在转入时，会将日志段复制到项目的"Common logging"文件夹中。 因此，必须具备适当的存储容量。

根据日志数据的不同，调用可能需要稍长的一段时间。 这可能会阻碍后续脚本的处理。 如果在“全局脚本”动作中启动调用（如通过触发变量启动动作），就可以避免画面中的动作受阻。

日志的连接/复制会产生 CPU 负载，因为 SQL 服务器会出现额外的负载，尤其是在激活了签名检查的情况下。 复制日志段会降低硬盘访问的速度。

如果激活了签名检查，则在转入未签名或已修改的日志时将会返回一条错误消息。在转入过程中，即使出现多个错误，也只会返回一个错误报警。 此外，会为每个日志段生成一个

WinCC 系统报警。 Windows 事件查看器的 “应用程序”(Application) 部分会添加一个条目。这有助于检查哪些日志段出现了错误。

- 对于未签名的日志，返回值为“0x8004720F”。 事件查看器包含条目 “数据库 <db_name> 验证失败！未找到签名！”(Validation of database <db_name> failed! No signature found!)。
  日志将被转入。
- 对于已更改的日志，返回值为“0x80047207”。 事件查看器包含条目 “数据库 <db_name> 验证失败！”(Validation of database <db_name> failed!)。
  日志不会转入。

说明

目前，只能在服务器上调用“Restore”方法。 不过，有一个示例显示了如何利用客户端从服务器启动此方法。

语法

对象 Logging、AlarmLogs

```
Expression.Restore [SourcePath] [TimeFrom] [TimeTo] [TimeOut]
[ServerPrefix]
```

表达式

必选。用于返回“Logging”或“AlarmLogs”类型对象的表达式。

对象 DataLogs

```
Expression.Restore [SourcePath] [TimeFrom] [TimeTo] [TimeOut]
[Type] [ServerPrefix]
```

表达式

必选。用于返回“DataLogs”类型对象的表达式。

参数

SourcePath

日志数据的路径。

TimeFrom

开始转入日志的时间点。
指定时间信息时，也可使用简略形式。“时间格式”部分对此进行了说明。

TimeTo

停止转入日志段的时间点。
指定时间信息时，也可使用简略形式。“时间格式”部分对此进行了说明。

Timeout

以毫秒表示的超时。
如果输入“-1”作为值，等待时间将无穷长。 如果输入值“0”，将没有等待时间。

类型

日志的类型。
此参数（可选）只能用于存储变量记录的日志段。
可输入以下值：

分配的值	类型	描述
1	hmiDataLogFast	快速变量记录数据
2	hmiDataLogSlow	慢速变量记录数据
3	hmiDataLogAll	快速和慢速变量记录数据

ServerPrefix

保留供以后的版本使用。

返回值

如果转入日志段时出现错误，此方法将返回错误消息。 更多信息，请参见“数据库区的错误报警”主题。

时间格式

下面定义了用于指定时间信息的格式： YYYY-MM-DD hh:mm:ss，其中 YYYY 表示年，MM 表示月，DD 表示日，hh 表示时，mm 表示分，ss 表示秒。 例如，2004 年 7 月 26 日 11 时 2 分 1 秒将显示为：2004-07-26 11:02:01。

对于参数“TimeFrom”和“TimeTo”，日期和时间语句也可以采用简略形式。 在这种情况下，不必填写所有的格式域。 简略形式表示日期与时间信息可以缺少一个或多个参数，并且以秒值开始。 例如，语句可以采用“YYYY-MM”或“YYYY-MM-DD hh”的形式。 使用语句 "TimeFrom" = "2004-09" 和 "TimeTo" = "2004-10-04" 时，将会交换 2004 年 9 月一直到 10 月 4 日（包括 10 月 4 日）之间的所有日志段。

示例

在以下示例中，将会再次转入指定时间段之后的所有日志段，并将返回值作为跟踪值输出。

```
'VBS184
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.Restore("D:\Folder","2004-09-14","",-1) &
vbNewLine
```

在以下示例中，将会再次转入指定时间段内的所有“慢速变量记录”日志段，并将返回值作为跟踪值输出。

```
'VBS185
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.DataLogs.Restore("D:\Folder","2004-09-14
12:30:05","2004-09-20 18:30",-1,2) & vbNewLine
```

在以下示例中，将会再次转入指定时间段之前的所有“报警记录”日志段，并将返回值作为跟踪值输出。

```
'VBS186
HMIRuntime.Trace "Ret: " & HMIRuntime.Logging.AlarmLogs.Remove("","2004-09-20",-1) &
vbNewLine
```

参见

Logging (页 229)

DataLogs（列表） (页 219)

AlarmLogs（列表） (页 216)

1.5.6.3 方法 S-Z

SelectAll

描述

在基于表格的控件中选择所有行。

语法

```
Expression.SelectAll()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

--

参见

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

AlarmControl (页 255)

SelectedStatisticArea

描述

执行表格视图的 "设置统计数据范围"(Set statistics range) 按钮功能。

语法

```
Expression.SelectedStatisticArea()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

--

参见

OnlineTableControl (页 402)

SelectRow

描述

在基于表格的控件中选择特定行。

语法

```
Expression.SelectRow ByVal IRow As Long, Optional
bExtendSelection As Boolean
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

参数	描述
IRow	要选择的行的编号。
bExtendSelection	以选项的形式表明是否扩展当前选择。仅当可进行多重选择时才相关。

示例

- 当前选择了行 1。如果调用 `SelectRow 4, True`，将选择行 1 和行 4。
- 当前选择了行 1。如果调用 `SelectRow 4, False` 或调用无可选参数的 `SelectRow 4`，则仅选择行 4。

参见

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

AlarmControl (页 255)

ServerExport

描述

执行配方视图的 “导出日志”(Export log) 按钮功能。

语法

```
Expression.ServerExport()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

用户归档控件 (页 566)

ServerImport

描述

执行配方视图的 “导入日志”(Import log) 按钮功能。

语法

```
Expression.ServerImport()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

用户归档控件 (页 566)

SetHTML

描述

写入处于 HTML 浏览器显示范围内的 HTML 代码。

语法

```
Epression.SetHTML string
```

Expression

必选项。返回"HTMLBrowser"类型对象的表达式。

示例

```
Dim objBrowser
Set objBrowser= HMIRuntime.Screens("Screen_1").ScreenItems("HTML-Browser_1")
objBrowser.SetHTML "<h1>This is a title</h1><p>This is a <b>bold</b> text.</p>"
```

参见

HTMLBrowser (页 379)

SetOperationLock

描述

暂时关闭画面窗口的可操作性。可将相应画面窗口定义为透明。您可自行定义透明级别。

说明

该函数适用于除根画面外的所有画面窗口。

说明

不会在采用 Windows 7 操作系统的 PC 上更新暂时不可用的画面窗口的内容。
将在采用 Windows 8 或 Windows 10 的 PC 上持续更新画面窗口的内容。

语法

```
Expression.SetOperationLock(ByVal LockState as Bool, ByVal
TransparencyLevel as Long)
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

参数	描述
True	暂时关闭画面窗口的可操作性。
False	未暂时关闭画面窗口的可操作性。
Value	画面窗口的透明值

示例

在以下示例中，通过将“SetOperationLock”参数设为“true”，使画面窗口“MyLockedWindow”的内容在 Screen_1 画面中不可用：

```
HmiRuntime.Screens("Screen_1").ScreenItems("MyLockedWindow").SetOperationL
ock true, 20
```

数字“20”确定了画面窗口的透明度。

如果将“SetOperationLock”参数设为“false”，则可再次使用“MyLockedWindow”画面窗口。

```
HmiRuntime.Screens("Screen_1").ScreenItems("MyLockedWindow").SetOperationL
ock true, 20
```

ShowColumnSelection

描述

执行表视图的“选择列”(Select columns) 按钮功能。

语法

```
Expression.ShowColumnSelection()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

OnlineTableControl (页 402)

ShowComment

描述

执行报警视图的 "注释对话框"(Comment dialog) 按钮功能。

语法

```
Expression.ShowComment()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

AlarmControl (页 255)

ShowDisplayOptionsDialog

描述

执行报警视图的 "显示选项对话框"(Display options dialog) 按钮功能。

语法

```
Expression.ShowDisplayOptionsDialog()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

AlarmControl (页 255)

ShowEmergencyQuitDialog

描述

执行报警视图的 "紧急确认"(Emergency acknowledgment) 按钮功能。

语法

```
Expression.ShowEmergencyQuitDialog()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

AlarmControl (页 255)

ShowHelp

描述

执行控件的 "帮助"(Help) 按钮功能。

语法

```
Expression.ShowHelp()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

FunctionTrendControl (页 351)

AlarmControl (页 255)

ShowHideList

描述

执行报警视图的 "要隐藏的报警的列表"(List of alarm to hide) 按钮功能。

语法

```
Expression.ShowHideList()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

AlarmControl (页 255)

ShowHitList

描述

执行报警视图的 “统计列表”(Hit list) 按钮功能。

语法

```
Expression.ShowHitList()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

AlarmControl (页 255)

ShowInfoText

描述

执行报警视图的 “关于对话框”(About dialog) 按钮功能。

语法

```
Expression.ShowInfoText()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

AlarmControl (页 255)

ShowLockDialog

描述

执行报警视图的 “锁定对话框”(Lock dialog) 按钮功能。

语法

```
Expression.ShowLockDialog()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

AlarmControl (页 255)

ShowLockList

描述

执行报警视图的 “锁定列表”(Lock list) 按钮功能。

语法

```
Expression.ShowLockList()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

AlarmControl (页 255)

ShowLongTermArchiveList

描述

执行报警视图的 “历史报警列表（长期）”(Historical alarm list (long-term)) 按钮功能。

语法

```
Expression.ShowLongTermArchiveList()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

AlarmControl (页 255)

ShowMessageList

描述

执行报警视图的 “报警列表”(Alarm list) 按钮功能。

语法

```
Expression.ShowMessageList()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

AlarmControl (页 255)

ShowPercentageAxis

描述

执行 f(t) 趋势视图的 “相对轴”(Relative axis) 按钮功能。

语法

```
Expression.ShowPercentageAxis()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

OnlineTrendControl (页 418)

ShowPropertyDialog

描述

执行控件的 “组态对话框”(Configuration dialog) 按钮功能。

语法

```
Expression.ShowPropertyDialog()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

FunctionTrendControl (页 351)

AlarmControl (页 255)

ShowSelectArchive

描述

执行配方视图的 “选择数据连接”(Select data connection) 按钮功能。

语法

```
Expression.ShowSelectArchive()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

- -

参见

用户归档控件 (页 566)

ShowSelection

描述

执行配方视图的 “选择对话框”(Selection dialog) 按钮功能。

语法

```
Expression.ShowSelection()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

用户归档控件 (页 566)

ShowSelectionDialog

描述

执行报警视图的 "选择对话框"(Selection dialog) 按钮功能。

语法

```
Expression.ShowSelectionDialog()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

AlarmControl (页 255)

ShowSelectTimeBase

描述

执行配方视图的 "时基对话框"(Timebase dialog) 按钮功能。

语法

```
Expression.ShowSelectTimeBase()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

用户归档控件 (页 566)

ShowShortTermArchiveList

描述

执行报警视图的 "历史报警列表（短期）"(Historical alarm list (short-term)) 按钮功能。

语法

```
Expression.ShowShortTermArchiveList()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

AlarmControl (页 255)

ShowSort

描述

执行配方视图的 "排序对话框"(Sorting dialog) 按钮功能。

语法

```
Expression.ShowSort()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

用户归档控件 (页 566)

ShowSortDialog

描述

执行报警视图的 "排序对话框"(Sorting dialog) 按钮功能。

语法

```
Expression.ShowSortDialog()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

AlarmControl (页 255)

ShowTagSelection

描述

执行控件的 "选择数据连接"(Select data connection) 按钮功能。

语法

```
Expression.ShowTagSelection()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

--

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

FunctionTrendControl (页 351)

ShowTimebaseDialog

描述

执行报警视图的 "时基对话框"(Timebase dialog) 按钮功能。

语法

```
Expression.ShowTimebaseDialog()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

--

参见

AlarmControl (页 255)

ShowTimeSelection

描述

执行控件的 "选择时间范围"(Select time range) 按钮功能。

语法

```
Expression.ShowTimeSelection()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

--

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

FunctionTrendControl (页 351)

ShowTrendSelection

描述

执行 f(t) 和 f(x) 趋势视图的 "选择趋势"(Select trends) 按钮功能。

语法

```
Expression.ShowTrendSelection()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

--

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

FunctionTrendControl (页 351)

StartStopUpdate

描述

执行控件的 “开始”(Start) 或 “停止”(Stop) 按钮功能。

语法

```
Expression.StartStopUpdate()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

OnlineTrendControl (页 418)

OnlineTableControl (页 402)

FunctionTrendControl (页 351)

Stop

描述

关闭 WinCC Runtime。

语法

```
Expression.Stop()
```

表达式

必需项。返回“HMIRuntime”类型对象的表达式。

参数

--

参见

HMIRuntime（页 224）

Trace

说明

通过操作系统通道返回用于调试报警的自定义文本。

HMIRuntime.Trace 方法仅适用于 WinCC Runtime Professional。可使用“打印作业/脚本诊断”对象或“ApDiag”诊断工具显示以参数形式传送的文本。默认情况下，“ApDiag”诊断工具随 Runtime Professional 一起安装。诊断工具位于以下路径下：“C:\Program Files (x86)\Siemens\Automation\SCADA-RT_V11\WinCC\uTools”。更多信息，请参见 Internet (https://support.industry.siemens.com/cs/de/en/view/102777629)。

如果 Runtime Advanced 或面板需要跟踪，则可以使用“ShowSystemAlarm”系统函数。

语法

Expression.Trace"STRING"

表达式

必选。返回“HMIRuntime”类型对象的表达式。

参数

字符串型

作为调试报警发出的文本。

示例

以下实例将发出调试报警：

```
'VBS example trace

HMIRuntime.Trace "Customized error message"
```

参见

HMIRuntime（页 224）

UnhideAlarm

描述

执行报警视图的 “显示报警”(Unhide alarm) 按钮功能。

语法

```
Expression.UnhideAlarm()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

AlarmControl (页 255)

UnlockAlarm

描述

执行报警视图的 “解锁报警”(Unlock alarm) 按钮功能。

语法

```
Expression.UnlockAlarm()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

AlarmControl (页 255)

UnselectAll

描述

在基于表格的控件的单元格中移除所有选择内容。

语法

```
Expression.UnselectAll()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

AlarmControl (页 255)

UnselectRow

描述

在基于表格的控件的特定单元格中移除选择内容。

语法

```
Expression.UnselectRow(ByVal IRow As Long)
```

Expression

必需项。返回“ScreenItem”类型对象的表达式。

参数

Long

参数	描述
IRow	要删除相应选项的行数。

参见

OnlineTableControl (页 402)

TrendRulerControl (页 532)

用户归档控件 (页 566)

AlarmControl (页 255)

Write

Tag 对象说明

将值写入变量。 “LastError”属性可用于确定是否调用成功。

若成功设置变量的值，则为变量对象的属性赋予以下值：

属性	分配
值	变量值由用户设置（不变）
名称	变量名称（不变）
QualityCode	严重超出范围
Timestamp	0
LastError	0
ErrorDescription	" "

若未成功设置变量的值，则为变量对象的属性赋予以下值：

属性	分配
值	变量值由用户设置（不变）
名称	变量名称（不变）
QualityCode	严重超出范围
Timestamp	0

属性	分配
LastError	写操作错误代码
ErrorDescription	关于 LastError 的错误描述

语法

```
Expression.Write [Value],[Writemode]
```

表达式

必需。 返回变量对象的表达式。

参数

要写入的值可以作为参数直接传递给方法。 若未指定参数，则使用“Value”属性的值。 使用“Writemode”选项参数可以选择变量值是同步写入还是异步写入。 若未使用“Writemode”参数，则执行异步写入作为其默认值。

写入过程期间，未提供关于变量状态的信息。

“Value”属性包含的值在写入操作之前或期间设置，因此可能不会与变量的当前真实值对应。 若应更新变量数据，使用“Read”方法。

参数	说明
Value（可选）	指定变量值。 指定的值覆盖变量对象“Value”属性中的值。 未指定变量值。 变量接受变量对象“Value”属性的当前值。
Writemode（可选）	0 或空： 异步写入变量值。 0 为默认值。 1: 同步写入变量值。

对于异步写入，值立即写入到变量映像中。 若还在编程控制器中写入值，则用户不会收到任何反馈。

对于同步写入（直接写入 PLC），在 PLC 运行准备就绪时，写入操作实际已发生。 若写入操作不成功，则用户收到一条核对消息。

示例

异步写入

```
'VBS104
Dim objTag
Set objTag = HMIRuntime.Tags("Var1")
objTag.Value = 5
objTag.Write
MsgBox objTag.Value
```

或

```
'VBS105
Dim objTag
Set objTag = HMIRuntime.Tags("Var1")
objTag.Write 5
MsgBox objTag.Value
```

同步写入

```
'VBS106
Dim objTag
Set objTag = HMIRuntime.Tags("Var1")
objTag.Value = 5
objTag.Write .1
MsgBox objTag.Value
```

或

```
'VBS107
Dim objTag
Set objTag = HMIRuntime.Tags("Var1")
objTag.Write 5, 1
MsgBox objTag.Value
```

TagSet 对象的描述

通过 TagSet 对象可选择在一次调用中写入多个变量。

相应的功能与变量对象的功能大致相同。 因此，下面仅介绍偏差。

表达式

必需。 返回"TagSet"类型对象的表达式。

参数

为了写入不同的值，必须设置每个变量对象的"Value"属性，然后必须调用写入，且不带"Value"参数。 因为在一次调用中可以对写入命令分组，因此相对于单次调用而言显著提高性能。

在 TagSet 对象中，不可能使用"Write"方法传递值。 每个值必须使用各个变量对象的"Value"属性设置。

示例

以下示例显示如何在 TagSet 列表中包含变量、如何设置变量值以及之后如何写入。

```
'VBS173
Dim group
Set group = HMIRuntime.Tags.CreateTagSet
group.Add "Wert1"
group.Add "Wert2"
group("Wert1").Value = 3
group("Wert2").Value = 9
group.Write
```

若设置可选参数"Writemode"等于 1，则过程变量同步写入（直接写入 AS）。

```
group.Write 1
```

WriteTags

描述

执行配方视图的 "写入变量"(Write tags) 按钮功能。

语法

```
Expression.WriteTags()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

用户归档控件 (页 566)

ZoomArea

描述

执行 f(t) 和 f(x) 趋势视图的 "缩放区域"(Zoom area) 按钮功能。

语法

```
Expression.ZoomArea()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

- -

参见

OnlineTrendControl (页 418)

FunctionTrendControl (页 351)

ZoomInOut

描述

执行 f(t) 和 f(x) 趋势视图的 "缩放"(Zoom +/-) 按钮功能。

语法

```
Expression.ZoomInOut()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

--

参见

OnlineTrendControl (页 418)

FunctionTrendControl (页 351)

ZoomInOutTime

描述

执行 f(t) 趋势视图的 "缩放时间轴"(Zoom time axis +/-) 按钮功能。

语法

```
Expression.ZoomInOutTime()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

--

参见

OnlineTrendControl (页 418)

ZoomInOutValues

描述

执行 f(t) 趋势视图的 “缩放数值轴”(Zoom value axis +/-) 按钮功能。

语法

```
Expression.ZoomInOutValues()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

OnlineTrendControl (页 418)

ZoomInOutX

描述

执行 f(x) 趋势视图的“X 轴缩放 +/-”(Zoom X axis +/-) 按钮功能。

语法

```
Expression.ZoomInOutX()
```

表达式

必需项。返回“ScreenItem”类型对象的表达式。

参数

--

参见

FunctionTrendControl (页 351)

ZoomInOutY

描述

执行 f(x) 趋势视图的"Y 轴缩放 +/-"(Zoom Y axis +/-) 按钮功能。

语法

```
Expression.ZoomInOutY()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

--

参见

FunctionTrendControl (页 351)

ZoomMove

描述

执行 f(t) 和 f(x) 趋势视图的 "移动趋势区域"(Move trend area) 按钮功能。

语法

```
Expression.ZoomMove()
```

表达式

必需项。返回"ScreenItem"类型对象的表达式。

参数

--

参见

OnlineTrendControl (页 418)

FunctionTrendControl (页 351)

1.5.7 来自数据库区域的错误消息

简介

访问数据库时，将在执行后返回一个值。以“0x8...”开头的值表示错误消息。不以“0x8...”开头的值表示状态消息。

状态消息

定义了以下状态消息：

0x0	OK
0x1	该功能未在参数分配中或在内部检测到任何错误。出现此返回值的可能原因包括： 连接数据库时： - 在给定时间窗中找不到归档。 - 在给定时间窗中找到归档，但归档已连接。 与数据库断开连接时： - 在给定时间窗中找不到连接的归档。此处不进行检查以确定是否存在连接的归档。

错误消息

定义了以下错误消息：

错误代码	错误消息
0x80047200	WinCC Runtime 未激活
0x80047201	归档类型无效
0x80047202	下限无效
0x80047203	上限无效
0x80047204	无法创建路径“CommonArchiving”
0x80047205	超时，请重试

错误代码	错误消息
0x80047206	WinCC Runtime 已停止
0x80047207	签名错误 至少有一个数据库的签名无效，无法进行连接
0x80047208	无法连接数据库
0x80047209	无法复制到“CommonArchiving”
0x8004720A	数据库文件名的语法无效
0x8004720B	数据库列表不存在
0x8004720C	已连接数据库
0x8004720D	无法断开数据库连接
0x8004720F	已连接无签名的数据库 至少已连接一个无签名的数据库。
0x80047210	路径错误： - 路径无效 - 指定的路径不包含 *.MDF 文件 - 无指定路径的访问权限

C 脚本

2

2.1 系统函数

2.1.1 ActivateNextScreen

说明

WinCC 将保存用户在运行期间所打开画面的名称以及这些画面的打开顺序。

只能在 C 脚本中使用。

在“运行系统设置 > 画面 > 画面缓冲区”(Runtime settings > Screens > Screen buffer) 编辑器中，指定画面缓冲区的最大大小。

ActivateNextScreen 系统函数现在用于打开上次调用 ActivatePreviousScreen 之前所打开的画面。

语法

```
BOOL ActivateNextScreen();
```

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

ActivateNextScreen 函数在以下程序代码中用于调用下一个画面并将返回值保存到 b_error 变量。

保存的返回值可在后续代码中进行处理。

```
{
BOOL b_error;

//Open next screen
b_error = ActivateNextScreen();

if(b_error)
 {
   // User defined code if
   // function succeeds without error
   ...
 }
 else
 {
   // User defined code in case of error
   ...
 }
...
}
```

2.1.2 ActivatePreviousScreen

说明

WinCC 将保存用户在运行期间所打开画面的名称以及这些画面的打开顺序。

该系统函数只能用于 C 脚本。

在“运行系统设置 > 画面 > 画面缓冲区”(Runtime settings > Screens > Screen buffer) 编辑器中，指定画面缓冲区的最大大小。

ActivatePreviousScreen 系统函数现在用于打开当前打开的画面之前所打开的画面。

语法

BOOL ActivatePreviousScreen();

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

ActivatePreviousScreen 函数在以下程序代码中用于调用上一个画面并将返回值保存到 b_error 变量。

保存的返回值可在后续代码中进行处理。

```
{
BOOL b_error;

//Open previous screen
b_error = ActivatePreviousScreen();

if(b_error)
 {
   // User defined code if
   // function succeeds without error
   ...
 }
 else
 {
   // User defined code in case of error
   ...
 }
...
}
```

2.1.3 ActivateScreen

描述

将画面切换到指定的画面。

使用“ActivateScreenByNumber”系统函数可以从根画面切换到永久性区域，反之亦然。

在函数列表中使用

激活画面（画面名称，对象编号）

在用户自定义函数中使用

ActivateScreen Screen_name, Object_number

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

画面名称

要切换到的画面的名称。

对象编号

画面切换后在指定画面中获得焦点的操作员控件元素。操作员控件元素的编号在组态期间使用 TAB 顺序确定。

在指定为“0”时：

- 如果调用该系统函数时焦点位于永久性区域，则永久性区域保留焦点。
- 如果调用该系统函数时焦点位于根画面，则指定画面中的第一个操作员控件元素获得焦点。

说明

如果将“到达边界”事件分配给“ActivateScreen”系统函数，则只有数值“0”对“对象编号”参数有效。活动对象不是由对象号定义的，而是由画面更改之前其 X 位置定义的。

示例

例如，单击按钮时，以下程序代码将使用 ActivateScreen 函数激活“Screen_2”。

```
Sub ActivateScreen_2()

'User-defined code
'' i. e. when pressing a button

ActivateScreen "Screen_2",0
```

2.1.4 ActivateScreenInScreenWindow

描述

将画面切换到在指定画面窗口中的指定画面。

在函数列表中使用

激活画面窗口中的画面（画面名称，画面窗口、新建画面名称）

在用户自定义函数中使用

ActivateScreenInScreenWindow Screen_name, Screen_window, New_screen_name

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“AUTOHOTSPOT”。

参数

画面名称

在画面窗口中显示的画面的名称。

画面窗口

要显示新画面的画面窗口的名称。

新建画面名称

要在画面窗口中显示的新画面的名称。

示例

单击按钮时，将使用以下程序代码中的 ActivateScreenInScreenWindow 函数激活“Screen_2”画面。

```
{
// User defined code
// i.e. when pressing a button
ActivateScreenInScreenWindow (GetParentScreen(screenName),
GetParentScreenWindow(screenName), "Screen_2");
...
}
```

2.1.5 ActivateStartScreen

描述

打开组态的启动画面。

只能在 C 脚本中使用。

语法

```
BOOL ActivateStartScreen();
```

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

ActivateStartScreen 函数在以下程序代码中用于调用组态的启动画面并将返回值保存到 b_error 变量。

保存的返回值可在后续代码中进行处理。

```
{
BOOL b_error;

//Open start screen
b_error = ActivateStartScreen();

if(b_error)
 {
   // User defined code if
   // function succeeds without error
   ...
 }
 else
 {
   // User defined code in case of error
   ...
 }
...
}
```

2.1.6 ActivateStoredScreen

描述

打开使用 StoreScreen 系统函数保存的画面。

只能在 C 脚本中使用。

语法

BOOL ActivateStoredScreen();

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

ActivateStoredScreen 函数在以下程序代码中用于调用存储的画面并将返回值保存到 b_error 变量。

保存的返回值可在后续代码中进行处理。

```
{
BOOL b_error;

//Open stored screen
b_error = ActivateStoredScreen();

if(b_error)
 {
   // User defined code if
   // function succeeds without error
   ...
 }
 else
 {
   // User defined code in case of error
   ...
 }
...
}
```

参见

StoreScreen（页 1739）

2.1.7 DateToSystemTime

说明

转换 DATE 格式的日期时间技术数据到 SYSTEMTIME 数据格式。

只能在 C 脚本中使用。

在函数列表中使用

DateToSystemTime （值、指向 Date/Time 的指针)

语法

```
DateToSystemTime(Value, PointerToTime);
```

参数

Value

DATE 数据格式的值

PointerToTime

指向 SYSTEMTIME 格式的结果的指针

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

```
BOOL BRet;
SYSTEMTIME st_1, st_2;
DATE       d_1, d_2;   // wtypes.h.  DATE type. Visual Studio documnetation.

GetSystemTime( &st_1 );

printf( "st_1.wYear = %d \r\n", st_1.wYear );
printf( "st_1.wMonth = %d \r\n", st_1.wMonth );
printf( "st_1.wDayOfWeek = %d \r\n", st_1.wDayOfWeek );
printf( "st_1.wDay = %d \r\n", st_1.wDay );printf( "st_1.wHour = %d \r\n", st_1.wHour );
printf( "st_1.wMinute = %d \r\n", st_1.wMinute );
printf( "st_1.wSecond = %d \r\n", st_1.wSecond );
printf( "st_1.wMilliseconds = %d \r\n", st_1.wMilliseconds );

BRet = SystemTimeToDate( st_1, &d_1 );
printf( "DATE d = %ld \r\n \r\n", d_1 );
printf( "DATE d = %lf \r\n \r\n", d_1 );
printf( "DATE d = %f \r\n \r\n", d_1 );

BRet = DateToSystemTime( d_1, &st_2 );
printf( "st_2.wYear = %d \r\n", st_2.wYear );
printf( "st_2.wMonth = %d \r\n", st_2.wMonth );
printf( "st_2.wDayOfWeek = %d \r\n", st_2.wDayOfWeek );
printf( "st_2.wDay = %d \r\n", st_2.wDay );
printf( "st_2.wHour = %d \r\n", st_2.wHour );
printf( "st_2.wMinute = %d \r\n", st_2.wMinute );
printf( "st_2.wSecond = %d \r\n", st_2.wSecond );
printf( "st_2.wMilliseconds = %d \r\n \r\n \r\n", st_2.wMilliseconds );
```

2.1.8 DecreaseTag

描述

从变量值中减去指定的值。

X = X - a

说明

系统函数使用同一变量作为输入和输出值。当该系统函数用于转换数值时，必须使用辅助变量。可以使用"SetTag"系统函数为辅助变量指定变量值。

如果为报警事件组态了系统函数，但变量未在当前画面中使用，则无法确定在 PLC 中使用的实际变量值。通过设置“连续循环”采集模式可以改善这种情况。

在函数列表中使用

减少变量（变量，值）

在用户自定义函数中使用

DecreaseTag Tag, Value

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“AUTOHOTSPOT”。

参数

变量

要减去指定值的变量。

值

其值作为减数。

示例

下面的程序代码会以 value 变量的值为减量对 varX 变量的值进行递减。输入的值将保存在 old_value 变量中，并与新的 varX 值一起输出。

```
{
BYTE varX;
BYTE value;

//user input
...
BYTE old_value = varX;

//Decrease tag
DecreaseTag(varX, value);

//print original value and function result
printf ("User input: %i\r\n, Result of function DecreaseTag: %i\r\n", old_value, varX);
...
}
```

2.1.9 GetLink

功能

指示对象属性的当前变量连接。

语法

BOOL GetLink(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName, LPLINKINFO *pLink);

参数

lpszPictureName

画面名称

lpszObjectName

对象名称

lpszPropertyName

对象属性

pLink

指向类型结构的指针：LINKINFO

返回值

TRUE

已执行完函数，未发生任何错误。

FALSE

发生错误。

参见

结构定义 LINKINFO (页 1802)

2.1.10 GetLinkedTag

说明

返回逻辑链接到指定对象属性的变量的名称。

只能在 C 脚本中使用。

语法

```
char* GetLinkedTag(ScreenName, Object, NameOfProperty);
```

参数

ScreenName

指向画面名称的指针。

Object

指向对象名称的指针。

NameOfProperty

指向对象属性名称的指针。

返回值

返回给逻辑链接到指定对象属性的变量的名称。

说明

面板上的用户数据类型的返回值

如果用户数据类型直接在面板上互连，则不会返回变量名称。只有用户数据类型的结构元素单独在面板上互连时才会返回变量名称。

示例

以下程序代码将函数“GetLinkedTag”的返回值保存到“pszVarName”变量。如果返回值有效，则将随最大路径长度一起保存到 szVAarName 变量。保存的返回值可在后续代码中进行处理。

```
char CFunktion()
{
char* pszVarName = NULL;
char szVarName[_MAX_PATH+1]; //Get the TagName
 pszVarName = GetLinkedVariable("gs_stand_graph_00","Textfield6","Visible"); //Copy the
string
 if (strcmp (pszVarName,"")!= 0)
{
     strncpy(szVarName,pszVarName,_MAX_PATH);
}
else printf("The property 'visible' is not dynamized\r\n");
//User defined code where the
//user can do something with the returnvalue
...
}
```

2.1.11 GetLocalScreen

说明

返回画面名称的指针。

只能在 C 脚本中使用。

语法

char* GetLocalScreen(ScreenName);

参数

ScreenName

指向画面名称的指针。

返回值

指向画面名称的指针。

说明

传送的调用参数"ScreenName"的语法必须与图形系统的画面路径的语法相对应：

<Screen_name>.<Screen_window_name>:<Screen_name>.<Screen_window_name>:<Screen_name>...

原理

"Screen_A"包含"Screen_window_B"。在画面窗口中调用另一画面并将其显示为最小化的图像等。

在画面窗口中调用画面

1. "Screen_window_B"显示"Screen_C"。
   "Screen_C"包含"Screen_window_D"。
2. "Screen_window_D"显示"Screen_E"。

画面名称指针

在下一步返回系统函数"`GetLocalScreen(ScreenName)`"和以下值：

- 如果在"Screen_E"中调用系统函数，则返回"Screen_E"
- 如果在"Screen_C"中调用系统函数，则返回"Screen_C"
- 如果在"Screen_A"中调用系统函数，则返回"Screen_A"

示例

以下程序代码将函数 GetLocalScreen 的返回值保存到 pszScrName 变量。如果返回值有效（非 NULL），则将随 maximum_MAX_PATH 字符一起保存到 szScrName 变量。

```
{
char* pszScrName = NULL;
char szScrName[_MAX_PATH+1];

//Get the Local Screen
pszScrName = GetLocalScreen(lpszScreenName);

//Copy the string
if (pszScrName != NULL)
 {
   strncpy(szScrName,pszScrName,_MAX_PATH);
// print local screen name
   printf ("Local screen name: %s\r\n", szScrName);
 }
...
}
```

2.1.12 GetLanguageByLocaleID

描述

确定当前运行系统的语言。

只能在 C 脚本中使用。

语法

DWORD GetLanguageByLocaleID ();

返回值

语言 ID。

可以应用以下赋值（十六进制语言代码）：

符号名称	值（十六进制）	缩写
LANG_ARABIC	0x0401	
LANG_AFRIKAANS	0x0436	
LANG_ALBANIAN	0x041C	
LANG_BASQUE	0x042D	
LANG_BULGARIAN	0x0402	
LANG_BYELORUSSIAN	0x0423	
LANG_CATALAN	0x0403	
LANG_CHINESE	0x0804	
LANG_CROATIAN	0x041A	
LANG_CZECH	0x0405	CSY
LANG_DANISH	0x0406	DAN
LANG_DUTCH	0x0413	NLD
LANG_ENGLISH	0x0409	ENU
LANG_ESTONIAN	0x0425	
LANG_FAEROESE	0x0438	
LANG_FARSI	0x0429	
LANG_FINNISH	0x040B	FIN
LANG_FRENCH	0x040C	FRA
LANG_GERMAN	0x0407	DEU
LANG_GREEK	0x0408	
LANG_HEBREW	0x040D	
LANG_HUNGARIAN	0x040E	HUN
LANG_ICELANDIC	0x040F	ISL
LANG_INDONESIAN	0x0421	
LANG_ITALIAN	0x0410	ITA
LANG_JAPANESE	0x0411	
LANG_KOREAN	0x0412	
LANG_LATVIAN	0x0426	
LANG_LITHUANIAN	0x0427	
LANG_NORWEGIAN	0x0414	NOR

符号名称	值（十六进制）	缩写
LANG_POLISH	0x0415	PLK
LANG_PORTUGUESE	0x0416	PTB
LANG_ROMANIAN	0x0418	
LANG_RUSSIAN	0x0419	RUS
LANG_SLOVAK	0x041B	SKY
LANG_SLOVENIAN	0x0424	
LANG_SORBIAN	0x042E	
LANG_SPANISH	0x040A	ESP
LANG_SWEDISH	0x041D	SVE
LANG_THAI	0x041E	
LANG_TURKISH	0x041F	TRK
LANG_UKRAINIAN	0x0422	

示例

以下程序代码读出当前的运行系统语言并将返回值保存在 rt_language 变量中。

保存的返回值可在后续代码中进行处理（本例中，使用 printf 输出）。

```
{
DWORD rt_language;

//Get the current language
rt_language = GetLanguageByLocaleID ();

//print language code
printf ("Language code: %d\r\n", rt_language);
...
}
```

2.1.13 GetParentScreen

说明

此函数用于确定特定画面路径中的父画面名称。

只能在 C 脚本中使用。

语法

```
char* GetParentScreen(ScreenName);
```

参数

ScreenName

指向画面名称的指针。传送的调用参数 Screen name 的语法必须与图形系统的画面路径的语法相对应：

<Screen_name>.<Screen_window_name>:<Screen_name>.<Screen_window_name>:<Screen_name>...

说明

“.”字符用于区分画面和画面对象名称。“:”字符用于层级结构化。因此，在名称标识中仅使用“-”或“_”作为定界符。

返回值

父画面的名称。

原理

“Screenwindow_1”画面窗口位于“Screen_1”画面中。在画面窗口中，该画面将变为包含“Screenwindow_2”画面窗口的“Screen_2”，依此类推。

画面路径：Screen_1.Screenwindow_1:Screen_2.Screenwindow_2:Screen_3

GetParentScreen 返回：

- Screen_2（针对在画面“Screen_3”中调用系统函数的情况）。
- Screen_1（针对在画面“Screen_2”中调用系统函数的情况）。

GetParentScreenwindow 返回：

- Screenwindow_2（针对在画面“Screen_3”中调用系统函数的情况）。
- Screenwindow_1（针对在画面“Screen_2”中调用系统函数的情况）。

示例

以下程序代码将函数 GetParentScreen 的返回值保存到 pszScrName 变量。如果返回值有效（非 NULL），则将随 maximum_MAX_PATH 字符一起保存到 szScrName 变量。

```
{
char* pszScrName = NULL;
char szScrName[_MAX_PATH+1];

//Get the Parent Screen
pszScrName = GetParentScreen(lpszScreenName);

//Copy the string
if (pszScrName != NULL)
 {
   strncpy(szScrName,pszScrName,_MAX_PATH);
// print Screen name
   printf ("Screen name: %s\r\n", szScrName);
 }
...
}
```

2.1.14 GetParentScreenWindow

说明

此函数用于确定所传送的画面路径中的父画面窗口名称。

只能在 C 脚本中使用。

语法

char* GetParentScreenWindow(ScreenName);

参数

ScreenName

指向画面名称的指针。传送的 Screen name 调用参数的语法必须与图形系统的画面路径的语法相对应：

<Screen name>.<screen window name>:<screen name>.<screen window name>:<screen name>...“.”字符用于区分画面和画面对象名称。“:”字符用于层级结构化。因此，在名称标识中仅使用“-”或“_”作为定界符。

说明

“.”字符用于区分画面和画面对象名称。“:”字符用于层级结构化。因此，在名称标识中仅使用“-”或“_”作为定界符。

返回值

父画面窗口的名称。

原理

“Screenwindow_1”画面窗口位于“Screen_1”画面中。在画面窗口中，该画面将变为包含“Screenwindow_2”画面窗口的“Screen_2”，依此类推。

画面路径：Screen_1.Screenwindow_1:Screen_2.Screenwindow_2:Screen_3

GetParentScreen 返回：

- Screen_2（针对在画面“Screen_3”中调用系统函数的情况）。
- Screen_1（针对在画面“Screen_2”中调用系统函数的情况）。

GetParentScreenwindow 返回：

- Screenwindow_2（针对在画面“Screen_3”中调用系统函数的情况）。
- Screenwindow_1（针对在画面“Screen_2”中调用系统函数的情况）。

示例

以下程序代码将函数 GetParentScreenWindow 的返回值保存到 pszScrName 变量。如果返回值有效（非 NULL），则将随 maximum_MAX_PATH 字符一起保存到 szScrName 变量。

```
{
char* pszScrName = NULL;
char szScrName[_MAX_PATH+1];

//Get the Parent Screen Window
pszScrName = GetParentScreenWindow(lpszScreenName);

//Copy the string
if (pszScrName != NULL)
 {
   strncpy(szScrName,pszScrName,_MAX_PATH);
// print name of the parent screen window
   printf ("Parent screen window: %s\r\n", szScrName);
 }
...
}
```

2.1.15 GetProp

2.1.15.1 GetPropBOOL

描述

返回数据类型"BOOL"属性的当前状态。

只能在 C 脚本中使用。

语法

BOOL GetPropBOOL(ScreenName, Object, NameOfTheProperty)

参数

ScreenName

画面名称

Object

对象名称。如果函数调用与画面对象属性相关，则必须设置参数 Object = NULL。

NameOfTheProperty

对象属性的名称

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

以下程序代码读出对象是否可见。该值将保存到 b_error 变量。

```
{
BOOL b_error;

//Get the property Visible of object IOField1 in screen gs_graph_iofield
b_error = GetPropBOOL("gs_graph_iofield","IOField1","Visible");

if(b_error)
{
  // User defined code if the
  // object is visible
  ...
}
else
{
  // User defined code if the
  // object is not visible
  ...
}
}
```

将根据返回值处理特定代码。

2.1.15.2 GetPropChar

说明

返回数据类型为“Char”的属性值。

只能在 C 脚本中使用。

语法

Char* GetPropChar(ScreenName, Object, NameOfTheProperty)

参数

ScreenName

画面名称

Object

对象名称。如果函数调用与画面对象属性相关，则必须设置参数 Object = NULL。

NameOfTheProperty

对象属性的名称

返回值

数据类型“Char”的属性值。

示例

以下程序代码使用 GetPropChar 函数读取对象中的工具提示文本并对其进行如下处理：

1. 将返回值保存到 pszProp 变量
2. 验证返回值：如果该值有效（非 NULL），则执行第 3 步。
3. 字符序列的前 13 个字符保存在 szProp 变量中。

```
{
char* pszProp = NULL;
char szProp[14];

//Get the property Tooltiptext of object IOField1 in screen Screen_1
pszProp = GetPropChar("Screen_1","IOField1","Tooltiptext");

if(pszProp != NULL)
 {
//Copy the string and trim
   strncpy(szProp,pszProp,13);
// print trimmed string
   printf ("Short description of tooltip: %s\r\n", szProp);
 }
...
}
```

保存的返回值可在后续代码中进行处理。

2.1.15.3 GetPropDouble

描述

返回数据类型"Double"的属性值。

只能在 C 脚本中使用。

语法

double GetPropDouble(ScreenName, Object, NameOfTheProperty)

参数

ScreenName

画面名称

Object

对象名称。如果函数调用与画面对象属性相关，则必须设置参数 Object = NULL。

NameOfTheProperty

对象属性的名称

返回值

数据类型“Double”的属性值。

示例

以下程序代码通过 GetPropDouble 函数读取 "BackColor" 属性（"Button_1" 对象的背景色）并对返回值进行如下处理：

1. 将返回值保存到 szprop 变量
2. 验证返回值：如果该值有效（非 NULL），则执行第 3 步。
3. 输出背景色

```
{
double szProp = NULL;

//Get the property Backcolor of object Button_1 in screen Screen_1
szProp = GetPropDouble("Screen_1","Button_1","BackColor");

if(szProp != NULL)
  {
// print output value
     printf ("Background color: %s\r\n", szProp);}
  }
...
}
```

2.1.15.4 GetPropLong

描述

返回数据类型为“long”的属性值。

只能在 C 脚本中使用。

语法

long GetPropLong(ScreenName, Object, NameOfTheProperty)

参数

ScreenName

画面名称

Object

对象名称。如果函数调用与画面对象属性相关，则必须设置参数 Object = NULL。

NameOfTheProperty

对象属性的名称

返回值

数据类型“long”的属性值。

示例

以下程序代码通过 GetPropLong 函数读取 CaptionBackColor 属性（TemperatureField 对象的背景色）并对返回值进行如下处理：

1. 将返回值保存到 szProp 变量
2. 验证返回值：如果该值有效（非 NULL），则执行第 3 步。
3. 输出

```
{
long szProp = NULL;

//Get the property CaptionBackColor of object TemperatureField in screen Screen_1
szProp = GetPropLong("Screen_1","TemperatureField","CaptionBackColor");

if(szProp != NULL)
 {
// print caption
   printf ("Caption of window: %d\r\n", szProp);
 }
...
}
```

保存的返回值可在后续代码中进行处理。

2.1.16 GetServerTagPrefix

描述

为了使分布式系统的 WinCC 客户端访问相关服务器的变量，必须对变量名进行扩展以包含服务器前缀。

说明

目前不支持此系统函数。

在每种情况下，返回针对服务器前缀、变量前缀和窗口前缀的“char”类型的指针。

用户无权更改存储器（不包括 strcat）或释放它。

只能在 C 脚本中使用。

语法

void GetServerTagPrefix(ppszServerPrefix, ppszTagPrefix, ppszWindowPrefix);

返回值

ppszServerPrefix

指向引用服务器前缀指针的指针。

ppszTagPrefix

指向引用变量前缀指针的指针。

ppszWindowPrefix

指向引用窗口前缀指针的指针。

示例

以下程序代码检索**服务器前缀**、**变量前缀**和**窗口前缀**并检查其有效性。如果出错，则输出文本并退出函数。如果检查成功，将创建并返回变量名称。处理过程以如下方式执行：

1. 声明以上三个前缀的指针 pszServerPrefix、pszTagPrefix 和 pszWindowPrefix
2. 初始化 nServerPrefixLen、nTagPrefixLen 和 nTagLen 变量
   它们作为将要读出的前缀字符串长度的缓冲区。
3. 初始化 myTagName 变量
4. 读出服务器前缀、变量前缀和窗口前缀

5. **可能出现的情况**：服务器前缀
   - **未**返回服务器前缀：输出文本并退出函数。
   - 返回服务器前缀：其长度在 nServerPrefixLen 变量中进行确定和保存。
6. 如果返回变量前缀，则其长度在 nTagPrefixLen 变量中进行确定和保存。
7. 确定变量名称的长度并将其保存到 nTagLen 变量中。
8. **可能出现的情况**：变量名称的允许长度
   - 超出允许长度：输出文本并退出函数。
   - **未**超出允许长度：编译客户端环境所需的变量名称。

```
{
char* pszServerPrefix;
char* pszTagPrefix;
char* pszWindowPrefix;
int nServerPrefixLen = 0;
int nTagPrefixLen = 0;
int nTagLen = 0;
char myTagName[MAX_DM_VAR_NAME+1];

//Initialize the return value
memset(myTagName,0,MAX_DM_VAR_NAME + 1);

//Get the serverprefix the tagprefix and the windowprefix
GetServerTagPrefix(&pszServerPrefix, &pszTagPrefix, &pszWindowPrefix);

//If a serverprefix exists
if (pszServerPrefix)
{
  //Get the length of the string
  nServerPrefixLen = strlen(pszServerPrefix);
}
Else
{
  printf("No server prefix was returned.");
  return;
}

//If a tagprefix exists
if (pszTagPrefix)
{
  //Get the length of the string
  nTagPrefixLen = strlen(pszTagPrefix);
}

//Get the length of the tag
nTagLen = strlen("TagName");

//Check if the lenght of the
//ServerPrefix+TagPrefix+VarName + the double points < MAX_DM_VAR_NAME)
if (nServerPrefixLen + nTagPrefixLen + nTagLen+2 < MAX_DM_VAR_NAME)
{
  sprintf(myTagName,"%s::%s%s",pszServerPrefix,pszTagPrefix,"TagName");
  //User defined code where the
  //user can do something with the returnvalue
  ...
}
Else
{
  printf("The resulting string is too long.");
  return;
```

```
}
}
```

2.1.17 GetTag

2.1.17.1 GetTag 函数

说明

GetTagXXX 函数计算指定数据类型的变量的值。

只能在 C 脚本中使用：

下表为可读取变量值的各类 GetTag 函数：

类型	函数名称	参数	PLC 数据类型	HMI 数据类型
布尔型	GetTagBit	Tag Tag_Name	二进制变量	布尔型
字节型	GetTagByte	Tag Tag_Name	无符号 8 位	USInt
char*	GetTagChar	Tag Tag_Name	文本变量 8 位或 文本变量 16 位	String 或 WString
SYSTEMTIME	GetTagDateTime	Tag Tag_Name	DTL	DateTime
双精度型	GetTagDouble	Tag Tag_Name	浮点数 64 位	LReal
DWORD	GetTagDWord	Tag Tag_Name	无符号 32 位	UDInt
浮点型	GetTagFloat	Tag Tag_Name	浮点数 32 位	Real
布尔型	GetTagRaw	Tag Tag_Name, BYTE* pValue, DWORD size	原始数据类型	原始类型
signed char	GetTagSByte	Tag Tag_Name	有符号 8 位	SInt
long int	GetTagSDWord	Tag Tag_Name	有符号 32 位	DInt

类型	函数名称	参数	PLC 数据类型	HMI 数据类型
short int	GetTagSWord	Tag Tag_Name	有符号 16 位	Int
WORD	GetTagWord	Tag Tag_Name	无符号 16 位	UInt

语法

<Type><FunctionName><(Parameter)>;

示例：BYTE GetTagByte(Tag_Name);

参数

Tag_Name

变量名称

pValue

包含原始数据变量值的字节字段指针。

size

字节字段的长度（以字节为单位）

返回值

指定类型的变量值。

系统函数"GetTagChar"返回包含变量值的字符串指针。

系统函数"GetTagRaw"返回 TRUE 或 FALSE：

TRUE:系统函数已成功执行，未发生任何错误。

FALSE:发生错误。

示例

以下程序代码使用 GetTagByte 函数读出 gs_tag_byte 变量的值并将其保存到 bvalue 变量。

```
{
BYTE bvalue;

//Get the current value of the tag
bvalue = GetTagByte("gs_tag_byte");

// print value
   printf ("Value of gs_tag_byte: %d\r\n", bvalue);
...
}
```

保存的返回值可在后续代码中进行处理。

2.1.17.2 GetTagDateTime 函数

功能

确定数据类型为 “日期/时间”(Date/Time) 的变量值。

语法

SYSTEMTIME GetTagDateTime(Tag_Name)；

参数

Tag_Name

变量的名称

返回值

“日期/时间”(Date/Time) 数据类型的变量值。

2.1.17.3 GetTagMultiStateQCWait 函数

描述

确定多个变量的值、状态和质量代码，以指定格式在各自的地址中保存。从 AS 中显式读取数值。

将传送两个 DWORD 型数组，以便在完成系统函数调用后，各个变量的状态和质量代码会存储在这两个数组的成员中。数组必须足够大，以为状态和质量代码提供足够的存储空间。

只能在 C 脚本中使用。

语法

BOOL GetTagMultiStateQCWait(pdwState, pdwQualityCode, pFormat);

参数

pdwState

用于在系统函数周期后为各个变量创建状态的域。

pdwQualityCode

用于在系统函数周期后为各个变量创建质量代码的域。

pFormat

所有请求变量的格式描述（类型）以及各变量值的名称和地址。

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

以下程序代码使用 GetTagMultiStateQCWait 函数读取 "gs_tag_XXX" 变量的值并将其保存到 Ivalue1、 Ivalue2 等变量。

1. “DATA_SIZE”的预处理定义（本例中为 5 个变量）
2. 创建 DWord 字段
   - dwState 变量状态字段
   - dwQc 质量代码字段
3. 针对缓冲的变量定义
4. 执行函数 GetTagMultiStateQCWait
   读取的值写入变量地址。

```
{
#define DATA_SIZE 5
DWORD dwState[DATA_SIZE];
DWORD dwQC[DATA_SIZE];

//define all Datas
BOOL lValue1;
long lValue2 ;
char* szValue3;
double dblValue4 ;
WORD lValue5 ;


//Set the tags
GetTagMultiStateQCWait(dwState,dwQC,"%d%d%s%f%d",
  "gs_tag_bit",&lValue1,
  "gs_tag_SByte",&lValue2,
  "gs_tag_char",&szValue3,
  "gs_tag_float",&dblValue4,
  "gs_tag_word",&lValue5);

//User defined code where the
//user can do something with the returnvalue
...
}
```

保存的返回值可在后续代码中进行处理。

参见

HMI 变量的质量代码 (页 1814)

常数 (页 1822)

2.1.17.4 GetTagMultiStateWait 函数

描述

确定多个变量的值和状态，以指定格式在各自的地址中保存。从 AS 中显式读取数值。

必须为系统函数提供一个 DWORD 型数组，以便在完成系统函数调用后，各个变量的状态会存储在该数组的成员中。数组必须足够大，以便为状态提供足够的存储空间。

只能在 C 脚本中使用。

语法

BOOL GetTagMultiStateWait(pdwState, pFormat);

参数

pdwState

保存变量状态的域。

pFormat

所有请求变量的格式描述（类型）以及各变量值的名称和地址。

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

以下程序代码使用 GetTagMultiStateWait 函数读取 "gs_tag_XXX" 变量的值并将其保存到 Ivalue1、 Ivalue2 等变量。

1. “DATA_SIZE”的预处理定义（本例中为 5 个变量）
2. 为变量状态创建 DWord 字段 dwState
3. 针对缓冲的变量定义
4. 执行函数 GetTagMultiStateWait
   读取的值写入变量地址。

```
{
#define DATA_SIZE 5
DWORD dwState[DATA_SIZE];

//define all Datas
BOOL lValue1;
long lValue2 ;
char* szValue3;
double dblValue4 ;
WORD lValue5 ;


//Set the tags
GetTagMultiStateWait(dwState,"%d%d%s%f%d",
  "gs_tag_bit",&lValue1,
  "gs_tag_SByte",&lValue2,
  "gs_tag_char",&szValue3,
  "gs_tag_float",&dblValue4,
  "gs_tag_word",&lValue5);

//User defined code where the
//user can do something with the returnvalue
...
}
```

保存的返回值可在后续代码中进行处理。

参见

常数 (页 1822)

2.1.17.5 GetTagMultiWait 函数

描述

确定多个变量的值，以指定格式在各自的地址中保存。从 AS 中显式读取数值。系统函数使用 SysMalloc 为变量值分配存储区。

只能在 C 脚本中使用。

语法

BOOL GetTagMultiWait(pFormat,...);

参数

pFormat

所有请求变量的格式描述以及每个变量值的名称和地址。

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

GetTagMultiWait 函数在以下程序代码中用于读取多个不同类型的变量：

1. 声明三个变量，作为三个不同变量类型的存储区
2. 声明布尔型变量 ok，用于缓冲返回值（TRUE/FALSE）
3. 读取这三个变量并将其值保存到相应的地址。
   函数的返回值保存到 ok 变量。
4. 输出带有变量前缀的三个变量

```
DWORD dwVar1Value;
char* szVar2Value;
//memory for values allocated via SysMalloc

double dbVar3Value;

BOOL ok;

ok=GetTagMultiWait("%d%s%f", "Ernie_word", &dwVar1Value,
  "Ernie_char", &szVar2Value,
  "Ernie_double", &dbVar3Value);

printf("Word %d, String %s, Double %f\r\n",
  dwVar1Value, szVar2Value, dbVar3Value);
```

2.1.17.6 GetTagState 函数

说明

GetTagStateXXX 函数计算指定数据类型的变量的值。也用于返回变量状态。

只能在 C 脚本中使用。

下表列出了可读取变量值的各类 GetTagStateXXX 函数：

类型	函数名称	参数	PLC 数据类型	HMI 数据类型
布尔型	GetTagBitState	Tag Tag_Name, PDWORD lp_dwstate	二进制变量	布尔型
字节型	GetTagByteState	Tag Tag_Name, PDWORD lp_dwstate	无符号 8 位	字节型
char*	GetTagCharState	Tag Tag_Name, PDWORD lp_dwstate	文本变量 8 位或文本变量 16 位	字符串型
双精度型	GetTagDoubleState	Tag Tag_Name, PDWORD lp_dwstate	浮点数 64 位	双精度型
DWORD	GetTagDWordState	Tag Tag_Name, PDWORD lp_dwstate	无符号 32 位	无符号整型
浮点型	GetTagFloatState	Tag Tag_Name, PDWORD lp_dwstate	浮点数 32 位	浮点型
布尔型	GetTagRawState	Tag Tag_Name, BYTE* pValue, DWORD size, PDWORD lp_dwstate	原始数据类型	原始类型
signed char	GetTagSByteState	Tag Tag_Name, PDWORD lp_dwstate	有符号 8 位	字节型
long int	GetTagSDWordState	Tag Tag_Name, PDWORD lp_dwstate	有符号 32 位	整型
short int	GetTagSWordState	Tag Tag_Name, PDWORD lp_dwstate	有符号 16 位	短整型
WORD	GetTagWordState	Tag Tag_Name, PDWORD lp_dwstate	无符号 16 位	无符号短整型

语法

<Type><FunctionName><(Parameter)>;

示例：BOOL GetTagBitState(Tag_Name, lp_dwstate);

参数

Tag_Name

变量名称

lp_dwstate

用于在系统函数周期完成时保存变量状态的 DWORD 指针。

pValue

包含原始数据变量值的字节字段指针。

size

字节字段的长度（以字节为单位）

返回值

指定类型的变量值。

系统函数“GetTagCharState”返回指向数据类型“char”的变量值的指针。

系统函数“GetTagRawState”返回 TRUE 或 FALSE：

TRUE:系统函数已成功执行，未发生任何错误。

FALSE:发生错误。

示例

以下程序代码使用 GetTagBitState 函数读出 gs_tag_bit 变量的值并将其保存到 bValue 变量。状态存储在 dwState 变量的地址中。

根据 bValue (TRUE/FALSE) 中的返回值执行特定的代码。

```
{
DWORD dwState;
BOOL bValue;

dwState = 0xFFFFFFFF;

//Get the tag value
//dwstate is the tag state
bValue = GetTagBitState("gs_tag_bit",&dwState);

//Create a string which includes the tag value
if (bValue)
{
  // User defined code if the
  // value of the tag is true
  ...
}
else
{
  // User defined code if the
  // value of the tag is false
  ...
}
}
```

参见

常数 (页 1822)

2.1.17.7 GetTagStateQC 函数

说明

GetTagStateQC 函数计算指定数据类型的变量的值。也用于返回变量的状态和质量代码。

只能在 C 脚本中使用。

下表为可读取变量值的各类 GetTagStateQC 函数：

类型	函数名称	参数	PLC 数据类型	HMI 数据类型
布尔型	GetTagBitStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	二进制变量	布尔型
字节型	GetTagByteStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	无符号 8 位	无符号字节型
char*	GetTagCharStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	文本变量 8 位或文本变量 16 位	字符串型
双精度型	GetTagDoubleStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	浮点数 64 位	双精度型
DWORD	GetTagDWordStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	无符号 32 位	无符号整型
浮点型	GetTagFloatStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	浮点数 32 位	浮点型

类型	函数名称	参数	PLC 数据类型	HMI 数据类型
布尔型	GetTagRawStateQC	Tag Tag_Name, BYTE pValue[], DWORD size, PDWORD lp_dwstate, PDWORD pdwQualityCode	原始数据类型	原始类型
signed char	GetTagSByteStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	有符号 8 位	字节型
long int	GetTagSDWordStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	有符号 32 位	整型
short int	GetTagSWordStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	有符号 16 位	短整型
WORD	GetTagWordStateQC	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	无符号 16 位	无符号短整型

语法

<Type><FunctionName><(Parameter)>;

示例：BOOL GetTagBitStateQC(Tag_Name, lp_dwstate, pdwQualityCode);

参数

Tag_Name

变量名称

lp_dwstate

用于在系统函数周期完成时保存变量状态的 DWORD 指针。

pdwQualityCode

用于在系统函数周期后为变量创建质量代码的 DWORD 指针。

pValue

包含原始数据变量值的字节字段指针。

size

字节字段的长度（以字节为单位）

返回值

指定类型的变量值。

系统函数“GetTagCharStateQC”返回指向数据类型“char”的变量值的指针。

系统函数“GetTagRawStateQC”返回 TRUE 或 FALSE：

TRUE:系统函数已成功执行，未发生任何错误。

FALSE:发生错误。

示例

以下程序代码使用 GetTagBitStateQC 函数读取 gs_tag_bit 变量的值并将其保存到 ok 变量。

状态和质量代码保存到变量的 dwState 和 dwQC 地址。

根据变量 (TRUE/FALSE) 中的返回值执行特定的代码。

```
{
DWORD dwState;
DWORD dwQC;
BOOL ok;

dwState = 0xFFFFFFFF;

//Get the tag value
//dwstate is the tag state
ok = GetTagBitStateQC("gs_tag_bit",&dwState,&dwQC);

//Create a string which includes the tag value
if (ok)
{
  // succeeded, print tag value
   printf ("Value at dwState: %x\r\n", dwState);
   printf ("Value at dwQC: %x\r\n", dwQC);
  ...
}
else
{
  // failed
  printf ( "Error - function failed." );
}
}
```

参见

HMI 变量的质量代码 (页 1814)

常数 (页 1822)

2.1.17.8 GetTagStateQCWait 函数

说明

GetTagStateQCWait 函数计算指定数据类型的变量的值。从 AS 中显式读取数值。也用于返回变量的状态和质量代码。

只能在 C 脚本中使用。

下表列出了可读取变量值的各类 GetTagStateQCWait 函数：

类型	函数名称	参数	PLC 数据类型	HMI 数据类型
布尔型	GetTagBitStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	二进制变量	布尔型
字节型	GetTagByteStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	无符号 8 位	无符号字节型
char*	GetTagCharStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	文本变量 8 位或文本变量 16 位	字符串型
双精度型	GetTagDoubleStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	浮点数 64 位	双精度型
DWORD	GetTagDWordStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	无符号 32 位	无符号整型
浮点型	GetTagFloatStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	浮点数 32 位	浮点型
布尔型	GetTagRawStateQCWait	Tag Tag_Name, BYTE pValue[], DWORD size, PDWORD lp_dwstate, PDWORD pdwQualityCode	原始数据类型	原始类型
signed char	GetTagSByteStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	有符号 8 位	字节型

类型	函数名称	参数	PLC 数据类型	HMI 数据类型
long int	GetTagSDWordStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	有符号 32 位	整型
short int	GetTagSWordStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	有符号 16 位	短整型
WORD	GetTagWordStateQCWait	Tag Tag_Name, PDWORD lp_dwstate, PDWORD pdwQualityCode	无符号 16 位	无符号短整型

语法

<Type><FunctionName><(Parameter)>;

示例：BOOL GetTagBitStateQC(Tag_Name, lp_dwstate, pdwQualityCode);

参数

Tag_Name

变量名称

lp_dwstate

用于在系统函数周期完成时保存变量状态的 DWORD 指针。

pdwQualityCode

用于在系统函数周期后为变量创建质量代码的 DWORD 指针。

pValue

包含原始数据变量值的字节字段指针。

size

字节字段的长度（以字节为单位）

返回值

指定类型的变量值。

系统函数"GetTagCharStateQCWait"返回指向数据类型"char"的变量值的指针。

系统函数"GetTagRawStateQCWait"返回 TRUE 或 FALSE：

TRUE:系统函数已成功执行，未发生任何错误。

FALSE:发生错误。

示例

以下程序代码使用 GetTagBitStateQCWait 函数读取 gs_tag_bit 变量的值并将其保存到 bValue 变量。

状态和质量代码保存到变量的 dwState 和 dwQC 地址。

根据 bValue (TRUE/FALSE) 中的返回值执行特定的代码。

```
{
DWORD dwState;
DWORD dwQC;
BOOL bValue;

dwState = 0xFFFFFFFF;

//Get the tag value
//dwstate is the tag state
bValue = GetTagBitStateQCWait("gs_tag_bit",&dwState,&dwQC);

//Create a string which includes the tag value
if (bValue)
{
  // User defined code if the
  // value of the tag is true
  ...
}
else
{
  // User defined code if the
  // value of the tag is false
  ...
}
}
```

参见

HMI 变量的质量代码 (页 1814)

常数 (页 1822)

2.1.17.9 GetTagStateWait 函数

说明

GetTagStateWait 函数计算指定数据类型的变量的值。从 AS 中显式读取数值。也用于返回变量状态。

只能在 C 脚本中使用。

下表列出了可读取变量值的各类 GetTagStateWait 函数：

类型	函数名称	参数	PLC 数据类型	HMI 数据类型
布尔型	GetTagBitStateWait	Tag Tag_Name, PDWORD lp_dwstate	二进制变量	布尔型
字节型	GetTagByteStateWait	Tag Tag_Name, PDWORD lp_dwstate	无符号 8 位	无符号字节型
char*	GetTagCharStateWait	Tag Tag_Name, PDWORD lp_dwstate	文本变量 8 位或文本变量 16 位	字符串型
双精度型	GetTagDoubleStateWait	Tag Tag_Name, PDWORD lp_dwstate	浮点数 64 位	双精度型
DWORD	GetTagDWordStateWait	Tag Tag_Name, PDWORD lp_dwstate	无符号 32 位	无符号整型
浮点型	GetTagFloatStateWait	Tag Tag_Name, PDWORD lp_dwstate	浮点数 32 位	浮点型
布尔型	GetTagRawStateWait	Tag Tag_Name, BYTE* pValue, DWORD size, PDWORD lp_dwstate	原始数据类型	原始类型
signed char	GetTagSByteStateWait	Tag Tag_Name, PDWORD lp_dwstate	有符号 8 位	字节型
long int	GetTagSDWordStateWait	Tag Tag_Name, PDWORD lp_dwstate	有符号 32 位	整型

类型	函数名称	参数	PLC 数据类型	HMI 数据类型
short int	GetTagSWordStateWait	Tag Tag_Name, PDWORD lp_dwstate	有符号 16 位	短整型
WORD	GetTagWordStateWait	Tag Tag_Name, PDWORD lp_dwstate	无符号 16 位	无符号短整型

语法

<Type><FunctionName><(Parameter)>

示例：BOOL GetTagBitStateWait(Tag_Name, lp_dwstate)

参数

Tag_Name

变量名称

lp_dwstate

用于在系统函数周期完成时保存变量状态的 DWORD 指针。

pValue

包含原始数据变量值的字节字段指针。

size

字节字段的长度（以字节为单位）

返回值

指定类型的变量值。

系统函数“GetTagCharStateWait”返回指向数据类型“char”的变量值的指针。

系统函数“GetTagRawState”返回 TRUE 或 FALSE：

TRUE:系统函数已成功执行，未发生任何错误。

FALSE:发生错误。

示例

以下程序代码使用 GetTagBitStateWait 函数读取 gs_tag_bit 变量的值并将其保存到 bValue 变量。

状态存储在 dwState 变量的地址中。

根据 bValue (TRUE/FALSE) 中的返回值执行特定的代码。

```
{
DWORD dwState;
BOOL bValue;

dwState = 0xFFFFFFFF;

//Get the tag value
//dwstate is the tag state
bValue = GetTagBitStateWait("gs_tag_bit",&dwState);

//Create a string which includes the tag value
if (bValue)
{
  // User defined code if the
  // value of the tag is true
  ...
}
else
{
  // User defined code if the
  // value of the tag is false
  ...
}
}
```

参见

常数 (页 1822)

2.1.17.10 GetTagValue 函数

描述

以变量形式传送值。确定含有值的结果结构的指针。

只能在 C 脚本中使用。

语法

BOOL GetTagValue(lpdmVarKey, lpdmresult, lpdmError);

参数

lpdmVarKey

指向数据类型"DM_VARKEY"结构的指针

lpdmresult

指向数据类型"DM_VAR_UPDATE_STRUCT"结构的指针

lpdmError

包含错误描述结构的指针。

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

GetTagValue 函数在以下程序代码中用于计算 varKey 中的值。

根据 keyFound (TRUE/FALSE) 中的返回值执行特定的代码。

```
{
DM_VARKEY varKey;
DM_VAR_UPDATE_STRUCT result;
CMN_ERROR error:

BOOL keyFound;

//Get the tag value
keyFound = GetTagValue(&varKey, &result, &error);

if (keyFound)
{
  // print tag value
   printf ("Value of varKey: %d\r\n", &varKey);
  ...
}
else
{
  // failed
  printf ( "Error - function failed." );
  ...
}
}
```

2.1.17.11 GetTagValueStateQC 函数

描述

以变量形式传送值。确定含有值的结果结构的指针。也用于返回变量的状态和质量代码。

该系统函数只能用于 C 脚本。

语法

BOOL GetTagValueStateQC(lpdmVarKey, lpdmresult, lpdmError);

参数

lpdmVarKey

指向数据类型“DM_VARKEY”结构的指针

lpdmresult

指向数据类型"DM_VAR_UPDATE_STRUCTEX"结构的指针

lpdmError

包含错误描述结构的指针。

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

GetTagValueStateQC 函数在以下程序代码中用于计算 varKey 中的值。

状态和质量代码保存到变量的 dwState 和 dwQC 地址。

根据 keyFound (TRUE/FALSE) 中的返回值执行特定的代码。

```
{
DM_VARKEY varKey;
DM_VAR_UPDATE_STRUCTEX result;
CMN_ERROR error:
// clarify (DM_VAR_UPDATE_STRUCTEX already contains QualityCode)
DWORD dwState;
DWORD dwQC;

BOOL keyFound;

//Get the tag value
keyFound = GetTagValueStateQC(&varKey, &result, &error,&dwState,&dwQC);

if (keyFound)
{
  // User defined code if the
  // value of the tag is true
  ...
}
else
{
  // User defined code if the
  // value of the tag is false
  ...
}
}
```

2.1.17.12 GetTagValueStateQCWait 函数

描述

以变量形式传送值。确定含有值的结果结构的指针。从 AS 中显式读取数值。也用于返回变量的状态和质量代码。

只能在 C 脚本中使用。

语法

BOOL GetTagValueStateQCWait(lpdmVarKey, lpdmresult, lpdmError);

参数

lpdmVarKey

指向数据类型"DM_VARKEY"结构的指针

lpdmresult

指向数据类型"DM_VAR_UPDATE_STRUCTEX"结构的指针

lpdmError

包含错误描述结构的指针。

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

以下程序代码使用函数 GetTagValueStateQCWait 进行读取

状态和质量代码保存到变量的 dwState 和 dwQC 地址。

根据 keyFound (TRUE/FALSE) 中的返回值执行特定的代码。

```
{
DM_VARKEY varKey;
DM_VAR_UPDATE_STRUCTEX result;
CMN_ERROR error:
// clarify (DM_VAR_UPDATE_STRUCTEX already contains QualityCode)
DWORD dwState;
DWORD dwQC;

BOOL keyFound;

//Get the tag value
keyFound = GetTagValueStateQC(&varKey, &result, &error,&dwState,&dwQC);

if (keyFound)
{
  // User defined code if the
  // value of the tag is true
  ...
}
else
{
  // User defined code if the
  // value of the tag is false
  ...
}
}
```

2.1.17.13 GetTagValueWait 函数

描述

以变量形式传送值。确定含有值的结果结构的指针。直接从 AS 读取值。

只能在 C 脚本中使用。

语法

BOOL GetTagValueWait(lpdmVarKey, lpdmresult, lpdmError);

参数

lpdmVarKey

指向数据类型“DM_VARKEY”结构的指针

lpdmresult

指向数据类型"DM_VAR_UPDATE_STRUCT"结构的指针

lpdmError

关于包含错误描述的结构的指针。

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

GetTagValueWait 函数在以下程序代码中用于计算 varKey 中的值。

根据 keyFound (TRUE/FALSE) 中的返回值执行特定的代码。

```
{
DM_VARKEY varKey;
DM_VAR_UPDATE_STRUCT result;
CMN_ERROR error:

BOOL keyFound;

//Get the tag value
keyFound = GetTagValueWait(&varKey, &result, &error);

if (keyFound)
{
  // succeeded, key found
  // print tag value
   printf ("Value of varKey: %d\r\n", &varKey);
  ...
}
else
{
  // failed
  printf ( "Error - function failed." );
}
  ...
}
```

2.1.17.14 GetTagWait 函数

描述

GetTagWaitXXX 函数计算指定数据类型的变量的值。从 AS 中显式读取数值。

只能在 C 脚本中使用。

下表为可读取变量值的各类 GetTagWait 函数：

类型	函数名称	参数	PLC 数据类型	HMI 数据类型
布尔型	GetTagBitWait	Tag Tag_Name	二进制变量	布尔型
字节型	GetTagByteWait	Tag Tag_Name	无符号 8 位	无符号字节型
char*	GetTagCharWait	Tag Tag_Name	文本变量 8 位或文本变量 16 位	字符串型
双精度型	GetTagDoubleWait	Tag Tag_Name	浮点数 64 位	双精度型
DWORD	GetTagDWordWait	Tag Tag_Name	无符号 32 位	无符号整型
浮点型	GetTagFloatWait	Tag Tag_Name	浮点数 32 位	浮点型
布尔型	GetTagRawWait	Tag Tag_Name, BYTE* pValue, DWORD size	原始数据类型	原始类型
char	GetTagSByteWait	Tag Tag_Name	有符号 8 位	字节型
long int	GetTagSDWordWait	Tag Tag_Name	有符号 32 位	整型
short int	GetTagSWordWait	Tag Tag_Name	有符号 16 位	短整型
WORD	GetTagWordWait	Tag Tag_Name	无符号 16 位	无符号短整型

语法

<Type><FunctionName><(Parameter)>;

示例：BYTE GetTagByteWait(Tag_Name);

参数

Tag_Name

变量名称

pValue

包含原始数据变量值的字节字段指针。

返回值

指定类型的变量值。

系统函数"GetTagCharWait"返回包含变量值的字符串指针。

系统函数"GetTagRawWait"返回 TRUE 或 FALSE：

TRUE:系统函数已成功执行，未发生任何错误。

FALSE:发生错误。

示例

以下程序代码使用 GetTagByteWait 函数读取 gs_tag_byte 变量的值并将其保存到 bvalue 变量。

```
{
BYTE bvalue;

//Get the current value of the tag
bvalue = GetTagByteWait("gs_tag_byte");

// print value
   printf ("Value of gs_tag_byte: %d\r\n", bvalue);
...
}
```

保存的返回值可在后续代码中进行处理。

2.1.18 IncreaseTag

描述

将给定值添加到变量值上。

X = X + a

说明

系统函数使用同一变量作为输入和输出值。当该系统函数用于转换数值时，必须使用辅助变量。可以使用“SetTag”系统函数为辅助变量指定变量值。

如果为报警事件组态了系统函数，但变量未在当前画面中使用，则无法确定在 PLC 中使用的实际变量值。通过设置“连续循环”采集模式可以改善这种情况。

在函数列表中使用

增加变量（变量，值）

在用户自定义函数中使用

IncreaseTag Tag, Value

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“AUTOHOTSPOT”。

参数

变量

为其添加给定值的变量。

值

作为加数的数值。

示例

下面的程序代码会以 value 变量的值为增量对 varX 变量的值进行递增。输入的值将保存在 old_value 变量中，并与新的 varX 值一起输出。

```
{
BYTE varX;
BYTE value;

//user input
...
BYTE old_value = varX;

//Increase tag
IncreaseTag(varX, value);

//print original value and function result
printf ("User input: %i\r\n, Result of function IncreaseTag: %i\r\n", old_value, varX);
...
}
```

2.1.19 InquireLanguage

描述

确定在运行系统文本库中组态的所有语言。

使用 Pointer to a counter 指定语言 ID 的存储位置。

该系统函数只能用于 C 脚本。

语法

DWORD* InquireLanguage(PointerToACounter);

参数

PointerToACounter

指向所找到的语言 ID 编号的指针

返回值

指向包含已确定语言 ID 的域的指针。

可以应用以下赋值（十六进制语言代码）：

符号名称	值（十六进制）	缩写
LANG_ARABIC	0x0401	
LANG_AFRIKAANS	0x0436	
LANG_ALBANIAN	0x041C	
LANG_BASQUE	0x042D	
LANG_BULGARIAN	0x0402	
LANG_BYELORUSSIAN	0x0423	
LANG_CATALAN	0x0403	
LANG_CHINESE	0x0404	
LANG_CROATIAN	0x041A	
LANG_CZECH	0x0405	CSY
LANG_DANISH	0x0406	DAN
LANG_DUTCH	0x0413	NLD
LANG_ENGLISH	0x0409	ENU
LANG_ESTONIAN	0x0425	
LANG_FAEROESE	0x0438	
LANG_FARSI	0x0429	
LANG_FINNISH	0x040B	FIN
LANG_FRENCH	0x040C	FRA
LANG_GERMAN	0x0407	DEU
LANG_GREEK	0x0408	
LANG_HEBREW	0x040D	
LANG_HUNGARIAN	0x040E	HUN
LANG_ICELANDIC	0x040F	ISL
LANG_INDONESIAN	0x0421	
LANG_ITALIAN	0x0410	ITA
LANG_JAPANESE	0x0411	
LANG_KOREAN	0x0412	

符号名称	值（十六进制）	缩写
LANG_LATVIAN	0x0426	
LANG_LITHUANIAN	0x0427	
LANG_NORWEGIAN	0x0414	NOR
LANG_POLISH	0x0415	PLK
LANG_PORTUGUESE	0x0416	PTB
LANG_ROMANIAN	0x0418	
LANG_RUSSIAN	0x0419	RUS
LANG_SLOVAK	0x041B	SKY
LANG_SLOVENIAN	0x0424	
LANG_SORBIAN	0x042E	
LANG_SPANISH	0x040A	ESP
LANG_SWEDISH	0x041D	SVE
LANG_THAI	0x041E	
LANG_TURKISH	0x041F	TRK
LANG_UKRAINIAN	0x0422	

示例

以下程序代码使用 InquireLanguage 函数查询在运行时组态的语言并对其进行如下处理：

1. 将查询的语言 ID 保存到变量 language
2. 将语言的数量保存到变量 count
3. 设置语言 ID 和数量的输出格式

```
{
DWORD count;
DWORD* language;
int i;

//Count the installed languages
language = InquireLanguage(&count);

printf("#################### INQUIRE LANGUAGE ####################");
//Print out the count of languages
printf ( "\r\nCount Languages=%d\r\n", count );

//print out which languages are installed
for (i=1;i<=count; i++)
{
printf ("\r\n%d.language=%x", i,*language++);
}
}
```

保存的返回值可在后续代码中进行处理。

2.1.20 InverseLinearScaling

描述

使用线性函数 X = (Y - b) / a，将通过给定变量 Y 的值计算得出的数值赋给变量 X。

变量 X 和 Y 不能相同。与此函数相反的系统函数是“线性转换”。

说明

变量 X 和 Y 不能相同。如果要将变量转换为其自身，必须使用一个辅助变量。
可用系统函数“设置变量”将要转换变量的值分配给辅助变量。

在函数列表中使用

转换线性转换 (X, Y, b, a)

在用户自定义函数中使用

InverseLinearScaling X, Y, b, a

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

X

要为其分配通过线性方程式计算得出的值的变量。

Y

包含用于计算的值的变量。

b

其值作为减数。

a

其值作为除数。

示例

下面的程序代码使用 InverseLinearScaling 函数为 varX 变量赋值。

```
{
BYTE varX;
BYTE Yvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

//Inverse linear scaling
 InverseLinearScaling (varX, Yvalue, bvalue, avalue);

printf ("varX = %d\r\n, varX);
...
}
```

保存的返回值可在后续代码中进行处理。

2.1.21 InvertBit

描述

对给定的“Bool”型变量的值取反。

- 如果变量具有值 1（真），它将被设置为 0（假）。
- 如果变量具有值 0（假），它将被设置为 1（真）。

在函数列表中使用

对位取反（变量）

在用户自定义函数中使用

InvertBit Tag

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

要设置其位的变量。

示例

以下程序代码会将布尔型变量 bStatus 的值取反，并将所得结果与原始 bSaved 值一起输出。

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Get current value
bValue=myTag.Value

'Save current value
bSaved=bValue

'Invert Bit
InvertBit myTag
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BOOL bStatus = 0;
BOOL bSaved = bStatus;

//Invert variable
invertBit(bStatus);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
  ...
}
```

2.1.22 InvertBitInTag

描述

对给定变量中的位取反：

- 如果变量中的位为值 1（真），它将被设置为 0（假）。
- 如果变量中的位为值 0（假），它将被设置为 1（真）。

在改变了给定位之后，系统函数将整个变量传送回 PLC。但是并不检查变量中的其它位是否同时改变。在所示变量被传回 PLC 之前，操作员和 PLC 对该变量只有只读权限。

说明

如果 PLC 支持 BOOL 型变量，不要使用该系统函数。而是使用“对位取反”系统函数。

在函数列表中使用

对变量中的位取反（变量，位）

在用户自定义函数中使用

InvertBitInTag Tag, Bit

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

要设置其给定位的变量。

位

要设置的位的编号。

在用户自定义函数中使用此系统函数时，变量中的位从右向左计数。计数从 0 开始。

示例

以下程序代码会将 bStatusWord 变量中指定的 bitposition 位取反，并将所得结果与原始 bSaved 值一起输出。

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Get current value
bValue=myTag.Value

'Save current value
bSaved=bValue

'Invert Bit in position
bitposition=2
InvertBit myTag, bitposition
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BYTE bStatusWord;
BYTE bsaved = bStatusWord;
BYTE bitposition = 2;

//Invert bit in bitposition
InvertBitInTag (bStatusWord, bitposition);
//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatusWord, bsaved);
  ...
}
```

2.1.23 IsUserAuthorized

描述

用户身份验证。

只能在 C 脚本中使用。

语法

BOOL IsUserAuthorized(AuthorizationNumber);

参数

AuthorizationNumber

要确认的身份验证（数字）。

返回值

TRUE

用户具有指定的授权。

FALSE

用户不具有指定的授权。

示例

以下程序代码通过 IsUserAuthorized 函数对用户身份进行验证并将值写入布尔型变量 ok。

```
{
BOOL ok;
DWORD authnumber;

//Check authorization and return value
ok = IsUserAuthorized (authnumber);

//error handling
if(ok)
{
  // user authorized
  printf ( "User is authorized." );
}
else
{
  // user not authorized
  printf ( "Authorization failed." );
}
...
}
```

返回值即为输出结果。

2.1.24 LinearScaling

描述

为变量 Y 赋值，该变量通过线性函数 Y= (a *X) + b 利用给定变量 X 的值计算得出。

与此功能相反的系统函数是“转换线性转换”。

说明

变量 X 和 Y 不能相同。如果要将变量转换为其自身，必须使用一个辅助变量。
可用系统函数“设置变量”将要转换变量的值分配给辅助变量。

在函数列表中使用

线性转换 (Y, a, X, b)

在用户自定义函数中使用

LinearScaling Y, a, X, b

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

Y

要为其分配通过线性方程式计算得出的值的变量。

a

作为乘数的数值。

X

包含用于计算的值的变量。

b

作为加数的数值。

示例

下面的程序代码使用 LinearScaling 函数为 Yvar 变量赋值。

```
{
BYTE Yvar;
BYTE Xvalue = 10;
BYTE bvalue = 3;
BYTE avalue = 4;

// linear scaling
 LinearScaling ( Yvar, avalue, Xvalue, bvalue);

printf ("Yvar = %d\r\n, Yvar);
...
}
```

保存的返回值可在后续代码中进行处理。

2.1.25 ReportJob

描述

根据 NameOfMethod 参数值，启动打印作业或打印作业预览。

只能在 C 脚本中使用。

语法

void ReportJob(PrintJobName, NameOfMethod)

参数

PrintJobName

指向打印作业名称的指针。

NameOfMethod

定义是否起动打印作业或打印作业预览：

- PRINTJOB：启动打印作业。
- PREVIEW：打印作业的预览已启动。

示例

以下程序代码针对 printmethod 变量中的各项内容执行预览或打印作业。printmethod 变量由连接至 strPrintJobMethod 变量的 I/O 域读取。

```
{
char* pszPrintjobName;
char* printmethod;

//Get the print method
  printf("Input PRINTJOB for printing or PREVIEW for a quick view");
  printmethod = GetTagChar("strPrintJobMethod")

//Print job or show preview
ReportJob(&PrintjobName, printmethod);

//error handling
if(printmethod=="PRINTJOB")
{
  // message for printing completed
  printf("printing done");
  ...
}
else
{
  // User defined code if the
  // job is a preview or failed
  ...
}
}
```

2.1.26 ResetBit

描述

将“Bool”型变量的值设置为 0（假）。

在函数列表中使用

复位（变量）

在用户自定义函数中使用

ResetBit Tag

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

设置为 0（假）的 BOOL 型变量。

示例

以下程序代码使用 ResetBit 函数将布尔型变量 bStatus 的值置 0，并将所得结果与原始 bSaved 值一起输出。

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Set value
bValue=1
myTag.Value=bValue

'Save current value
bSaved=bValue

'Reset Bit
ResetBit myTag
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BOOL bStatus = 1;
BOOL bSaved = bStatus;

//Reset bit
  ResetBit (bStatus);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
...
}
```

2.1.27 ResetBitInTag

描述

将指定变量中的一个位设置为 0（假）。

在改变了给定位之后，系统函数将整个变量传送回 PLC。但是并不检查变量中的其它位是否同时改变。在所示变量被传回 PLC 之前，操作员和 PLC 对该变量只有只读权限。

说明

如果 PLC 支持 BOOL 型变量，不要使用该系统函数。而使用系统函数“复位”。

在函数列表中使用

复位变量中的位（变量，位）

在用户自定义函数中使用

ResetBitInTag Tag, Bit

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

其中一个位要设置为 0（假）的变量。

位

要设置为 0（假）的位的编号。

在用户自定义函数中使用此系统函数时，无论使用何种 PLC，指定变量中的位都是从右向左计数。计数从 0 开始。

示例

以下程序代码会将 bStatusWord 变量中指定的 bitposition 位置 0，并将所得结果与原始 bSaved 值一起输出。

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Save current value
bValue=myTag.Value
bSaved=bValue

'Reset Bit
bitposition=2
ResetBitInTag myTag, bitposition
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{

BYTE bSaved;
BYTE bitposition = 2;

bSaved = GetTagByte("bStatusWord");

//Reset bit in bitposition
ResetBitInTag ("bStatusWord", bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",GetTagByte("bStatusWord"), bSaved);
  ...
}
```

2.1.28 Set_Focus

函数

设置指定对象的焦点。

语法

BOOL Set_Focus(lpszPictureName, lpszObjectName)

参数

lpszPictureName

画面名称

lpszObjectName

对象名称

返回值

TRUE

已执行完函数，未发生任何错误。

FALSE

发生错误。

2.1.29 SetBit

描述

将"Bool"型变量的值设置为 1（真）。

在函数列表中使用

置位（变量）

在用户自定义函数中使用

SetBit Tag

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

要将其值设置为 1（真）的 BOOL 型变量。

示例

以下程序代码使用 SetBit 函数将布尔型变量 bStatus 的值置 1，并将所得结果与原始 bSaved 值一起输出。

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, strResult

Set myTag = SmartTags("bStatus")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Set value
bValue=0
myTag.Value=bValue

'Save current value
bSaved=bValue

'Set Bit
SetBit myTag
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "&bSaved &Chr(13)&"New Value: "&bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{
BOOL bStatus = 0;
BOOL bSaved = bStatus;

//Set bit
  SetBit (bStatus);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",bStatus, bSaved);
...
}
```

2.1.30 SetBitInTag

描述

将给定变量中的一个位设置为 1（真）。

在改变了给定位之后，系统函数将整个变量传送回 PLC。但是并不检查变量中的其它位是否同时改变。在所示变量被传回 PLC 之前，操作员和 PLC 对该变量只有只读权限。

说明

如果 PLC 支持 BOOL 型变量，不要使用该系统函数。而使用系统函数“置位”。

在函数列表中使用

置位变量中的位（变量，位）

在用户自定义函数中使用

SetBitInTag Tag, Bit

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

要将其中的一个位设置为 1（真）的变量。

位

要设置为 1（真）的位的编号。

在用户自定义函数中使用此系统函数时，无论使用何种 PLC，指定变量中的位都是从右向左计数。计数从 0 开始。

说明

要实现可靠的功能，必须保证与实际过程值一起使用的变量的更新。因此，应在 I/O 域中组态变量或将系统函数分配给画面对象（如按钮）。

如果为系统函数组态了短事件（如激活报警），则只能通过设置连续读取的变量访问实际过程值。

示例

以下程序代码会将 bStatusWord 变量中指定的 bitposition 位置 1，并将所得结果与原始 bSaved 值一起输出。

```
'Programming language: VB
Dim myTag
Dim myOutputField
Dim bValue, bSaved, bitposition, strResult

Set myTag = SmartTags("bStatusWord")
Set myOutputField=HMIRuntime.Screens("MyScreen").ScreenItems("objTextField")

'Save current value
bValue=myTag.Value
bSaved=bValue

'Set Bit in tag
bitposition=1
SetBitInTag "bStatusWord", bitposition
bValue=myTag.Value

'Output result old and new value:
strResult="Old Value: "& bSaved & "New Value: " & bValue
myOutputField.Text=strResult
```

```
//Programming language: C
{

BYTE bSaved;
BYTE bitposition = 1;

bSaved = GetTagByte("bStatusWord");

//Reset bit in bitposition
SetBitInTag ("bStatusWord", bitposition);

//print current and saved value
printf ("Current value: %d\r\n, Saved value: %d\r\n",GetTagByte("bStatusWord"), bSaved);

}
```

2.1.31 SetLanguageByLocaleID

描述

更改运行系统的语言设置。

只能在 C 脚本中使用。

语法

BOOL SetLanguageByLocaleID(dwLocaleID);

参数

dwLocaleID

要设置语言的语言 ID

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

以下程序代码使用 SetLanguage 函数将当前运行系统语言设置为德语并将返回值保存到 ok 变量。

```
{
BOOL ok;
DWORD old_language;
DWORD new_language;

//Get the current language and save it
old_language = GetLanguageByLocaleID ();

//Set the current language to German
ok = SetLanguageByLocaleID (0x0407);

//Get the current language and save it
new_language = GetLanguageByLocaleID ();

//error handling
if(ok)
{
  // succeeded
  printf ( "RT language is now German." );
}
else
{
  // failed
  printf ( "RT language was not updated." );
}
//print language code
printf ("Former language code: %d\r\n", old_language);
printf ("Current language code: %d\r\n", new_language);
}
```

保存的返回值可在后续代码中进行处理。

参见

GetLanguageByLocaleID (页 1622)

2.1.32 SetLanguageByName

说明

通过要设定的语言的国家名称更改运行时的语言设置。

只能在 C 脚本中使用。

在函数列表中使用

设置语言（语言））

语法

Bool SetLanguageByName(lpszLanguage);

参数

lpszLanguage

根据"RFC 1766"格式的语言缩略的指针。

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

2.1.33 SetLink

功能

创建对象属性的变量连接

语法

BOOL SetLink(LPCTSTR lpszPictureName, LPCTSTR lpszObjectName, LPCTSTR lpszPropertyName, LPLINKINFO *pLink);

参数

lpszPictureName

画面名称

lpszObjectName

对象名称

lpszPropertyName

对象属性的名称

pLink

指向类型结构的指针：LINKINFO

返回值

TRUE

已执行完函数，未发生任何错误。

FALSE

发生错误。

参见

结构定义 LINKINFO (页 1802)

2.1.34 SetProp

2.1.34.1 SetPropBOOL

描述

设置数据类型为“BOOL”的对象属性。

只能在 C 脚本中使用。

语法

BOOL SetPropBOOL(ScreenName, Object, NameOfTheProperty, Value)

参数

ScreenName

画面名称

Object

对象名称。如果函数调用与画面对象属性相关，则必须设置参数 Object = NULL。

NameOfTheProperty

对象属性的名称

Value

分配给数据类型为"BOOL"对象属性的值。

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

以下程序代码使用 SetPropBool 函数将 gs_graph_iofield 对象的属性设置为“可见”。返回值将保存到 ok 变量。

```
{
BOOL ok;

//Set the visibility TRUE
ok = SetPropBOOL("gs_graph_iofield","IOField1","Visible",TRUE);

//error handling
if(ok)
{
  // succeeded
  printf ( "IO field is visible." );
}
else
{
  // failed
  printf ( "Error - visibility not set" );
}
...
}
```

保存的返回值可在后续代码中进行处理。

2.1.34.2 SetPropChar

描述

设置数据类型“char”的对象属性。

只能在 C 脚本中使用。

语法

BOOL SetPropChar(ScreenName, Object, NameOfTheProperty, Value)

参数

ScreenName

画面名称

Object

对象名称。如果函数调用与画面对象属性相关，则必须设置参数 Object = NULL。

NameOfTheProperty

对象属性的名称

Value

分配给数据类型“char”对象属性的值。

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

以下程序代码使用 SetPropChar 函数将 gs_graph_iofield 对象的 Tooltiptext 属性设置为“Tooltiptext 1”值。返回值将保存到 ok 变量。

```
{
BOOL ok;

//Set the property Tooltiptext
ok = SetPropChar("Screen_1","gs_graph_iofield","ToolTipText","Tooltiptext 1");

//error handling
if(ok)
{
  // succeeded
  printf ( "Property of Tooltiptext is now Tooltiptext 1." );
}
else
{
  // failed
  printf ( "Error - property not set" );
}
...
}
```

保存的返回值可在后续代码中进行处理。

2.1.34.3 SetPropDouble

描述

设置数据类型“Double”的对象属性。

只能在 C 脚本中使用。

语法

BOOL SetPropDouble(ScreenName, Object, NameOfTheProperty, Value)

参数

ScreenName

画面名称

Object

对象名称。如果函数调用与画面对象属性相关，则必须设置参数 Object = NULL。

NameOfTheProperty

对象属性的名称

Value

分配给数据类型“double”对象属性的值。

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

以下程序代码访问 screenName 画面中的对象属性。在本示例中，SetPropDouble 函数将“Circle_1”的属性（半径）设置为值 10。返回值将保存到 ok 变量。

```
{
BOOL ok;


//Set the property of circle
ok = SetPropDouble(screenName,"Circle_1","Radius",10);

//error handling
if(ok)
{
  // succeeded
  printf ( "Radius was set." );
}
else
{
  // failed
  printf ( "Error - radius not set" );
}
...
}
```

保存的返回值可在后续代码中进行处理。

2.1.34.4 SetPropLong

描述

设置数据类型“long”的对象属性。

只能在 C 脚本中使用。

语法

BOOL SetPropLong(ScreenName, Object, NameOfTheProperty, Value)

参数

ScreenName

画面名称

Object

对象名称。如果函数调用与画面对象属性相关，则必须设置参数 Object = NULL。

NameOfTheProperty

对象属性的名称

Value

分配给数据类型“long”对象属性的值。

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

以下程序代码使用 SetPropLong 函数更改对象的前景色：在“Screen_1”中，将“Button1”对象的“ForeColor”属性设置为值 65333 （红色）。返回值将保存到 ok 变量。

```
{
BOOL ok;

//Set the property Tooltiptext
ok = SetPropLong("Screen_1","Button1","ForeColor",65333);

//error handling
if(ok)
{
  // succeeded
  printf ( "Color was set." );
}
else
{
  // failed
  printf ( "Error - color not set" );
}
...
}
```

保存的返回值可在后续代码中进行处理。

2.1.35 SetPropertyByConstant

说明

指定对象属性值为字符串。

在函数列表中使用

按常量设置属性（画面名称，画面对象，属性名称，值）

在用户自定义函数中使用

SetPropertyByConstant Screen_name, Screen_object, Property_name, Value

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见 “AUTOHOTSPOT”。

如要更改画面的属性，则参数“对象”必须为空。为此，请使用以下语法，例如：

SetPropertyByConstant Screen_name, Property_name, Value

参数

画面名称

包含对象的画面的名称。

画面对象

更改其属性的对象名称。

属性名称

要更改的属性名称。

值

分配给属性的值。

示例

下面的程序代码将使用 SetPropertyByConstant 函数更改对象属性：在“Trends”画面中，“Control_1”对象的“ToolbarButtonClick”属性设置为值 26。

```
'Programming language: VBS
'Name of the picture: Trends
'Name of the f(t) trend view control: Control_1

SetPropertyByConstant "Trends", "Control_1", "ToolbarButtonClick", "26"

'User defined code
...
```

```
{
//Programming language: C
//Name of the picture: Trends
//Name of the f(t) trend view control: Control_1

SetPropertyByConstant ("Trends", "Control_1", "ToolbarButtonClick", "26");

// User defined code
...
}
```

示例：更改画面属性

下面的程序代码将使用 SetPropertyByConstant 函数更改画面属性：在“Trends”画面中，“Backcolor”属性设置为值 255。

```
'Programming language: VBS
'Name of the picture: Trends

SetPropertyByConstant "Trends", "Trends", "Backcolor", "255"

'User defined code
...
```

```
{
//Programming language: C
//Name of the picture: Trends

SetPropertyByConstant ("Trends", "Trends", "Backcolor", "255");

// User defined code
...
}
```

此外，还可以使用密码 ZERO 或空格字符串代替第二个参数（对象）。

2.1.36 SetPropertyByProperty

说明

通过其它的对象属性指定某一对象属性的值。

在函数列表中使用

SetPropertyByProperty（画面名称，对象，属性名称，目标画面名称，目标画面对象，目标属性名称）

在用户自定义函数中使用

SetPropertyByProperty Screen_name, Screen_object, Property_name, Source_screen_name, Source_screen_object, Source_property_name

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“AUTOHOTSPOT”。

如果想使用另一个画面属性来指定某个画面的属性，则参数“对象”和“目标对象”必须为空。为此，请使用以下语法，例如：

SetPropertyByProperty Screen_name, Property_name, Source_screen_name, Source_property_name

参数

画面名称

包含对象的画面的名称。

对象

属性值要传递给目标对象的对象名称。

属性名称

值要传递给目标对象的属性名称。

目标画面名称

包含目标对象的画面的名称。

目标画面对象

更改属性的目标对象的名称。

目标属性名称

要更改的属性名称。

示例

以下程序代码使用 SetPropertyByProperty 函数将原始画面"Trend_1"中对象 "Control_1" 的属性"ToolbarButtonClick"设置成目标画面"Trend_2"中的相应属性。

```
'Programming language: VBS
'Name of source picture: Trend_1
'Name of target picture: Trend_2
'Name of the f(t) trend view control: Control_1

SetPropertyByProperty "Trend_1", "Control_1", "ToolbarButtonClick", "Trend_2",
"Control_2", "ToolbarButtonClick"

'User defined code
...
```

```
{
//Programming language: C
//Name of source picture: Trend_1
//Name of target picture: Trend_2
//Name of the f(t) trend view control: Control_1

SetPropertyByProperty ("Trend_1", "Control_1", "ToolbarButtonClick", "Trend_2",
"Control_2", "ToolbarButtonClick");

// User defined code
...
}
```

2.1.37 SetPropertyByTag

说明

通过变量值指定对象属性的值。

在函数列表中使用

按变量设置属性（画面名称，画面对象，属性名称，变量名称）

在用户自定义函数中使用

SetPropertyByTag Screen_name, Screen_object, Property_name, Tag_name

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“AUTOHOTSPOT”。

如果要通过变量值指定画面的属性，则“对象”参数必须为空。为此，请使用以下语法，例如：

SetPropertyByTag Screen_name, Property_name, Tag_name

参数

画面名称

包含对象的画面的名称。

画面对象

要使用变量值设置其属性的对象的名称。

属性名称

使用变量值设置的属性的名称。

变量名称

包含属性值的变量名称。

示例

下面的程序代码将使用 SetPropertyByTag 函数更改对象属性：单击对象时，可传送对象名称和包含对象的画面。画面窗口中的 CaptionText 包含 HMI_value_1 变量的值。

```
'Programming language: VBS
SetPropertyByTag screenName, objectName, "CaptionText", "HMI_value_1"

'User defined code
...
```

```
{
//Programming language: C
SetPropertyByTag (screenName, objectName, "CaptionText", "HMI_value_1");

// User defined code
...
}
```

示例

下面的程序代码将使用 SetPropertyByTag 函数更改对象属性：在“Trends”画面中，“Control_1”对象的“ToolbarButtonClick”属性设置为值 26。

```
'Programming language: VBS
'Name of the picture: Trends
'Name of the f(t) trend view control: Control_1

SetPropertyByConstant "Trends", "Control_1", "ToolbarButtonClick", "26"

'User defined code
...
```

```
{
//Programming language: C
//Name of the picture: Trends
//Name of the f(t) trend view control: Control_1

SetPropertyByConstant ("Trends", "Control_1", "ToolbarButtonClick", "26");

// User defined code
...
}
```

2.1.38 SetPropertyByTagIndirect

说明

将间接寻址的变量值写入对象属性。作为参数传送的变量包含其值被读取的另一变量的名称。

在函数列表中使用

按间接变量设置属性（画面名称，画面对象，属性名称，变量名称）

在用户自定义函数中使用

SetPropertyByTagIndirect Screen_name, Screen_object, Property_name, Tag_name

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“AUTOHOTSPOT”。

如果要通过变量指定画面的属性，则“对象”参数必须为空。为此，请使用以下语法，例如：

SetPropertyByTagIndirect Screen_name, Property_name, Tag_name

参数

画面名称

包含对象的画面的名称。

画面对象

要更改其属性的对象的名称。

属性名称

要更改的属性的名称。

变量名称

包含其值被读取的其它变量名称的变量的名称。

示例

下面的程序代码将使用 SetPropertyByTagIndirect 函数更改对象属性：

```
'Programming language: VBS
SetPropertyByTagIndirect GetParentScreen(screenName), GetParentScreenWindow(screenName),
"ScreenName", "HMI_value_1"

'User defined code
...
```

```
{
//Programming language: C
SetPropertyByTagIndirect (GetParentScreen(screenName), GetParentScreenWindow(screenName),
"ScreenName", "HMI_value_1");

// User defined code
...
}
```

2.1.39 SetTag

2.1.39.1 SetTag 函数

说明

SetTagXXX 函数计算指定数据类型的变量的值。

只能在 C 脚本中使用。

下表为用于设置变量值的各类 SetTag 函数：

函数名称	参数	PLC 数据类型	HMI 数据类型
SetTagBit	Tag Tag_Name, short int value	二进制变量	布尔型
SetTagByte	Tag Tag_Name, BYTE value	无符号 8 位	USInt
SetTagDateTime	Tag Tag_Name, SYSTEMTIME value	DTL	DateTime
SetTagChar	Tag Tag_Name, LPSTR value	文本变量 8 位或文本变量 16 位	String 或 WString
SetTagDouble	Tag Tag_Name, double value	浮点数 64 位	LReal
SetTagDWord	Tag Tag_Name, DWORD value	无符号 32 位	UDInt
SetTagFloat	Tag Tag_Name, float value	浮点数 32 位	Real
SetTagRaw	Tag Tag_Name, BYTE* pValue, DWORD size	原始数据类型	原始类型
SetTagSByte	Tag Tag_Name, char value	有符号 8 位	SInt
SetTagSDWord	Tag Tag_Name, long int value	有符号 32 位	DInt
SetTagSWord	Tag Tag_Name, short int value	有符号 16 位	Int
SetTagWord	Tag Tag_Name, WORD value	无符号 16 位	UInt

说明

使用 WString 类型变量时的错误消息

如果在已转换为 C 脚本的 SetTag 系统函数中使用 WString 类型变量，则执行函数时将在全局脚本诊断窗口中生成错误消息。不过，即使出现错误消息也能够正确执行该函数。

下列 SetTag 函数将受到影响：

- SetTagWithOperatorEvent
- SetTagIndirect
- SetTagByTagIndirect
- SetTagIndirectByTagIndirect
- SetTagByProperty
- SetTagIndirectByProperty

语法

BOOL<FunctionName><(Parameter)>;

示例：BOOL SetTagBit(Tag_Name, value);

参数

Tag_Name

变量名称

value

指定数据类型的变量值。

pValue

包含原始数据变量值的字节字段指针。

size

字节字段的长度（以字节为单位）

返回值

TRUE

系统函数已成功执行，未发生任何错误。

但是，未进行任何检查以验证写入的变量没有任何错误。

FALSE

发生错误。

示例

以下程序代码使用 SetTagBit 函数将 gs_tag_bit 变量的值设置为 TRUE，并将返回值保存到 ok 变量中。

```
{
BOOL ok;
BOOL bvalue;

//Set the tag to true
ok = SetTagBit("gs_tag_bit", TRUE);
//error handling
if(ok)
{
  // succeeded
  printf ( "Function has run through.\r\n" );
bvalue = GetTagBit("gs_tag_bit");
  printf ("Value of gs_tag_bit: %d\r\n", bvalue);
}
else
{
  // failed
  printf ( "Error - function failed." );
}
...
}
```

保存的返回值可在后续代码中进行处理。

参见

HMI 变量的质量代码 (页 1814)

VariableStateType 属性 (页 1821)

2.1.39.2 SetTagDateTime

功能

设置数据类型为 “日期/时间”(Date/Time) 的变量值。

语法

BOOL SetTagDateTime(Tag_Name, value);

参数

Tag_Name

变量名称

value

“日期/时间”(Date/Time) 数据类型的变量值。

返回值

TRUE

函数本身已执行，未发生任何错误。

但是，未进行任何检查以验证写入的变量没有任何错误。

FALSE

发生错误。

2.1.39.3 SetTagMultiStateWait 函数

说明

设置多个变量的值。该系统函数只有在 AS 报告已接收到数值后才会结束。

该系统函数只能用于 C 脚本。

必须为系统函数提供一个 DWORD 型数组，以便在完成系统函数调用后，各个变量的状态会存储在该数组的成员中。数组必须足够大，以便为状态提供足够的存储空间。

有关详细信息，请参见 Internet 上的 FAQ，条目 ID 为 26712371 (https://support.industry.siemens.com/cs/cn/zh/view/26712371)。

语法

BOOL SetTagMultiStateWait(pdwState, pFormat,...);

参数

pdwState

保存变量状态的域。

pFormat

所有请求变量的格式描述以及每个变量的名称和值。

返回值

TRUE

系统函数已成功执行，未发生任何错误。

但是，未进行任何检查以验证写入的变量没有任何错误。

FALSE

发生错误。

示例

以下程序代码使用 SetTagMultiStateWait 函数设置多个变量的值。

1. 创建一个所需大小（变量数）的 DWord 数组
2. 通过 SetTagMultiStateWait 函数创建变量，以存储要传送到 WinCC 变量的值
3. 将最近声明的变量的值写入 WinCC 变量：
   - gs_tag_bit 包括变量值 "lValue1"
   - gs_tag_SByte 包括地址 "&lValue2" 中的变量值
   - 等等

```
{
#define DATA_SIZE 5
DWORD dwData[DATA_SIZE];

//define all tags
BOOL lValue1;
long lValue2;
char szValue3[_MAX_PATH];
float lValue4;
char lValue5;

// Fill the tags with the values
// you want to set into the WinCC tags
...

//Set the WinCC tags
SetTagMultiStateWait(dwData,"%d%d%s%f%d","gs_tag_bit",lValue1,
 "gs_tag_SByte",lValue2,
 "gs_tag_char",szValue3,
 "gs_tag_float",lValue4,
 "gs_tag_word",lValue5);
...
}
```

保存的返回值可在后续代码中进行处理。

参见

常数 (页 1822)

条目 26712371 (https://support.industry.siemens.com/cs/cn/zh/view/26712371)

2.1.39.4 SetTagMultiWait 函数

说明

以指定的格式设置多个变量的值。该系统函数只有在 AS 报告已接收到数值后才会结束。

只能在 C 脚本中使用。

有关详细信息，请参见 Internet 上的 FAQ，条目 ID 为 26712371 (https://support.industry.siemens.com/cs/cn/zh/view/26712371)。

语法

```
BOOL SetTagMultiWait(pFormat,...);
```

参数

pFormat

所有请求变量的格式描述以及每个变量的名称和值。

返回值

TRUE

系统函数已成功执行，未发生任何错误。

但是，未进行任何检查以验证写入的变量没有任何错误。

FALSE

发生错误。

示例

以下程序代码使用 SetTagMultiWait 函数更改多个变量的值。返回值将保存到 ok 变量。

```
{
BOOL ok;
//memory for values allocated via SysMalloc
DWORD dwVar1Value;
char* szVar2Value;
double dbVar3Value;

//settings
ok=SetTagMultiWait("%d%s%f", "Ernie_word", 16,
  "Ernie_char", "Hallo Welt",
  "Ernie_double", 55.4711);

//error handling
if(ok)
{
  // succeeded
  printf ( "Function has run through.\r\n" );

// Get values and print
GetTagMultiWait("%d%s%f", "Ernie_word", &dwVar1Value,
  "Ernie_char", &szVar2Value,
  "Ernie_double", &dbVar3Value);

printf("Word %d, String %s, Double %f\r\n",
  dwVar1Value, szVar2Value, dbVar3Value);
}
else
{
  // failed
  printf ( "Error - function failed." );
}
...
}
```

示例

GetTagMultiWait 函数在以下程序代码中用于读取多个不同类型的变量：

1. 声明三个变量，作为三个不同变量类型的存储区
2. 声明布尔型变量 ok，用于缓冲返回值（TRUE/FALSE）
3. 读取这三个变量并将其值保存到相应的地址。
   函数的返回值保存到 ok 变量。
4. 输出带有变量前缀的三个变量

```
DWORD dwVar1Value;
char* szVar2Value;
//Memory for the tag value is allocated by the function SysMalloc
double dbVar3Value;

BOOL ok;

ok=GetTagMultiWait("%d%s%f", "Ernie_word", &dwVar1Value,
  "Ernie_char", &szVar2Value,
  "Ernie_double", &dbVar3Value);

printf("Word %d, String %s, Double %f\r\n",
  dwVar1Value, szVar2Value, dbVar3Value);
```

参见

条目 26712371 (https://support.industry.siemens.com/cs/cn/zh/view/26712371)

2.1.39.5 SetTagState 函数

描述

用于设置所指定数据类型的变量值。也用于返回变量状态。

只能在 C 脚本中使用。

下表列出了可读取变量值的各类 SetTagStateXXX 函数：

函数名称	参数	PLC 数据类型	HMI 数据类型
SetTagBitState	Tag Tag_Name, short int value, PDWORD lp_dwstate	二进制变量	布尔型
SetTagByteState	Tag Tag_Name, BYTE value, PDWORD lp_dwstate	无符号 8 位	无符号字节型
SetTagCharState	Tag Tag_Name, LPSTR value, PDWORD lp_dwstate	文本变量 8 位或文本变量 16 位	字符串型

函数名称	参数	PLC 数据类型	HMI 数据类型
SetTagDoubleState	Tag Tag_Name, double value, PDWORD lp_dwstate	浮点数 64 位	双精度型
SetTagDWordState	Tag Tag_Name, DWORD value, PDWORD lp_dwstate	无符号 32 位	无符号整型
SetTagFloatState	Tag Tag_Name, float value, PDWORD lp_dwstate	浮点数 32 位	浮点型
SetTagRawState	Tag Tag_Name, BYTE* pValue, DWORD size, PDWORD lp_dwstate	原始数据类型	原始类型
SetTagSByteState	Tag Tag_Name, signed char value, PDWORD lp_dwstate	有符号 8 位	字节型
SetTagSDWordState	Tag Tag_Name, long int value, PDWORD lp_dwstate	有符号 32 位	整型
SetTagSWordState	Tag Tag_Name, short int value, PDWORD lp_dwstate	有符号 16 位	短整型
SetTagWordState	Tag Tag_Name, WORD value, PDWORD lp_dwstate	无符号 16 位	无符号短整型

语法

BOOL <FunctionName><(Parameter)>;

示例：BOOL SetTagBitState(Tag_Name, value, lp_dwstate);

参数

Tag_Name

变量名称

value

指定类型的变量值。

lp_dwstate

用于在系统函数周期完成时保存变量状态的 DWORD 指针。

pFormat

所有请求变量的格式描述以及每个变量的名称和值。

pValue

包含原始数据变量值的字节字段指针。

size

字节字段的长度（以字节为单位）

返回值

TRUE

系统函数已成功执行，未发生任何错误。

但是，未进行任何检查以验证写入的变量没有任何错误。

FALSE

发生错误。

示例

以下程序代码使用 SetTagBitState 函数将 gs_tag_bit 变量的值设置为 TRUE，并将返回值保存到 ok 变量中。“"&dwstate”是变量的地址，用于保存变量状态。

保存的返回值可在后续代码中进行处理。

```
{
DWORD dwstate;
BOOL ok;

//Load dwState with default values
dwstate = 0xFFFFFFFF;
//Set the value of the tag to TRUE
//dwstate is the tag state
ok = SetTagBitState("gs_tag_bit",TRUE,&dwstate);

//error handling
if(ok)
{
  // succeeded
  printf ( "Function has run through.\r\n" );
  printf ("Status of gs_tag_bit: %d\r\n", dwstate);
}
else
{
  // failed
  printf ( "Error - function failed." );
}
...
}
```

参见

常数 (页 1822)

2.1.39.6 SetTagStateWait 函数

描述

用于设置所指定数据类型的变量值。该系统函数只有在 AS 报告已接收到数值后才会结束。也用于返回变量状态。

只能在 C 脚本中使用。

下表列出了可设置变量值的各类 SetTagStateWait 函数：

函数名称	参数	PLC 数据类型	HMI 数据类型
SetTagBitStateWait	Tag Tag_Name, short int value, PDWORD lp_dwstate	二进制变量	布尔型
SetTagByteStateWait	Tag Tag_Name, BYTE value, PDWORD lp_dwstate	无符号 8 位	无符号字节型
SetTagCharStateWait	Tag Tag_Name, LPSTR value, PDWORD lp_dwstate	文本变量 8 位或文本变量 16 位	字符串型
SetTagDoubleStateWait	Tag Tag_Name, double value, PDWORD lp_dwstate	浮点数 64 位	双精度型
SetTagDWordStateWait	Tag Tag_Name, DWORD value, PDWORD lp_dwstate	无符号 32 位	无符号整型
SetTagFloatStateWait	Tag Tag_Name, float value, PDWORD lp_dwstate	浮点数 32 位	浮点型
SetTagRawStateWait	Tag Tag_Name, BYTE* pValue, DWORD size, PDWORD lp_dwstate	原始数据类型	原始类型
SetTagSByteStateWait	Tag Tag_Name, char value, PDWORD lp_dwstate	有符号 8 位	字节型
SetTagSDWordStateWait	Tag Tag_Name, long int value, PDWORD lp_dwstate	有符号 32 位	整型
SetTagSWordStateWait	Tag Tag_Name, short int value, PDWORD lp_dwstate	有符号 16 位	短整型
SetTagWordStateWait	Tag Tag_Name, WORD value, PDWORD lp_dwstate	无符号 16 位	无符号短整型

语法

BOOL<FunctionName><(Parameter)>;

示例：BOOL SetTagBitStateWait(Tag_Name, value, lp_dwstate);

参数

Tag_Name

变量名称

value

指定类型的变量值。

lp_dwstate

用于在系统函数周期完成时保存变量状态的 DWORD 指针。

pValue

包含原始数据变量值的字节字段指针。

size

字节字段的长度（以字节为单位）

返回值

TRUE

系统函数已成功执行，未发生任何错误。

但是，未进行任何检查以验证写入的变量没有任何错误。

FALSE

发生错误。

示例

以下程序代码使用 SetTagBitStateWait 函数将 gs_tag_bit 变量的值设置为 TRUE，并将返回值保存到 ok 变量中。“"&dwstate”是变量的地址，用于保存变量状态。

保存的返回值可在后续代码中进行处理。

```
{
DWORD dwstate;
BOOL ok;

//Load dwState with default values
dwstate = 0xFFFFFFFF;
//Set the value of the tag to TRUE
//dwstate is the tag state
ok = SetTagBitStateWait("gs_tag_bit",TRUE,&dwstate);

//error handling
if(ok)
{
  // succeeded
  printf ( "Function has run through.\r\n" );
  printf ("Status of gs_tag_bit: %d\r\n", dwstate);
}
else
{
  // failed
  printf ( "Error - function failed." );
}
...
}
```

参见

常数 (页 1822)

2.1.39.7 SetTagValue 函数

说明

实现以变体格式传输值，并设置指向数据类型为“Variant”的值的指针。

只能在 C 脚本中使用。

语法

BOOL SetTagValue(lpdmVarKey, lpdmValue, dwState, lpdmError);

参数

lpdmVarKey

指向数据类型"DM_VARKEY"结构的指针

lpdmValue

指向数据类型"Variant"结构的指针。有关数据类型 VARIANT 的说明，请参见相关的技术参考材料。

lpdmError

包含错误描述结构的指针。

返回值

TRUE

系统函数已成功执行，未发生任何错误。

但是，未进行任何检查以验证写入的变量没有任何错误。

FALSE

发生错误。

示例

SetTagValue 函数在以下程序代码中用于传送 varKey 中的值。

根据 keyFound (TRUE/FALSE) 中的返回值执行特定的代码。

```
{
// tags for setting the value
DM_VARKEY varKey;
LPVARIANT value;
LPCMN_ERROR error1:

// tags for getting the value
DM_VAR_UPDATE_STRUCT result;
CMN_ERROR error:

BOOL keyFound;

ok = SetTagValue(&varKey, &value, &error1);

if (keyFound)
{
  // succeeded, get the new value
GetTagValue(&varKey, &result, &error);
  // print tag value
   printf ("Value of varKey: %d\r\n", &varKey);
  ...
}
else
{
  // failed
  printf ( "Error - function failed." );
  ...
}
}
```

2.1.39.8 SetTagValueWait 函数

说明

实现以变体格式传输值，并设置指向数据类型为“Variant”的值的指针。该系统函数只有在 AS 报告已接收到数值后才会结束。

只能在 C 脚本中使用。

语法

BOOL SetTagValueWait(lpdmVarKey, lpdmValue, dwState, lpdmError);

参数

lpdmVarKey

指向数据类型"DM_VARKEY"结构的指针

lpdmValue

指向数据类型"Variant"结构的指针。有关数据类型 VARIANT 的说明，请参见相关的技术参考材料。

dwState

在系统函数周期后所返回的变量状态。

lpdmError

包含错误描述结构的指针。

返回值

TRUE

系统函数已成功执行，未发生任何错误。

无需检查在写入变量时是否未发生任何错误。

FALSE

发生错误。

示例

SetTagValueWait 函数在以下程序代码中用于传送 varKey 中的值。

根据 keyFound (TRUE/FALSE) 中的返回值执行特定的代码。

```
{
// tags for setting the value
DM_VARKEY varKey;
LPVARIANT value;
LPCMN_ERROR error1:

// tags for getting the value
DM_VAR_UPDATE_STRUCT result;
CMN_ERROR error:

BOOL keyFound;

ok = SetTagValueWait(&varKey, &value, &error1);

if (keyFound)
{
  // succeeded, get the new value
GetTagValueWait(&varKey, &result, &error);
  // print tag value
   printf ("Value of varKey: %d\r\n", &varKey);
  ...
}
else
{
  // failed
  printf ( "Error - function failed." );
  ...
}
}
```

2.1.39.9 SetTagWait 函数

描述

用于设置所指定数据类型的变量值。该系统函数只有在 AS 报告已接收到数值后才会结束。

只能在 C 脚本中使用。

下表列出了可设置变量值的各类 SetTagWait 函数：

函数名称	参数	PLC 数据类型	HMI 数据类型
SetTagBitWait	Tag Tag_Name, short int value	二进制变量	布尔型
SetTagByteWait	Tag Tag_Name, BYTE value	无符号 8 位	无符号字节型
SetTagCharWait	Tag Tag_Name, char* value	文本变量 8 位或文本变量 16 位	字符串型
SetTagDoubleWait	Tag Tag_Name, double value	浮点数 64 位	双精度型
SetTagDWordWait	Tag Tag_Name, DWORD value	无符号 32 位	无符号整型
SetTagFloatWait	Tag Tag_Name, float value	浮点数 32 位	浮点型
SetTagRawWait	Tag Tag_Name, BYTE* pValue, DWORD size	原始数据类型	原始类型
SetTagSByteWait	Tag Tag_Name, char value	有符号 8 位	字节型
SetTagSDWordWait	Tag Tag_Name, long int value	有符号 32 位	整型
SetTagSWordWait	Tag Tag_Name, short int value	有符号 16 位	短整型
SetTagWordWait	Tag Tag_Name, WORD value	无符号 16 位	无符号短整型

语法

BOOL <FunctionName><(Parameter)>;

示例：BOOL SetTagBitWait(Tag_Name, value);

参数

Tag_Name

变量名称

value

指定类型的变量值。

pValue

包含原始数据变量值的字节字段指针。

size

字节字段的长度（以字节为单位）

返回值

TRUE

系统函数已成功执行，未发生任何错误。

但是，未进行任何检查以验证写入的变量没有任何错误。

FALSE

发生错误。

示例

以下程序代码使用 SetTagBitWait 函数将 gs_tag_bit 变量的值设置为 TRUE，并将返回值保存到 ok 变量中。

```
{
BOOL ok;
BOOL bvalue;

//Set the tag to true
ok = SetTagBitWait("gs_tag_bit", TRUE);
//error handling
if(ok)
{
  // succeeded
  printf ( "Function has run through.\r\n" );
bvalue = GetTagBitWait("gs_tag_bit");
  printf ("Value of gs_tag_bit: %d\r\n", bvalue);
}
else
{
  // failed
  printf ( "Error - function failed." );
}
...
}
```

保存的返回值可在后续代码中进行处理。

2.1.39.10 SetTag

描述

将新值赋给给定的变量。

说明

该系统函数可用于根据变量类型分配字符串和数字。

在函数列表中使用

设置变量（变量，值）

在用户自定义函数中使用

SetTag Tag, Value

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

变量

为其分配给定值的变量。

值

为给定变量所赋的值。

说明

“SetTag”系统函数只能在建立连接后执行。

示例

下面的程序代码使用 SetTag 函数将 gs_tag_bit 变量的值设置为 TRUE，并将返回值保存到 ok 变量中。

```
{
BOOL ok;
BOOL bvalue;

//Set the tag to true
ok = SetTag("gs_tag_bit", TRUE);
//error handling
if(ok)
{
  // succeeded
  printf ( "Function has run through.\r\n" );
bvalue = GetTagBit("gs_tag_bit");
  printf ("Value of gs_tag_bit: %d\r\n", bvalue);
}
else
{
  // failed
  printf ( "Error - function failed." );
}
...
}
```

保存的返回值可在后续代码中进行处理。

2.1.39.11 SetTagByProperty

描述

通过对象属性的值指定变量值。更改还会记录在报警系统中。

在函数列表中使用

按属性设置间接变量（变量名称，画面名称，画面对象，属性名称，包含或不包含操作员事件）

在用户自定义函数中使用

SetTagByProperty Tag_name, Screen_name, Screen_object, Property_name, With_or_without_operator_event

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见 “AUTOHOTSPOT”。

参数

变量名称

值由对象属性指定的变量名称。

画面名称

包含对象的画面的名称。

画面对象

其属性提供变量值的对象名称。

属性名称

提供变量值的属性名称。

有或没有操作员事件

0 (hmiWithoutOperatorEvent) = 没有操作员事件

1 (hmiWithOperatorEvent) = 有操作员事件

示例

在组合框中单击时，下面的程序代码将返回所选文本的值。

```
{
char* rt_value;

SetTagByProperty (rt_value, screenName, objectName, "SelectedText",
hmiWithoutOperatorEvent);

...
}
```

2.1.39.12 SetTagByTagIndirect

描述

将间接寻址的变量值写入变量。作为参数传送的变量包含其值被读取的另一变量的名称。通过操作员输入报警记录报警系统的变化。

在函数列表中使用

按间接变量设置变量（此变量的名称，变量名称，带或不带操作员输入报警）

在用户自定义函数中使用

SetTagByTagIndirect Tag_name, Source_tag_name, With_or_without_operator_event

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“AUTOHOTSPOT”。

参数

此变量的名称

要设置其值的变量名称。

变量名称

字符串变量的名称，该变量包含提供变量值的变量名。

有或没有操作员事件

0 (hmiWithoutOperatorEvent) = 无操作员输入报警

1 (hmiWithOperatorEvent) = 有操作员输入报警

示例

以下程序代码会将变量 Tag4 的值写入变量 Tag1。

```
{
SetTag ("IndirectRead", "Tag4");
SetTagByTagIndirect ("Tag1", "IndirectRead", hmiWithoutOperatorEvent);

...
}
```

2.1.39.13 SetTagIndirect

描述

向间接寻址变量写入值。作为输出参数传送的变量包含其值通过函数进行更改的另一变量的名称。通过操作员输入报警记录报警系统的变化。

在函数列表中使用

设置间接变量（变量名称，值，包含或不包含操作员事件）

在用户自定义函数中使用

SetTagIndirect Tag_name, Value, With_or_without_operator_event

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见 “AUTOHOTSPOT”。

参数

变量名称

字符串变量的名称，该变量包含要更改其值的变量名。

值

要写入的值。

包含或不包含操作员事件

0 (hmiWithoutOperatorEvent) = 不包含操作员事件

1 (hmiWithOperatorEvent) = 包含操作员事件

示例

以下程序代码会将值 "value" 写入变量 Tag3。

```
{
int value;

SetTag ("IndirectWrite", "Tag3");
SetTagIndirect ("IndirectWrite", "value", hmiWithoutOperatorEvent);
...
}
```

2.1.39.14 SetTagIndirectByProperty

描述

将对象属性的值写入间接寻址的变量。作为输出参数传送的变量包含其值通过函数进行更改的另一变量的名称。通过操作员输入报警记录报警系统的变化。

在函数列表中使用

按属性设置间接变量（变量名称，画面名称，画面对象，属性名称，有或没有操作员事件）

在用户自定义函数中使用

SetTagIndirectByProperty Tag_name, Screen_name, Screen_object, Property_name, With_or_without_operator_event

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“AUTOHOTSPOT”。

参数

变量名称

字符串变量的名称，该变量包含要更改其值的变量名。

画面名称

包含对象的画面的名称。

画面对象

其属性可提供数值的对象名称。

属性名称

可提供数值的属性名称。

有或没有操作员事件

0 (hmiWithoutOperatorEvent) = 无输入报警

1 (hmiWithOperatorEvent) = 有操作员输入报警

示例

以下程序代码将对象属性“背景色”(Background color) 的值写入变量 Tag2。

```
{
SetTag ("IndirectWrite", "Tag2");
SetTagIndirectByProperty ("IndirectWrite", screenName, objectName, "BackColor",
hmiWithoutOperatorEvent);
...
}
```

2.1.39.15 SetTagIndirectByTagIndirect

说明

将间接寻址的变量值写入间接寻址的变量。作为输出参数传送的变量包含其值通过函数进行更改的另一变量的名称。作为参数传送的变量包含其值被读取的另一变量的名称。通过操作员输入报警记录报警系统的变化。

在函数列表中使用

按间接变量设置间接变量（变量名，变量名，带或不带操作员输入报警）

在用户自定义函数中使用

SetTagIndirectByTagIndirect Tag_name, Source_tag_name,
With_or_without_operator_event

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“AUTOHOTSPOT”。

参数

变量名称

字符串变量的名称，该变量包含要更改其值的变量名。

变量名称

返回变量值的间接变量的名称。

有或没有操作员事件

0 (hmiWithoutOperatorEvent) = 没有操作员事件

1 (hmiWithOperatorEvent) = 有操作员事件

示例

以下程序代码会将变量 "Tag4" 的值写入变量 "Tag2"。

```
{
SetTag ("IndirectWrite", "Tag2");
SetTag ("IndirectRead", "Tag4");
SetTagIndirectByTagIndirect ("IndirectWrite", "IndirectRead");
...
}
```

2.1.39.16 SetTagIndirectWithOperatorEvent

描述

为变量指定间接名称。 更改还会记录在报警系统中。

在函数列表中使用

使用操作员输入报警设置间接变量（变量名称（输出），LpValue）

在用户自定义函数中使用

SetTagIndirectWithOperatorEvent

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“AUTOHOTSPOT”。

参数

变量名称 (输出)

要将变量名写入到的变量的名称。

LpValue

被写入到变量中的变量的名称。

2.1.39.17 SetTagWithOperatorEvent

说明

为变量指定值。更改还会记录在报警系统中。

在函数列表中使用

SetTagWithOperatorEvent（变量名称，值）

在用户自定义函数中使用

SetTagWithOperatorEvent Tag_name, Value

如果组态的设备支持用户自定义函数，则可以使用。有关详细信息，请参见“AUTOHOTSPOT”。

参数

变量名称

要设置其值的变量名称。

值

已写入变量的值。

示例

单击相应按钮时，下面的程序代码将 "value" 变量的值传送到 "result" 变量。

```
'Programming language: VBS
SetTagWithOperatorEvent result, value
...
```

```
//Programming language: C
{
SetTagWithOperatorEvent ("result", "value");
...
}
```

2.1.40 StartProgram

说明

启动具有特定参数的特定程序。

该系统函数只能用于 C 脚本

语法

void StartProgram(Program_name, Program_parameters, Display_mode, Wait_for_program_end);

参数

Program_name

要启动的程序的路径和名称。

Program_parameters

启动时使用的参数。有关可以使用的参数的信息，请参见要启动的程序的说明。

Display_mode

定义程序启动时使用的显示模式：

0 (hmiShowNormal) = 以正常窗口显示

1 (hmiShowMinimized) = 以最小化窗口显示

2 (hmiShowMaximized) = 以最大化窗口显示

3 (hmiShowMinimizedAndInactive) = 以未激活的最小化窗口显示

Wait_for_program_end

该参数不由 WinCC Runtime Professional 评估。

示例

以下程序代码以最小化窗口启动 calc.exe 程序。

```
{
BOOL Wait_for_program_end;
char* number;

//start the program calc.exe
StartProgram("C:\\Winnt\\system32\\calc.exe",number,hmiShowMinimized,
Wait_for_program_end);
...
}
```

2.1.41 StopRuntime

描述

退出运行系统软件，从而退出运行在 HMI 设备上的项目。

在函数列表中使用

停止运行系统（模式）

在用户自定义函数中使用

StopRuntime Mode

如果组态的设备支持用户自定义函数，则可以使用。更多信息，请参考“设备相关性”。

参数

模式

确定在退出运行系统后操作系统是否关闭。

0 (hmiStopRuntime) = 运行系统：操作系统不关闭

1 (hmiStopRuntimeAndOperatingSystem) = 运行系统和操作系统：操作系统关闭(对于 WinCE 不适用)

示例

下面的程序代码将关闭运行系统和操作系统。

```
{

//Stop runtime and shutdown
StopRuntime (hmiStopRuntimeAndOperationSystem);

}
```

保存的返回值可在后续代码中进行处理。

2.1.42 StoreScreen

描述

保存当前画面。 此画面可使用 ActivateStoredScreen 系统函数打开。

只能在 C 脚本中使用。

语法

BOOL StoreScreen();

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

以下程序代码将 StoreScreen 函数的返回值写入 screen_stored 变量并调用存储的画面（前提是保存过程中未发生错误）。

```
{
BOOL screen_stored;
screen_stored = StoreScreen();
//user defined code
...
//error handling
if(screen_stored)
{
  // succeeded
  ActivateStoredScreen();
  printf ( "Stored screen is now activated.\r\n" );
}
else
{
  // failed
  printf ( "Error - no screen stored." );
}
...
}
```

参见

ActivateStoredScreen (页 1613)

2.1.43 SystemTimeToDate

说明

转换 SYSTEMTIME 格式的日期时间技术数据到 DATE 数据格式。

只能在 C 脚本中使用。

在函数列表中使用

SystemTimeToDate（值、时间指针）

语法

SystemTimeToDate(Value, PointerToTime);

参数

值

SYSTEMTIME 数据格式的值

时间的指针

指向 DATE 格式的结果的指针

返回值

TRUE

系统函数已成功执行，未发生任何错误。

FALSE

发生错误。

示例

```
BOOL BRet;
SYSTEMTIME st_1, st_2;
DATE       d_1, d_2;   // wtypes.h.  DATE type. Visual Studio documnetation.

GetSystemTime( &st_1 );

printf( "st_1.wYear = %d \r\n", st_1.wYear );
printf( "st_1.wMonth = %d \r\n", st_1.wMonth );
printf( "st_1.wDayOfWeek = %d \r\n", st_1.wDayOfWeek );
printf( "st_1.wDay = %d \r\n", st_1.wDay );printf( "st_1.wHour = %d \r\n", st_1.wHour );
printf( "st_1.wMinute = %d \r\n", st_1.wMinute );
printf( "st_1.wSecond = %d \r\n", st_1.wSecond );
printf( "st_1.wMilliseconds = %d \r\n", st_1.wMilliseconds );

BRet = SystemTimeToDate( st_1, &d_1 );
printf( "DATE d = %ld \r\n \r\n", d_1 );
printf( "DATE d = %lf \r\n \r\n", d_1 );
printf( "DATE d = %f \r\n \r\n", d_1 );

BRet = DateToSystemTime( d_1, &st_2 );
printf( "st_2.wYear = %d \r\n", st_2.wYear );
printf( "st_2.wMonth = %d \r\n", st_2.wMonth );
printf( "st_2.wDayOfWeek = %d \r\n", st_2.wDayOfWeek );
printf( "st_2.wDay = %d \r\n", st_2.wDay );
printf( "st_2.wHour = %d \r\n", st_2.wHour );
printf( "st_2.wMinute = %d \r\n", st_2.wMinute );
printf( "st_2.wSecond = %d \r\n", st_2.wSecond );
printf( "st_2.wMilliseconds = %d \r\n \r\n \r\n", st_2.wMilliseconds );
```

2.1.44 TriggerOperatorEvent

描述

TriggerOperatorEvent 系统函数用于触发操作员输入报警。

语法

int TriggerOperatorEvent(dwFlags, dwMsgNum, lpszObjectName, dwMyTextID, doValueOld, doValueNew, pszComment);

参数

dwFlags

FLAG_COMMENT_PARAMETER (0x001)：指定通过此参数输入注释。

FLAG_COMMENT_DIALOG (0x003)：指定通过对话框输入注释。

FLAG_TEXTID_PARAMETER (0x100)：指定通过文本 ID 指定注释。为此，在“文本和图形列表”(Text and graphics list) 编辑器的“C 文本列表”(C text list) 选项卡中组态一个文本列表条目。

dwMsgNum

触发的操作员输入报警的编号。

lpszObjectName

指向具有旧值和新值的变量的名称的指针。

dwMyTextID

要用作注释的文本的 ID。

如果您为参数“dwFlags”使用“FLAG_COMMENT_PARAMETER”(0x001) 或“FLAG_COMMENT_DIALOG”(0x003)，则为参数“dwMyTextID”输入值 0。

如果您为参数“dwFlags”使用“FLAG_TEXTID_PARAMETER”(0x100)则为参数“dwMyTextID”输入文本列表条目的 ID。可以在“文本和图形列表”(Text and graphics list) 编辑器的“C 文本列表”(C text list) 选项卡中组态文本列表条目。

doValueOld

旧值。

doValueNew

新值。

pszComment

指向要用作注释的文本的指针。

返回值

0

系统函数已成功执行，未发生任何错误。

-101

无法开始编辑操作员输入报警。

-201

调用"MSRTGetComment()"时发生错误。

-301

调用"MSRTCreateMsgInstanceWithComment()"时发生错误。

2.1.45 UA（配方）

2.1.45.1 uaArchiveClose

说明

终止与当前配方的连接。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveClose (
UAHARCHIVE hArchive )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

成功关闭配方。

FALSE

错误

2.1.45.2 uaArchiveDelete

说明

从配方中删除数据。 然而，保留已组态的配方。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveDelete (
UAHARCHIVE hArchive,
LPCSTR pszWhere )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LPCSTR pszWhere

该字符串包含 SQL 选择表达式。 它定义要删除哪些数据记录。 该表达式等同于 SQL 指令 “DELETE FROM <archive> WHERE pszWhere”。

警告！ 若字符串为空，则将删除整个配方。

返回值

TRUE

成功删除配方。

FALSE

错误

2.1.45.3 uaArchiveExport

描述：

将当前配方以 CSV 格式导出为日志。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveExport (
 UAHARCHIVE hArchive,
 LPCSTR pszDestination,
 LONG lType,
 LONG lOptions )
```

参数

UAHARCHIVE hArchive

配方的句柄。此句柄通过 UaQueryArchive 或 UaQueryArchiveByName 生成。

LPCSTR pszDestination

目标归档的文件名。在客户端上调用该函数时，指定的路径表示服务器。

LONG lType

目标归档的数据格式。两种格式可用：

- UA_FILETYPE_DEFAULT = 0：默认文件格式 = CSV
- UA_FILETYPE_CSV = 1：CSV 文件格式

LONG lOptions

保留以供将来扩展。必须为 0。

返回值

TRUE

成功导出配方。

FALSE

错误

2.1.45.4 uaArchiveGetCount

说明

读取数据记录的数量。

只能在 C 脚本中使用。

语法

```
LONG uaArchiveGetCount(
 UAHARCHIVE hArchive,
 LONG * plCount )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LONG plCount

指向用于储存数据记录数量的变量的指针。

返回值

数据记录的数量。

0 = 记录为空或发生了错误。 必须通过 uaGetLastError() 进行查询。

2.1.45.5 uaArchiveGetFieldLength

说明

读取当前数据记录中域的长度。

只能在 C 脚本中使用。

语法

```
LONG uaArchiveGetFieldLength(
 UAHARCHIVE hArchive,
```

```
 LONG lField )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LONG lField

字段编号，其中 lField = 1 表示指向第一个字段的地址。

返回值

当前域的长度。

2.1.45.6 uaArchiveGetFieldName

说明

读取当前数据记录中域的名称。

只能在 C 脚本中使用。

语法

```
VOID uaArchiveGetFieldName (
 UAHARCHIVE hArchive,
 LONG lField,
 LPCSTR pszName,
 LONG cMaxLen )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LONG lField

字段编号，其中 lField = 1 表示指向第一个字段的地址。

LPCSTR pszName

域名称

LONG cMaxLen

最大长度

2.1.45.7 uaArchiveGetFields

说明

读取已组态数据域（包括"ID"、“上个用户”和“上次访问”域）的数量。 在运行系统调用中，已组态域的索引标记为 1 至 N。域 ID 索引为 0。“上个用户”和“上次访问”域附加在已组态域的末尾。

只能在 C 脚本中使用。

语法

```
LONG uaArchiveGetFields (
 UAHARCHIVE hArchive )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

已组态域的数目。

2.1.45.8 uaArchiveGetFieldType

说明

读取当前数据记录中域的类型。

只能在 C 脚本中使用。

语法

```
LONG uaArchiveGetFieldType (
UAHARCHIVE hArchive,
LONG lField )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LONG lField

字段编号，其中 lField = 1 表示指向第一个字段的地址。

返回值

当前域的类型。

域类型的符号定义是：

UA_FIELDTYPE_INTEGER

UA_FIELDTYPE_DOUBLE

UA_FIELDTYPE_STRING

UA_FIELDTYPE_DATETIME

2.1.45.9 uaArchiveGetFieldValueDate

说明

读取当前数据记录中域的日期和时间。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveGetFieldValueDate (
UAHARCHIVE hArchive,
LONG lField,
```

```
LPSYSTEMTIME pstDateTime )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LONG lField

字段编号，其中 lField = 1 表示指向第一个字段的地址。

LPSYSTEMTIME pstDateTime

类型 SYSTEMTIME 变量的指针

返回值

TRUE

成功读取日期和时间

FALSE

错误

2.1.45.10 uaArchiveGetFieldValueDouble

说明

读取当前数据记录中域的双精度值。

该系统函数只能用于 C 脚本。

```
BOOL uaArchiveGetFieldValueDouble (
UAHARCHIVE hArchive,
LONG lField,
double* pdValue )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LONG lField

字段编号，其中 lField = 1 表示指向第一个字段的地址。

double* pdValue

指向当前域内容的变量的指针。

返回值

TRUE

成功读取字段值

FALSE

错误

2.1.45.11 uaArchiveGetFieldValueFloat

说明

读取当前数据记录中域的浮点值。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveGetFieldValueFloat (
UAHARCHIVE hArchive,
LONG lField,
FLOAT* pfValue )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LONG lField

字段编号，其中 lField = 1 表示指向第一个字段的地址。

FLOAT* pfValue

指向当前域内容的浮点变量的指针。

返回值

TRUE

成功读取字段值

FALSE

错误

2.1.45.12 uaArchiveGetFieldValueLong

说明

读取当前数据记录中域的长整型值。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveGetFieldValueLong (
UAHARCHIVE hArchive,
LONG lField,
LONG* pdValue )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LONG lField

字段编号，其中 lField = 1 表示指向第一个字段的地址。

LONG* pdValue

指向当前域内容的长整型变量的指针。

返回值

TRUE

成功读取字段值

FALSE

错误

2.1.45.13 uaArchiveGetFieldValueString

说明

读取当前数据记录中域的字符串。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveGetFieldValueString (
UAHARCHIVE hArchive,
LONG lField,
LPSTR pszString,
LONG cMaxLen )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LONG lField

字段编号，其中 lField = 1 表示指向第一个字段的地址。

LPCSTR pszString

域值为字符串。

LONG cMaxLen

字符串的最大长度。

返回值

TRUE

成功读取字段值

FALSE

错误

2.1.45.14 uaArchiveGetFilter

说明

读取当前数据记录的过滤器。 更多信息，请参见附录中"SQL 语句"。

只能在 C 脚本中使用。

语法

```
VOID uaArchiveGetFilter (
 UAHARCHIVE hArchive,
 LPSTR pszFilter,
 LONG cMaxLen )
```

该系统函数只能用于 C 脚本。

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LPSTR pszFilter

读取过滤器。

LONG cMaxLen

最大长度

2.1.45.15 uaArchiveGetID

说明

uaArchiveGetID 读取配方的 ID。

配方 ID 用于内部用途，并且可能与配方中给出的编号不同。

只能在 C 脚本中使用。

语法

```
LONG uaArchiveGetID (
 UAHARCHIVE hArchive )
```

参数

UAHARCHIVE hArchive
配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

配方 ID。

2.1.45.16 uaArchiveGetName

说明

读取配方的名称。

只能在 C 脚本中使用。

Syntax

```
VOID uaArchiveGetName (
UAHARCHIVE hArchive,
LPSTR pszName,
LONG cMaxLen )
```

参数

UAHARCHIVE hArchive
配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LPSTR pszName
配方名称缓冲区指针。

LONG cMaxLen
最大长度

示例

```
char Filling [40];
uaArchiveGetName( hArchive, bottling, 39 );
```

2.1.45.17 uaArchiveGetSor

说明

uaArchiveGetSort 将读取配方的排序结果。

只能在 C 脚本中使用。

语法

```
VOID uaArchiveGetSort (
UAHARCHIVE hArchive,
LPSTR pszSort,
LONG cMaxLen )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LPCSTR pszSort

排序

LONG cMaxLen

最大长度

2.1.45.18 uaArchiveImport

说明

uaArchivImport 以 CSV 数据格式导入配方。 目标配方的结构必须与导入配方的结构相同。

语法

```
BOOL uaArchiveImport (
UAHARCHIVE hArchive,
LPCSTR pszSource,
LONG lType,
LONG lOptions )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LPCSTR pszSource

源归档的文件名。

LONG lType

源归档的数据格式。 两种格式可用：

UA_FILETYPE_DEFAULT = 0：默认文件格式 = CSV

UA_FILETYPE_CSV = 1：CSV 文件格式

LONG lOptions

保留以供将来扩展。必须为 0。

返回值

TRUE

成功导入配方。

FALSE

错误

2.1.45.19 uaArchiveInsert

说明

将本地数据记录缓冲区插入当前数据库中。 调用 uaArchiveInsert 之前，请使用 “uaArchiveSetFieldValue...”系统函数在本地数据缓冲区的字段中输入数据，以便新数据记录中包含有用的数据。
使用系统函数“uaArchiveSetFieldValueLong”在当前数据记录中输入内部列“ID”。 只能在 C 脚本中使用。

语法

```
BOOL uaArchiveInsert (
UAHARCHIVE hArchive )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

成功插入数据记录。

2.1.45.20 uaArchiveMoveFirst

说明

跳转到第一条数据记录。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveMoveFirst (
UAHARCHIVE hArchive )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

在配方中成功跳转

FALSE

错误

2.1.45.21 uaArchiveMoveLast

说明

跳转到最后一条数据记录。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveMoveLast (
UAHARCHIVE hArchive )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

在配方中成功跳转

FALSE

错误

2.1.45.22 uaArchiveMoveNext

说明

跳转到下一条数据记录。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveMoveNext (
UAHARCHIVE hArchive )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

在配方中成功跳转

FALSE

错误

2.1.45.23 uaArchiveMovePrevious

说明

跳转到前一条数据记录。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveMovePrevious (
UAHARCHIVE hArchive )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

在配方中成功跳转

FALSE

错误

2.1.45.24 uaArchiveOpen

说明

uaArchiveOpen 必须在所有 RT 函数（例如 uaArchiveMoveFirst、uaArchiveMoveLast、uaArchiveMoveNext、uaArchiveMovePrevious、uaArchiveDelete、uaArchiveUpdate、uaArchiveInsert、uaArchiveGetID、uaArchiveGetFields、uaArchiveGetFieldType、uaArchiveGetFieldValueDate、uaArchiveGetFieldValueDouble、uaArchiveGetFieldValueFloat、uaArchiveGetFieldValueLong、uaArchiveGetFieldValueString、uaArchiveSetFieldValueDate、uaArchiveSetFieldValueDouble、uaArchiveSetFieldValueFloat、uaArchiveSetFieldValueLong 和 uaArchiveSetFieldValueString）之前调用。

说明

对配方进行排序和过滤

无需通过“uaArchiveOpen”打开此配方，即可将“uaArchiveSetSort”和“uaArchiveSetFilter”系统函数应用到配方。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveOpen (
UAHARCHIVE hArchive )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

成功打开配方。

FALSE

错误

2.1.45.25 uaArchiveReadTagValues

说明

从域变量中读取当前值。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveReadTagValues (
UAHARCHIVE hArchive,
LONG* pnFields,
LONG cFields,
LONG lOptions )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LONG* pnFields

保留供日后使用 (NULL)

LONG cFields

已传送的域索引数量（数组 pnFields 的大小）。

保留供将来使用 (0)

LONG lOptions

保留供日后使用 (0)

如果 lOptions 值为其它值，则数据在指针的位置插入。

返回值

TRUE

成功读取配方

FALSE

错误

2.1.45.26 uaArchiveReadTagValuesByName

说明

读取当前数据中的变量值。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveReadTagValuesByName (
UAHARCHIVE hArchive,
LPCSTR pszFields,
LONG lOptions )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LPCSTR pszFields

保留供日后使用 (NULL)

LONG lOptions

保留供日后使用 (0)

返回值

TRUE

成功读取配方

FALSE

错误

2.1.45.27 uaArchiveRequery

说明

调用 uaArchiveSetFilter 和 uaArchiveSetSort 后，通过 uaArchiveRequery 重新加载配方。

说明

对配方进行排序和过滤

无需通过“uaArchiveOpen”打开此配方，即可将“uaArchiveSetSort”和“uaArchiveSetFilter”系统函数应用到配方。 在这种情况下，不必调用系统函数“uaArchiveRequery”。

对于下列情况，同样需要调用 uaArchiveRequery：

- 若已在配方视图中输入。
- 若已在要传送到表格域的“Recipes”编辑器中输入了内容。

语法

```
BOOL uaArchiveRequery(
UAHARCHIVE hArchive )
```

只能在 C 脚本中使用。

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

重新查询成功

FALSE

错误

2.1.45.28 uaArchiveSetFieldValueDate

说明

将日期和时间写入当前数据记录的域中。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveSetFieldValueDate (
UAHARCHIVE hArchive,
LONG lField,
LPSYSTEMTIME pstDateTime )
```

参数

UAHARCHIVE hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LONG lField

字段编号，其中 lField = 1 表示指向第一个组态字段的地址。ID 字段的地址表示为 lField = 0。

LPSYSTEMTIME pstDateTime

日期和时间

返回值

TRUE

成功写入日期和时间

FALSE

错误

2.1.45.29 uaArchiveSetFieldValueDouble

说明

将双精度值写入当前数据记录的域中。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveSetFieldValueDouble (
UAHARCHIVE hArchive,
LONG lField,
double dValue )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LONG lField

字段编号，其中 lField = 1 表示指向第一个组态字段的地址。 ID 字段的地址表示为 lField = 0。

double dValue

字段值

返回值

TRUE

成功写入字段值

FALSE

错误

2.1.45.30 uaArchiveSetFieldValueFloat

描述：

将浮点值写入当前数据记录的域中。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveSetFieldValueFloat (
UAHARCHIVE hArchive,
LONG lField,
float fValue )
```

参数：

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LONG lField

字段编号，其中 lField = 1 表示指向第一个组态字段的地址。 ID 字段的地址表示为 lField = 0。

float fValue

字段值

返回值：

TRUE

成功写入字段值

FALSE

错误

2.1.45.31 uaArchiveSetFieldValueLong

说明

将长整型值写入当前数据记录的域中。

只能在 C 脚本中使用。

Syntax

```
BOOL uaArchiveSetFieldValueLong (
UAHARCHIVE hArchive,
LONG lField,
LONG dValue )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LONG lField

字段编号，其中 lField = 1 表示指向第一个组态字段的地址。 ID 字段的地址表示为 lField = 0。

LONG dValue

字段值

返回值

TRUE

成功写入字段值

FALSE

错误

2.1.45.32 uaArchiveSetFieldValueString

说明

将字符串写入当前数据记录的域中。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveSetFieldValueString (
```

```
UAHARCHIVE hArchive,
LONG lField,
LPCSTR pszString )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LONG lField

字段编号，其中 lField = 1 表示指向第一个组态字段的地址。 ID 字段的地址表示为 lField = 0 。

LPCSTR pszString

字段值

返回值

TRUE

成功写入字段值

FALSE

错误

2.1.45.33 uaArchiveSetFilter

说明

设置过滤器。 无需使用“uaArchiveOpen”打开配方，即可调用此系统函数。

说明

如果已使用“uaArchiveOpen”打开配方，则过滤后请使用“uaArchiveRequery”重新加载配方。

只能在 C 脚本中使用。

语法

```
VOID uaArchiveSetFilter (
 UAHARCHIVE hArchive,
```

```
 LPSTR pszFilter )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LPSTR pszFilter

要设置的过滤器。

2.1.45.34 uaArchiveSetSort

说明

设置配方的排序。 无需使用“uaArchiveOpen”打开配方，即可调用此系统函数。

说明

如果已使用“uaArchiveOpen”打开配方，则排序后请使用“uaArchiveRequery”重新加载配方。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveSetSort (
UAHARCHIVE hArchive,
LPSTR pszSort )
```

该系统函数只能用于 C 脚本。

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LPCSTR pszSort

排序

返回值

TRUE

成功设置排序

FALSE

错误

2.1.45.35 uaArchiveUpdate

说明

更新打开的配方。 将配方的所有数据更改都传送给数据库。 配方的组态保持不变。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveUpdate (
UAHARCHIVE hArchive )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

成功更新配方。

FALSE

"Update_failed" 错误 = 106

如出现不一致情况，即会发生该错误， 示例： 在域中设置了“域要求值”标记，但无值输入。

2.1.45.36 uaArchiveWriteTagValues

说明

将当前数据记录的值写入变量中。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveWriteTagValues (
UAHARCHIVE hArchive,
LONG* pnFields,
LONG cFields,
LONG lOptions )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LONG* pnFields

保留供日后使用 (NULL)

LONG cFields

保留供日后使用 (0)

LONG lOptions

保留供日后使用 (0)

返回值

TRUE

成功读取配方

FALSE

错误

2.1.45.37 uaArchiveWriteTagValuesByName

说明

将当前数据记录的值写入变量中。 访问基于配方和域的名称。

只能在 C 脚本中使用。

语法

```
BOOL uaArchiveWriteTagValuesByName (
UAHARCHIVE hArchive,
LPCSTR pszFields,
LONG lOptions )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

LPCSTR pszFields

保留供日后使用 (NULL)

LONG lOptions

保留供日后使用 (0)

返回值

TRUE

成功读取配方

FALSE

错误

2.1.45.38 uaConnect

说明

建立与配方的连接（运行系统）。

只能在 C 脚本中使用。

语法

```
BOOL uaConnect (
 UAHCONNECT* phConnect )
```

参数

UAHCONNECT* phConnect

最新连接配方的句柄的指针。

返回值

TRUE

成功连接配方。

FALSE

错误

2.1.45.39 uaDisconnect

说明

如存在与配方（运行系统）的连接，将断开该连接。

只能在 C 脚本中使用。

语法

```
BOOL uaDisconnect (
 UAHCONNECT hConnect )
```

参数

UAHCONNECT hConnect

已连接的配方的句柄（运行系统）。 此句柄通过 uaConnect 生成。

返回值

TRUE

成功断开配方连接。

FALSE

错误

2.1.45.40 uaGetArchive

说明

读取配方组态。

只能在 C 脚本中使用。

语法

```
BOOL uaGetArchive (
UAHCONFIG hConfig,
long lArchive,
UACONFIGARCHIVE* pArchive )
```

参数

UAHCONFIG hConfig,

配方的组态句柄。 此句柄通过 uaQueryConfiguration 生成。

long lArchive,

归档索引（0 至 (uaGetNumArchives()-1)）

UACONFIGARCHIVE* pArchive

用于接收配方组态的缓冲区的指针

返回值

TRUE

成功访问配方

FALSE

错误

2.1.45.41 uaGetField

描述：

读取域组态。

只能在 C 脚本中使用。

语法

```
BOOL uaGetField (
UAHCONFIG hConfig,
long lArchive,
long lField,
UACONFIGFIELD* pField )
```

参数：

UAHCONFIG hConfig,

配方的组态句柄。 此句柄通过 uaQueryConfiguration 生成。

long lArchive,

归档索引（0 至 (uaGetNumArchives()-1)）

long lField,

域编号，若 lField = 0 表示指向第一个域的地址。

UACONFIGFIELD* pArchive

用于接收域组态的缓冲区的指针。

返回值

TRUE

成功访问配方

FALSE

错误

2.1.45.42 uaGetLastError

说明

WinCC 脚本语言的系统函数返回一个 BOOL 值；TRUE 表示处理过程中未出现错误。 如果返回 FALSE，则可以使用“uaGetLastError()”和“uaGetLastHResult()”读取最后一个系统函数的错误。

只能在 C 脚本中使用。

如果在处理多个系统函数后才调用 uaGetLastError()，那么 uaGetLastError() 将返回最近发生的错误。 只要返回“FALSE”，就应调用“uaGetLastError()”和“uaGetLastHResult()”系统函数以识别触发错误的系统函数。

示例：

```
if ( uaArchiveGetFieldValueLong ( hArchive, Index, &IntValue ) ==
TRUE )
printf( "Field Value = %u\n", IntValue );
else
printf("Error calling uaArchiveGetFieldValueLong: %d / %08lx\n",
uaGetLastError(), uaGetLastHResult());
```

对于不返回值 (VOID) 的系统函数，务必通过调用 uaGetLastError() 对其进行查询。

示例：

```
  uaArchiveGetFilter(hArchive, pszFilter, cMaxLen);
    INT nUAError = uaGetLastError (  );
    if ( UA_ERROR_SUCCESS != nUAError)
    {
        printf( "Filter = [%s]\n", pszFilter );
    }
    else
    {
```

```
        printf("Error calling uaArchiveGetFilter: %d,
hr=0x%08lX\n", nUAError, uaGetLastHResult());
    }

INT uaGetLastError()
```

返回值

最后执行的系统函数的错误状态。 uaGetLastError() 可返回以下错误：

UA_ERROR_SUCCESS

UA_ERROR_GENERIC

UA_ERROR_CONNECT_FAILED

UA_ERROR_OPEN_FAILED

UA_ERROR_CLOSE_FAILED

UA_ERROR_REQUERY_FAILED

UA_ERROR_MOVE_FAILED

UA_ERROR_INSERT_FAILED

UA_ERROR_UPDATE_FAILED

UA_ERROR_DELETE_FAILED

UA_ERROR_IMPORT_FAILED

UA_ERROR_EXPORT_FAILED

UA_ERROR_READ_FAILED

UA_ERROR_WRITE_FAILED

UA_ERROR_GET_FAILED

UA_ERROR_SET_FAILED

UA_ERROR_INVALID_NAME

UA_ERROR_INVALID_TYPE

UA_ERROR_INVALID_NUMRECS

UA_ERROR_INVALID_COMMTYPE

UA_ERROR_INVALID_LENGTH

UA_ERROR_INVALID_PRECISION

UA_ERROR_NULL_POINTER

UA_ERROR_INVALID_POINTER

UA_ERROR_INVALID_HANDLE

UA_ERROR_INVALID_INDEX

UA_ERROR_SERVER_UNKNOWN

这些错误常量以及用户归档例程的预定义内容均位于 CCUACAPI.H. 中

2.1.45.43 uaGetLastHResult

说明

读取最后发生的 COM 错误。 此系统函数主要用于分析 COM 执行过程中的不兼容问题，或用于识别注册错误和通信错误。
如果某个用户归档系统函数（如 uaConnect）返回值"FALSE"来指示错误，那么除了 UAGetLastError 外，还必须同时调用此系统函数。

只能在 C 脚本中使用。

语法

```
LONG uaGetLastHResult()
```

返回值

最后发生的 COM 错误

2.1.45.44 uaGetNumArchives

说明

读取当前已组态的配方的数目。

只能在 C 脚本中使用。

语法

```
LONG uaGetNumArchives (
```

```
UAHCONFIG hConfig )
```

参数

UAHCONFIG hConfig
配方的组态句柄。 此句柄通过 uaQueryConfiguration 生成。

返回值

当前已组态的配方的数目。 如果发生错误，将返回 -1。

2.1.45.45 uaGetNumFields

说明

提供已组态域的数目。 不包括"ID"、“上个用户”和“上次访问”域。 在组态调用中使用 0 至 uaGetNumFields() -1 指定索引。

只能在 C 脚本中使用。

语法

```
LONG uaGetNumFields (
UAHCONFIG hConfig,
long lArchive )
```

参数

UAHCONFIG hConfig,
配方的组态句柄。 此句柄通过 uaQueryConfiguration 生成。

long lArchive,
归档索引（0 至 (uaGetNumArchives()-1)）

返回值

已组态域的数目。 如果发生错误，将返回 -1。

2.1.45.46 uaQueryArchive

说明

建立与配方（用于运行系统操作）的连接。UaQueryArchive 创建 UAHARCHIVE 句柄。

只能在 C 脚本中使用。

语法

```
BOOL uaQueryArchive (
UAHCONNECT hConnect,
 LONG lArchive,
 UAHARCHIVE* phArchive )
```

参数

UAHCONNECT hConnect

已连接配方的句柄（运行系统）。此句柄通过 uaConnect 生成。

LONG lArchive

待连接归档的 ID

UAHARCHIVE* phArchive

配方句柄的指针。

返回值

TRUE

成功生成配方句柄。

FALSE

错误

注释

如果在客户端项目中使用针对冗余服务器对的用户归档函数，则切换主站时用户归档连接无法自动切换至新主站。在这种情况下，所有用户归档调用都返回 LastError UA_ERROR_SERVER_UNKNOWN = 1004，这表示用户程序必须执行新的 uaQueryArchive() 或 uaQueryArchiveByName() 和 uaArchiveOpen()。

2.1.45.47 uaQueryArchiveByName

说明

通过配方名称建立与配方（用于运行系统操作）的连接。 UaQueryArchiveByName 为配方创建 UAHARCHIVE 句柄。

只能在 C 脚本中使用。

语法

```
BOOL uaQueryArchiveByName (
UAHCONNECT hConnect,
 LPCSTR pszName,
 UAHARCHIVE* phArchive )
```

参数

UAHCONNECT hConnect

已连接配方的句柄（运行系统）。 此句柄通过 uaConnect 生成。

LPCSTR pszName

配方名称。 对于客户机项目，如使用的是缺省服务器以外的服务器，则可添加服务器前缀"::"作为配方名称的分隔符。

UAHARCHIVE* phArchive

配方句柄的指针。

返回值

TRUE

成功生成配方句柄。

FALSE

错误

注释

如果在客户端项目中使用针对冗余服务器对的用户归档函数，则切换主站时用户归档连接无法自动切换至新主站。 在这种情况下，所有用户归档调用都返回 LastError UA_ERROR_SERVER_UNKNOWN = 1004，这表示用户程序必须执行新的 uaQueryArchive() 或 uaQueryArchiveByName() 和 uaArchiveOpen()。

2.1.45.48 UaQueryConfiguration

说明

建立与配方（用于组态）的连接。

只能在 C 脚本中使用。

语法

```
BOOL uaQueryConfiguration (
UAHCONFIG* phConfig )
```

参数

UAHCONFIG* phConfig,
配方句柄的指针。

返回值

TRUE

成功访问配方。

FALSE

错误

2.1.45.49 uaReleaseArchive

说明

释放与当前配方的连接。

只能在 C 脚本中使用。

语法

```
BOOL uaReleaseArchive (
 UAHARCHIVE hArchive )
```

参数

UAHARCHIVE hArchive

配方的句柄。 此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

成功释放与配方的连接。

FALSE

错误

注释

成功释放后必须将"hArchive"句柄设置为"NULL"以完全确保在日后进一步使用失效的句柄而 COM 接口中又没有对应的函数时将触发"UA_ERROR_INVALID_HANDLE"错误。

2.1.45.50 uaReleaseConfiguration

说明

释放与配方的连接（组态）。

只能在 C 脚本中使用。

语法

```
BOOL uaReleaseConfiguration (
UAHCONFIG hConfig,
BOOL bSave )
```

参数

UAHCONFIG hConfig

配方的组态句柄。 此句柄通过 uaQueryConfiguration 生成。

BOOL bSave

释放组态的配方连接之前保存所作的组态更改。

TRUE = 保存更改，FALSE = 放弃更改

警告： 只能在运行系统未激活时使用“保存更改”（bSave = TRUE）！ 您可以通过请求 uaIsActive() 来检查运行系统是否为激活状态。

返回值

TRUE

成功释放连接

FALSE

错误

2.2 C-bib

2.2.1 ctype 函数

函数概述

可使用下列 ctype 函数：

- long int isalnum (long int x);
- long int isalpha (long int x);
- long int isdigit (long int x);

- long int isgraph (long int x);
- long int islower (long int x);
- long int isprint (long int x);
- long int ispunct (long int x);
- long int isspace (long int x);
- long int isupper (long int x);
- long int isxdigit (long int x);
- long int tolower (long int x);
- long int toupper (long int x);

有关 ctype 函数的介绍，请参见有关 C 语言编程专业手册。

该函数只能用于 C 脚本。

2.2.2 函数组 c_bib

简介

c_bib 函数组包含 C 库中的 C 函数，并分成：

- ctype
- math
- memory
- stdio
- stdlib
- string
- time

stdio 本身又分为：

- char_io
- directio
- error

- file
- file_pos
- output

有关这些函数的介绍，参见相关的专业手册。

WinCC 函数的特别功能

发出日期时，localtime 函数具有以下特性：

- 月计数从 0 开始。
- 年计数从 1990 年开始，初值为 0。

只有 C 库的 printf()、sprintf()、fprintf() 函数可在 WinCC 中编辑，最多 360 个字符。

2.2.3 数学函数

函数概述

可使用下列 math 函数：

- double acos (double x);
- double asin (double x);
- double atan (double x);
- double atan2 (double x, double y);
- double ceil (double x);
- double cos (double x);
- double cosh (double x);
- double exp (double x);
- double fabs (double x);
- double floor (double x);
- double fmod (double x, double y);
- double frexp (double x, long int* y);
- double ldexp (double x, long int y);
- double log (double x);

- double log10 (double x);
- double modf (double x, double* y);
- double pow (double x, double y);
- double sin (double x);
- double sinh (double x);
- double sqrt (double x);
- double tan (double x);
- double tanh (double x);

有关 math 函数的介绍，请参见有关 C 语言编程专业手册。

该函数只能用于 C 脚本。

2.2.4 memory 函数

函数概述

可使用下列 memory 函数：

- long int memcmp (const void* cs, const void* ct, size_t n);
- void* memchr (const void* cs, long int c, size_t n);
- void* memcpy (void* s, const void* ct, size_t n);
- void* memmove (void* s, const void* ct, size_t n);
- void* memset (void* s, long int c, size_t n);

有关 memory 函数的介绍，请参见有关 C 语言编程专业手册。

该函数只能用于 C 脚本。

2.2.5 multibyte 函数

函数概述

可使用下列 multibyte 函数：

- int _ismbcalnum(unsigned int c)
- int _ismbcalpha(unsigned int c)
- int _ismbcdigit(unsigned int c)
- int _ismbcgraph(unsigned int c)
- int _ismbclower(unsigned int c)
- int _ismbcprint(unsigned int c)
- int _ismbcpunct(unsigned int c)
- int _ismbcspace(unsigned int c)
- int _ismbcupper(unsigned int c)
- int _mbscmp(const unsigned char *string1, const unsigned char *string2)
- int _mbsncmp(const unsigned char *string1, const unsigned char *string2, size_t count)
- int _mbsrchr(const unsigned char *string, unsigned int c)
- size_t _mbscspn(const unsigned char *string, const unsigned char *strCharSet)
- size_t _mbsspn(const unsigned char *string, const unsigned char *strCharSet)
- size_t _mbstrlen(const char *string)
- size_t _mbslen(const unsigned char *string)
- unsigned char *_mbscat(unsigned char *strDestination, const unsigned char *strSource)
- unsigned char *_mbschr(const unsigned char *string, unsigned int c)
- unsigned char *_mbscpy(unsigned char *strDestination, const unsigned char *strSource)
- unsigned char *_mbsdec(const unsigned char *start, const unsigned char *current)
- unsigned char *_mbsinc(const unsigned char *current) size_t _mbclen(const unsigned char *c);

- unsigned char *_mbsncat(unsigned char *strDest, const unsigned char *strSource, size_t count)
- unsigned char *_mbsncpy(unsigned char *strDest, const unsigned char *strSource, size_t count)
- unsigned char *_mbspbrk(const unsigned char*string, const unsigned char *strCharSet)
- unsigned char *_mbsstr(const unsigned char *string, const unsigned char *strCharSet)
- unsigned char *_mbstok(unsigned char*strToken, const unsigned char *strDelimit)

有关 multibyte 函数的描述，请参见有关 C 语言编程专业手册。

该函数只能用于 C 脚本。

2.2.6 stdio 函数

函数概述

可使用下列 stdio 函数：

- char* fgets (char* s, long int n, FILE* stream);
- char* tmpnam (char* s);
- FILE* fopen (const char* name, const char* mode);
- FILE* freopen (const char* filename, const char* mode, FILE* stream);
- FILE* tmpfile ();
- fprintf();
- long int fclose (FILE* stream);
- long int feof (FILE* stream);
- long int ferror (FILE* stream);
- long int fflush (FILE* stream);
- long int fgetc (FILE* stream);
- long int fgetpos (FILE* stream, fpos_t* ptr);
- long int fputc (long int c, FILE* stream);
- long int fputs (const char* s, FILE* stream);

- long int fseek (FILE* stream, long int offset, long int origin);
- long int fsetpos (FILE* stream, const fpos_t* ptr);
- long int ftell (FILE* stream);
- long int getc (FILE* stream);
- long int putc (long int c, FILE* stream);
- long int remove (const char* filename);
- long int rename (const char* oldname, const char* newname);
- long int setvbuf (FILE* stream, char* buf, long int mode, size_t size);
- long int ungetc (long int c, FILE* stream);
- long int vfprintf (FILE* stream, const char* format, va_list arg);
- long int vsprintf (char* s, const char* format, va_list arg);
- printf();
- size_t fread (void* ptr, size_t size, size_t nobj, FILE* stream);
- size_t fwrite (void* ptr, size_t size, size_t nobj, FILE* stream);
- void clearerr (FILE* stream);
- void rewind (FILE* stream);
- void setbuf (FILE* stream, char* buf);

有关 studio 函数的介绍，请参见有关 C 语言编程专业手册。

该函数只能用于 C 脚本。

2.2.7 stdlib 函数

函数概述

可使用下列 stdlib 函数：

- char* getenv (const char* name);
- div_t div (long int num, long int denom);
- double atof (const char* s);
- double strtod (const char* s, char** endp);

- ldiv_t ldiv (long int num, long int denom);
- long int abs (long int n);
- long int atoi (const char* s);
- long int atol (const char* s);
- long int labs (long int n);
- long int rand ();
- long int srand (unsigned long int seed);
- long int strtol (const char* s, char** endp, long int base);
- long int system (const char* s);
- unsigned long int strtoul (const char* s, char** endp, long int base);
- void abort ();
- void* bsearch (const void* key, const void* base, size_t n, size_t size, long int(* cmp) (const
- void* calloc (size_t nobj, size_t size);
- void exit (long int status);
- void free (void* p);
- void* keyval, const void* datum));
- void* malloc (size_t size);
- void qsort (void* base, size_t n, size_t size, long int* cmp, const void* , const void*);
- void* realloc (void* p, size_t size);

有关 stdlib 函数的描述，请参见有关 C 语言编程专业手册。

该函数只能用于 C 脚本。

2.2.8 string 函数

函数概述

可使用下列 string 函数：

- char* strcat (char* s, const char* ct);
- char* strchr (const char* cs, long int c);

- char* strcpy (char* s, const char* ct);
- char* strerror (size_t n);
- char* strncat (char* s, const char* ct, size_t n);
- char* strncpy (char* s, const char* ct, size_t n);
- char* strpbrk (const char* cs, const char* ct);
- char* strrchr (const char* cs, long int c);
- char* strstr (const char* cs, const char* ct);
- char* strtok (char* s, const char* ct);
- long int strcmp (const char* cs, const char* ct);
- long int strncmp (const char* cs, const char* ct, size_t n);
- size_t strcspn (const char* cs, const char* ct);
- size_t strlen (const char* cs);
- size_t strspn (const char* cs, const char* ct);

有关 string 函数的描述，请参见有关 C 语言编程专业手册。

该函数只能用于 C 脚本。

2.2.9 time 函数

函数概述

可使用下列 time 函数：

- char* asctime (const struct tm* tp);
- char* ctime (const time_t* tp);
- clock_t clock ();
- double difftime (time_t time2, time_t time1);
- size_t strftime (char* s, size_t smax, const char* fmt, const struct tm* tp);
- struct tm* gmtime (const time_t* tp);
- struct tm* localtime (const time_t* tp);

- time_t mktime (struct tm* tp);
- time_t time (time_t* tp);

有关 time 函数的描述，请参见有关 C 语言编程专业手册。

该函数只能用于 C 脚本。

2.3 结构定义

2.3.1 结构定义 CCAPErrorExecute

```
typedef struct {
DWORD dwCurrentThreadID; 当前线程的线程 ID
DWORD dwErrorCode1;      错误代码 1
DWORD dwErrorCode2;      错误代码 2
BOOL bCycle;             cycle/acycle
char* szApplicationName; 应用名称
char* szFunctionName;    函数名称
char* szTagName;         变量名称
LPVOID lpParam;          操作堆栈的指针
DWORD dwParamSize;       操作堆栈大小
DWORD dwCycle;           变量周期
CMN_ERROR* pError;       CMN_ERROR 的指针
} CCAPErrorExecute;
```

Members

各个错误代码的含义以及其相关的结构元素，请参见下表：

dwErrorCode1	dwErrorCode2	bCycle	szApplicationName	szFunctionName	szTagName	lpParamSize	dwParamSize	dwCycle	pError	description
1007001	0	x	x	x		x	x			操作请求异常
1007001	1	x	x	x		x	x			访问返回结果时异常
1007001	4097	x	x	x		x	x			执行操作时堆栈溢出
1007001	4098	x	x	x		x	x			0 作为除法操作的除数

1007001	4099	x	x	x		x	x			操作包含对不存在的符号的访问
1007001	4100	x	x	x		x	x			操作包含访问违例
1007004	0	x	x	x						函数未知
1007005	1	x	x							操作不包含 P 代码。
1007005	2	x	x							不正确的函数名称
1007005	4	x	x	x		x	x			返回值类型无效
1007005	32768ff	x	x	x		x	x			加载操作时，Ciss Compiler 出错
1007006	0	x	x	x	x	x	x	x		未定义变量
1007006	1	x	x	x	x	x	x	x		变量超时
1007006	2	x	x	x	x	x	x	x	x	变量无法以指定格式返回
1007006	3	x	x	x	x	x	x	x	x	变量返回状态违例，状态存在于 CMN_ERROR.dwError1 中
1007007	1	x	x	x		x	x		x	PDLRTGetProp 错误
1007007	2	x	x	x		x	x		x	PDLRTSetProp 错误
1007007	3	x	x	x		x	x		x	DM 调用错误

错误结构

若在 pError 列标记为“x”，则 OnErrorExecute 使用错误结构来判断或输出错误消息。

2.3.2 结构定义 CCAPTime

```
typedef struct {
DWORD dwCurrentThreadID; 当前线程的线程 ID
DWORD dwCode;            代码
BOOL bCycle;             cycle/acycle
char* szApplicationName; 应用名称
char* szFunctionName;    函数名称
LPVOID lpParam;          操作堆栈的指针
DWORD dwParamSize;       操作堆栈大小
double dblTime;
DWORD dwFlags;           标记
} CCAPTime;
```

Members

dwCode

结构元素 dwCode 提供关于 OnTime 调用的信息：

dwCode = 113	调用每个操作的时间定义
dwCode = 114	调用每个操作的时间监控

dwFlags

结构元素 dwFlags 提供关于输出类型的的信息：

dwFlags = TRUE	结果输出到文件中
dwFlags = FALSE	结果输出到诊断窗口

2.3.3 结构定义 CMN_ERROR

```
struct CMNERRORSTRUCT {
DWORD       dwError1,
DWORD       dwError2,
DWORD       dwError3,
DWORD       dwError4,
DWORD       dwError5;
TCHAR       szErrorText[MAX_ERROR_LEN];
}
CMN_ERROR
```

说明

扩展的错误结构包含已发生错误的错误代码和错误文本。 每个应用都可以通过错误结构来判断或输出错误信息。

Members

dwError1 .. dwError5

API 函数可以任何方式使用这些项。

API 中描述了发生错误时相关项所包含值的信息。 若未指定，则错误代码存在于 dwError1 中。

szErrorText

有关错误原因文本说明的缓冲区

内容由资源决定，因此具有语言相关性。

2.3.4 结构定义 DM_TYPEREF

```
typedef struct {
DWORD dwType;
DWORD dwSize;
char szTypeName[MAX_DM_TYPE_NAME + 1];
}
DM_TYPEREF;
```

Members

dwType

指定变量类型。

dwType	PLC 数据类型	HMI 数据类型
DM_VARTYPE_BIT	二进制变量	Bool
DM_VARTYPE_SBYTE	有符号 8 位数	Byte
DM_VARTYPE_BYTE	无符号 8 位数	UByte
DM_VARTYPE_SWORD	有符号 16 位数	Short
DM_VARTYPE_WORD	无符号 16 位数	UShort
DM_VARTYPE_SDWORD	有符号 32 位数	Integer
DM_VARTYPE_DWORD	无符号 32 位数	UInteger
DM_VARTYPE_FLOAT	32 位 IEEE 754 浮点数	Float
DM_VARTYPE_DOUBLE	64 位 IEEE 754 浮点数	Double
DM_VARTYPE_TEXT_8	文本变量，8 位字符集	Char
DM_VARTYPE_TEXT_16	文本变量，16 位字符集	String
DM_VARTYPE_RAW	原始数据类型	Raw
DM_VARTYPE_STRUCT	结构变量	Struct
DM_VARTYPE_TEXTREF	文本参考变量	String

dwSize

以字节为单位，指定数据类型的长度。

szTypeName

若是结构变量，则包含结构类型的名称。

2.3.5 结构定义 DM_VAR_UPDATE_STRUCT

```
typedef struct {
DM_TYPEREF dmTypeRef;
DM_VARKEY dmVarKey;
VARIANT dmValue;
DWORD dwState;
}
DM_VAR_UPDATE_STRUCT;
```

Members

dmTypeRef

包含关于数据类型的信息。 基于性能的考虑，若是循环请求，则并不会向该结构输入任何内容。

dmVarKey

选择要编辑的变量。

dmValue

变量值

若要访问 VARIANT 值，则必须在 VARIANT 名称和项名称之间插入".u."。

示例

```
// 提供变量
myVariant.vt = VT_I4;
myVariant.u.lVal = 233;
```

有关 VARIANT 数据类型的描述，请参见相关文档。 首次使用前，VARIANT dmValue 必须使用 VariantInit() 进行初始化，使用之后通过 VariantClear(&dmValue) 再次启用。 因此，具有 ZeroMemory() 或 memset() 的结构 DM_VAR_UPDATE_STRUCT 一定不能删除。

dwState

标识变量状态。

2.3.6 结构定义 DM_VAR_UPDATE_STRUCTEX

```
typedef struct {
DM_TYPEREF dmTypeRef;
DM_VARKEY dmVarKey;
VARIANT dmValue;
DWORD dwState;
DWORD dwQualityCode;
}
DM_VAR_UPDATE_STRUCTEX;
```

Members

dmTypeRef

包含关于数据类型的信息。 基于性能的考虑，若是循环请求，则并不会向该结构输入任何内容。

dmVarKey

选择要编辑的变量。

dmValue

变量值

若要访问 VARIANT 值，则必须在 VARIANT 名称和项名称之间插入“.u.”。

示例

```
// 提供变量
myVariant.vt = VT_I4;
myVariant.u.lVal = 233;
```

有关 VARIANT 数据类型的描述，请参见相关文档。 首次使用前，VARIANT dmValue 必须使用 VariantInit() 进行初始化，使用之后通过 VariantClear(&dmValue) 再次启用。 因此，具有 ZeroMemory() 或 memset() 的结构 DM_VAR_UPDATE_STRUCTEX 一定不能删除。

dwState

标识变量状态。

dwQualityCode

标识 QualityCode 变量。

2.3.7 结构定义 DM_VARKEY

```
typedef struct {
DWORD dwKeyType;
DWORD dwID;
char szName[ MAX_DM_VAR_NAME + 1 ];
LPVOID lpvUserData;
}
DM_VARKEY;
```

Members

dwKeyType

定义通过关键字 ID 还是名称来寻址变量。

DM_VARKEY_ID 通过关键字 ID 指定

DM_VARKEY_NAME 通过变量名指定

dwID

若相应设置了 dwKey 类型，则包含变量的关键字 ID。

szName

若相应设置了 dwKey 类型，则包含变量的名称。

lpvUserData

特定应用数据的指针

2.3.8 结构定义 LINKINFO

```
typedef struct {
LINKTYPE LinkType;
DWORD dwCycle;
TCHAR szLinkName[256];
}
LINKINFO;
```

Members

LinkType

LinkType 是在“Trigger.h”文件中定义的枚举常量。 通过“#include "Trigger.h”命令可将它们与脚本和相应的枚举常量集成。

BUBRT_LT_NOLINK	0	无快捷方式
BUBRT_LT_VARIABLE_DIRECT	1	直接变量
BUBRT_LT_VARIABLE_INDIRECT	2	间接变量
BUBRT_LT_ACTION	3	C 操作
BUBRT_LT_ACTION_WIZARD	4	动态对话框
BUB_LT_DIRECT_CONNECTION	5	直接连接
BUBRT_LT_ACTION_WIZARD_INPROC	6	动态对话框

对于 SetLink 函数，只能使用枚举常量 BUBRT_LT_VARIABLE_DIREKT 和 BUBRT_LT_VARIABLE_INDIRECT 。 GetLink 函数用于返回所有列出的枚举常量。

dwCycle

更新周期时间

255	画面周期
235	窗口周期
0	根据更改
1	250ms
2	500 ms
3	1 s
4	2 s
5	5s
6	10s
7	1min
8	5min
9	10min

10	1h
11-15	用户周期 1-5

szLinkName

变量名称

参见

SetLink (页 1691)

GetLink (页 1618)

2.3.9 结构定义 MSG_FILTER_STRUCT

```
typedef struct {
CHAR        szFilterName[MSG_MAX_TEXTLEN+1];
WORD        dwFilter;
SYSTEMTIME  st[2];
DWORD       dwMsgNr[2];
DWORD       dwMsgClass;
DWORD       dwMsgType[MSG_MAX_CLASS];
DWORD               dwMsgState;
WORD        wAGNr[2];
WORD        wAGSubNr[2];
DWORD       dwArchivMode;
char        szTB[MSG_MAX_TB][
MSG_MAX_TB_CONTENT+1]
DWORD       dwTB;
Double      dPValue[MSG_MAX_PVALUE][2];
DWORD       dwPValue[2];
DWORD       dwMsgCounter[2];
DWORD       dwQuickSelect;
}
MSG_FILTER_STRUCT;
```

说明

在该结构中指定标准。

Members

dwFilter

通过文件"m_global.h"中的以下常量定义过滤条件：

MSG_FILTER_DATE_FROM	开始日期
MSG_FILTER_DATE_TO	结束日期
MSG_FILTER_TIME_FROM	开始时间
MSG_FILTER_TIME_TO	结束时间
MSG_FILTER_NR_FROM	消息起始编号
MSG_FILTER_NR_TO	消息结束编号
MSG_FILTER_CLASS	消息类别
MSG_FILTER_STATE	消息状态
MSG_FILTER_AG_FROM	AS 起始编号
MSG_FILTER_AG_TO	AS 结束编号
MSG_FILTER_AGSUB_FROM	AG 起始子编号
MSG_FILTER_AGSUB_TO	AG 结束子编号
MSG_FILTER_TEXT	消息文本
MSG_FILTER_PVALUE	过程值

MSG_FILTER_COUNTER_FROM	内部消息计数器初值
MSG_FILTER_COUNTER_TO	内部消息计数器终值
MSG_FILTER_PROCESSMSG	过程消息
MSG_FILTER_SYSMSG	系统消息
MSG_FILTER_BEDMSG	操作员消息
MSG_FILTER_DATE	日期跨度
MSG_FILTER_TIME	时间跨度
MSG_FILTER_NR	消息编号范围
MSG_FILTER_VISIBLEONLY	显示可见消息
MSG_FILTER_HIDDENONLY	显示隐藏消息

st

日期/时间范围

其中 st[0] 是起始时间 (from)，st[1] 是结束时间 (to)

指定这些域的条件： MSG_FILTER_DATE, MSG_FILTER_DATE_FROM, MSG_FILTER_DATE_TO, MSG_FILTER_TIME, MSG_FILTER_TIME_FROM, bzw. MSG_FILTER_TIME_TO

若为传送 SYSTEMTIME- 参数需要当前的时间，使用 GetLocalTime 函数而非 GetSystemTime.。 通常，这两种函数区别很大。

dwMsgNr

消息编号范围

其中 dwMsgNr[0] 是起始编号 (from)，dwMsgNr[1] 是结束编号 (to)

指定这些域的条件： MSG_FILTER_NR, MSG_FILTER_NR_FROM bzw. MSG_FILTER_NR_TO

dwMsgClass

按位编码的消息类别。

指定该域的条件： MSG_FILTER_CLASS

dwMsgType

按报警类别划分的消息类型，按位编码

指定该域的条件： MSG_FILTER_CLASS

dwMsgState

按位编码的消息状态。

指定该域的条件： MSG_FILTER_STATE

wAGNr

AGNr 范围

指定这些域的条件：MSG_FILTER_AG_FROM 或 MSG_FILTER_AG_TO

wAGSubNr

AGSubNr 范围

指定该域的条件：MSG_FILTER_AGSUB_FROM 或 MSG_FILTER_AGSUB_TO

dwArchivMode

记录/报告

必须指定 0。

szTB

文本块的文本

指定这些域的条件： MSG_FILTER_TEXT

dwTB

活动文本块（范围、按位编码）

指定该域的条件： MSG_FILTER_TEXT

dPValue

过程值范围

指定这些域的条件： MSG_FILTER_PVALUE

dwPValue

当前过程值（范围、按位编码）

指定该域的条件： MSG_FILTER_PVALUE

dwMsgCounter

内部消息计数器范围

指定这些域的条件： MSG_FILTER_COUNTER_FROM, MSG_FILTER_COUNTER_TO

dwQuickSelect

快速选择小时、天、月

该参数预留作将来升级，必须预设为 0。

指定该域的条件： MSG_FILTER_QUICKSELECT

LOWORD 类型：

MSG_FILTER_QUICK_MONTH	快速选择最后 n 个月
MSG_FILTER_QUICK_DAYS	快速选择最后 n 天
MSG_FILTER_QUICK_HOUR	快速选择最后 n 个小时

HIWORD 的值： 1...n

快速选择的终止时间，请参考本地计算机的当前系统时间。 起始时间则可以通下方式进行回推：n *（月、天、小时）。

2.3.10 结构定义 MSG_RTDATA_STRUCT

```
typedef struct {
DWORD                dwMsgState;
DWORD                dwMsgNr;
SYSTEMTIME           stMsgTime;
DWORD                dwTimeDiff;
DWORD                dwCounter;
DWORD                dwFlags;
WORD                 wPValueUsed;
WORD                 wTextValueUsed;
double               dPValue[MSG_MAX_PVALUE];
MSG_TEXTVAL_STRUCT   mtTextValue[MSG_MAX_PVALUE];
}
MSG_RTDATA_STRUCT;
```

Members

dwMsgState

消息状态

MSG_STATE_COME	0x00000001	消息输入
MSG_STATE_GO	0x00000002	消息输出
MSG_STATE_QUIT	0x00000003	消息确认
MSG_STATE_LOCK	0x00000004	消息锁定
MSG_STATE_UNLOCK	0x00000005	报警锁定
MSG_STATE_QUIT_SYSTEM	0x00000010	系统确认的消息
MSG_STATE_QUIT_EMERGENCY	0x00000011	紧急情况确认
MSG_STATE_QUIT_HORN	0x00000012	报警更改确认
MSG_STATE_COMEGO	0x00000013	输入和输出的消息，仅在“当前消息”画面中显示
MSG_STATE_UPDATE	0x00010000	消息更新位
MSG_STATE_RESET	0x00020000	消息复位位
MSG_STATE_SUMTIME	0x00040000	夏时制激活位
MSG_STATE_INSTANCE	0x00080000	实例消息位（第 n 条消息)

dwMsgNr

消息编号

stMsgTime

日期/时间： 由调用函数决定的报文时间

dwTimeDiff

输入时间/报文时间（单位：秒）

dwCounter

内部消息计数器

dwFlags

数据库的消息标记

MSG_FLAG_SUMTIME	0x00000001	激活夏时制
MSG_FLAG_COMMENT	0x00000002	消息包含注释
MSG_FLAG_ARCHIV	0x00000004	归档
MSG_FLAG_PROTOCOL	0x00000008	Logging
MSG_FLAG_TEXTVALUES	0x00000010	消息包含文本和数值
MSG_FLAG_TIMEINVALID	0x00000020	表示无效日期/时间戳的位
MSG_FLAG_INSTANCE	0x00000040	实例消息标识 (185269)

wPValueUsed

所用的过程值，按位编码。 每个位只可能在结构元素“wPValueUsed”或“wTextValueUsed”之一中设置。 相应的值可以是数值或文本。

wTextValueUsed

使用的文本值，按位编码。 每个位只可能在结构元素“wPValueUsed”或“wTextValueUsed”之一中设置。 相应的值可以是数值或文本。

运行系统 API

3

3.1 运行系统 API

简介

运行系统 API 描述 WinCC 的开放式编程接口。借助 API 函数，可以在单独的应用程序中使用 WinCC 的内部函数以及访问 HMI 变量的数据或归档数据。

说明

Siemens 对于通过 API 接口传输的数据和信息是否与第三方软件兼容不承担任何责任，也不承诺任何担保。

我们明确指出，若是 API 接口使用不当，可能导致数据丢失或生产中断。

示例：

- MSRTCreateMsgPlus()：创建报警
- DMGetValue()：获取变量值

要求

- 已安装编程环境，如 MS Visual Studio
- 已安装 WinCC Runtime Professional。

使用

可在以下位置使用 API 函数:

- WinCC 内部：在用户自定义 C 函数和局部 C 脚本中。
- WinCC 外部：使用 C/C++ 编程语言创建的 Windows 应用程序中。要使用以编程语言 C# 或 VB.net 编写的运行系统 API，必须编写相应的转换程序。

包括文件和库文件

要生成可执行代码，需要包括文件和库文件。这些文件位于以下目录下：

- %ProgramFiles%\SIEMENS\Automation\SCADA-RT_V11\WinCC\include
- %ProgramFiles%\SIEMENS\Automation\SCADA-RT_V11\WinCC\lib

3.2 数据管理函数

3.2.1 基本知识

3.2.1.1 结构概览

概述

DM_CONNECTION_DATA (页 1839)	连接数据
DM_CONNKEY (页 1840)	列出连接数据（过滤结构）
DM_CYCLE_INFO (页 1841)	更新周期
DM_DATA_SERVICE (页 1842)	数据传输通道
DM_DIRECTORY_INFO (页 1843)	组态数据的路径和文件名
DM_DLGOPTIONS (页 1844)	对话框规范
DM_MACHINE_TABLE (页 1846)	格式转换
DM_PROJECT_INFO (页 1847)	项目信息
DM_SD_TARGET_APP (页 1852)	指定应用程序
DM_SD_TARGET_MACHINE (页 1851)	计算机信息
DM_SEND_DATA_STRUCT (页 1848)	指定数据交换伙伴
DM_TYPEREF (页 1852)	变量类型首选项
DM_VAR_UPDATE_STRUCTEX (页 1856)	获取变量值
DM_VAR_UPDATE_STRUCT (页 1854)	获取变量值
DM_VARFILTER (页 1859)	选择变量
DM_VARGRP_DATA (页 1861)	有关变量组的信息
DM_VARGRPKEY (页 1862)	指定变量组
DM_VARIABLE_DATA4 (页 1866)	包括变量组的相关信息

DM_VARIABLE_DATA (页 1863)	包括变量组的相关信息
DM_FORMAT_INFO (页 1845)	格式转换
DM_VARKEY (页 1869)	指定变量
DM_VARLIMIT (页 1871)	变量限制
MCP_NEWVARIABLE_DATA_4 (页 1875)	变量定义
MCP_NEWVARIABLE_DATA_5 (页 1878)	变量定义
MCP_NEWVARIABLE_DATA_EX4 (页 1883)	变量定义
MCP_NEWVARIABLE_DATA_EX (页 1880)	变量定义
MCP_NEWVARIABLE_DATA (页 1873)	变量定义
MCP_VARIABLE_COMMON_EX (页 1887)	变量描述
MCP_VARIABLE_COMMON (页 1885)	变量描述
MCP_VARIABLE_LIMITS_EX (页 1894)	变量限制
MCP_VARIABLE_LIMITS5 (页 1892)	变量限制
MCP_VARIABLE_LIMITS (页 1890)	变量限制
MCP_VARIABLE_PROTOCOL_EX (页 1897)	变量限制处理
MCP_VARIABLE_PROTOCOL (页 1896)	变量限制处理
MCP_VARIABLE_SCALES (页 1898)	变量标定

参见

VariableStateType 属性 (页 1821)

函数概览 (页 1813)

3.2.1.2 函数概览

概述

DM_ENUM_CYCLES_PROC (页 1923)	列出更新周期（回调）
DM_ENUM_FORMATS_PROC (页 1919)	列出格式转换（回调）
DM_NOTIFY_PROC (页 1912)	通知函数
DMActivateRTProject (页 1899)	激活项目
DMChangeDataLocale (页 1908)	语言切换（通知应用程序）
DMConnect (页 1909)	与数据管理器进行连接

DMDeactivateRTProject (页 1915)	取消激活运行项目
DMDisconnect (页 1916)	终止与数据管理器的连接
DMEnumNumberFormats (页 1918)	列出格式转换
DMEnumUpdateCycles (页 1921)	列出更新周期
DMExitWinCCEx (页 1925)	默认退出 WinCC
DMExitWinCC (页 1924)	退出 WinCC
DMGetConnectionState (页 1929)	查询与数据管理器的连接
DMGetHotkey (页 1932)	获取热键 ID
DMGetMachineInfo (页 1933)	查询应用程序的启动参数
DMGetMachineTable (页 1934)	查询计算机列表

参见

DMGetDataLocale (页 1930)

结构概览 (页 1812)

转换例程（控制中心） (页 1831)

3.2.1.3 HMI 变量的质量代码

简介

需要使用"Quality Code"来评估变量的状态和质量。 相应 HMI 变量的整个值传送以及值处理的质量通过相关的 Quality Code 来体现。 例如，可根据 Quality Code 来确定当前值是起始值还是替换值。

质量代码具有优先次序。 如果同时产生多个代码，则显示反映最差质量的 Quality Code。

质量代码的计算

可通过多种不同方法来计算 Quality Code：

- 使用 VB 函数计算
- 使用 C 函数计算
- 使用 I/O 域的“Quality Code 已更改”事件计算。

说明

为了将整个值传送和值处理包含在过程变量的 Quality Code 中，所连接的自动化系统必须支持 Quality Code。 在自动化系统中组态 PLC 变量时，需确保有足够的存储空间用于 Quality Code。 例如，在 S7 自动化系统中，Quality Code 需要将一个额外字节附加到过程值上。 为了防止错误，在组态变量时必须将该字节考虑进去，例如在数据块的末尾。

结构

Quality Code 具有如下二进制结构：

QQSSSSLL

Q： 质量

S： 质量的子状态

L： 限制。 该数值是可选的。

说明

显示在“质量”表格中的 Quality Code 是质量级的基本数值。 使用子状态和限制元素导致中间数值超过相关的质量级。

质量

前两位指定变量的质量。

	Q	Q	S	S	S	S	L	L
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Bad - The value is not useful	0	0	-	-	-	-	-	-
Uncertain - The quality of the value is less than normal, but the value may still be useful.	0	1	-	-	-	-	-	-

Good (Non-Cascade) - The quality of the value is good. Possible alarm conditions may be indicated by the sub-status.	1	0	-	-	-	-	-	-
Good (Cascade) - The value may be used in control.	1	1	-	-	-	-	-	-

子状态

仅仅使用质量是不够的。 单个的质量分为子状态。 Quality Code 采用二进制编码。 该值必须转换为十六进制格式，以便于分析 Quality Code。

变量的质量代码

下表列出了可能的 Quality Code。 在表的顶行是反映最差质量的 Quality Code；在表的底行则是反映最好质量的 Quality Code。 为代表最佳质量的 Quality Code 分配最低优先级，而为代表最差质量的 Quality Code 分配最高优先级。 如果此过程中的一个变量发生多种状态，则将传送最差的代码。

代码（十六进制）	质量		Q	Q	S	S	S	S	L	L
0x23	Bad	Device passivated - Diagnostic alerts inhibited	0	0	1	0	0	0	1	1
0x3F	Bad	Function check - Local override	0	0	1	1	1	1	1	1
0x1C	Bad	Out of Service - The value is not reliable because the block is not being evaluated, and may be under construction by a configurer. Set if the block mode is O/S.	0	0	0	1	1	1	-	-
0x73	Uncertain	Simulated value - Start	0	1	1	1	0	0	1	1
0x74	Uncertain	Simulated value - End	0	1	1	1	0	1	-	-
0x84	Good (Non-Cascade)	Active Update event - Set if the value is good and the block has an active Update event.	1	0	0	0	0	1	-	-
0x24	Bad	Maintenance alarm - More diagnostics available.	0	0	1	0	0	1	-	-
0x18	Bad	No Communication, with no usable value - Set if there has never been any communication with this value since it was last "Out of Service".	0	0	0	1	1	0	-	-

代码（十六进制）	质量		Q	Q	S	S	S	S	L	L
0x14	Bad	No Communication, with last usable value - Set if this value had been set by communication, which has now failed.	0	0	0	1	0	1	-	-
0x0C	Bad	Device Failure - Set if the source of the value is affected by a device failure.	0	0	0	0	1	1	-	-
0x10	Bad	Sensor failure	0	0	0	1	0	0	-	-
0x08	Bad	Not Connected - Set if this input is required to be connected and is not connected.	0	0	0	0	1	0	-	-
0x04	Bad	Configuration Error - Set if the value is not useful because there is some inconsistency regarding the parameterization or configuration, depending on what a specific manufacturer can detect.	0	0	0	0	0	1	-	-
0x00	Bad	non-specific - There is no specific reason why the value is bad. Used for propagation.	0	0	0	0	0	0	-	-
0x28	Bad	Process related - Substitute value	0	0	1	0	1	0	-	-
0x2B	Bad	Process related - No maintenance	0	0	1	0	1	0	1	1
0x68	Uncertain	Maintenance demanded	0	1	1	0	1	0	-	-
0x60	Uncertain	Simulated value - Set when the process value is written by the operator while the block is in manual mode.	0	1	1	0	0	0	-	-
0x64	Uncertain	Sensor calibration	0	1	1	0	0	1	-	-
0x5C	Uncertain	Configuration error	0	1	0	1	1	1	-	-
0x58	Uncertain	Sub-normal	0	1	0	1	1	0	-	-
0x54	Uncertain	Engineering Unit Range Violation - Set if the value lies outside of the set of values defined for this parameter. The Limits define which direction has been exceeded.	0	1	0	1	0	1	-	-
0x50	Uncertain	Sensor conversion not accurate	0	1	0	1	0	0	-	-

代码（十六进制）	质量		Q	Q	S	S	S	S	L	L
0x4B	Uncertain	Substitute (constant)	0	1	0	0	1	0	1	1
0x78	Uncertain	Process related - No maintenance	0	1	1	1	1	0	-	-
0x4C	Uncertain	Initial Value - Value of volatile parameters during and after reset of the device or of a parameter.	0	1	0	0	1	1	-	-
0x48	Uncertain	Substitute value - Predefined value is used instead of the calculated one. This is used for fail safe handling.	0	1	0	0	1	0	-	-
0x44	Uncertain	Last Usable Value - Whatever was writing this value has stopped doing so. This is used for fail safe handling.	0	1	0	0	0	1	-	-
0x40	Uncertain	Non-specific - There is no specific reason why the value is uncertain. Used for propagation.	0	1	0	0	0	0	-	-
0xE0	Good (Cascade)	Initiate Fail Safe (IFS) - The value is from a block that wants its downstream output block (e.g. AO) to go to Fail Safe.	1	1	1	0	0	0	-	-
0xD8	Good (Cascade)	Local Override (LO) - The value is from a block that has been locked out by a local key switch or is a Complex AO/DO with interlock logic active. The failure of normal control must be propagated to a function running in a host system for alarm and display purposes. This also implies "Not Invited".	1	1	0	1	1	0	-	-
0xD4	Good (Cascade)	Do Not Select (DNS) - The value is from a block which should not be selected, due to conditions in or above the block.	1	1	0	1	0	1	-	-
0xCC	Good (Cascade)	Not Invited (NI) - The value is from a block which does not have a target mode that would use this input.	1	1	0	0	1	1	-	-
0xC8	Good (Cascade)	Initialization Request (IR) - The value is an initialization value for a source (back calculation input parameter), because the lower loop is broken or the mode is wrong.	1	1	0	0	1	0	-	-

代码（十六进制）	质量		Q	Q	S	S	S	S	L	L
0xC4	Good (Cascade)	Initialization Acknowledge (IA) - The value is an initialized value from a source (cascade input, remote-cascade in, and remote-output in parameters).	1	1	0	0	0	1	-	-
0xC0	Good (Cascade)	OK - No error or special condition is associated with this value.	1	1	0	0	0	0	-	-
0xA0	Good (Non-Cascade)	Initiate fail safe	1	0	1	0	0	0	-	-
0x98	Good (Non-Cascade)	Unacknowledged Critical Alarm - Set if the value is good and the block has an unacknowledged Alarm with a priority greater than or equal to 8.	1	0	0	1	1	0	-	-
0x94	Good (Non-Cascade)	Unacknowledged Advisory Alarm - Set if the value is good and the block has an unacknowledged Alarm with a priority less than 8.	1	0	0	1	0	1	-	-
0x90	Good (Non-Cascade)	Unacknowledged Update event - Set if the value is good and the block has an unacknowledged Update event.	1	0	0	1	0	0	-	-
0x8C	Good (Non-Cascade)	Active Critical Alarm - Set if the value is good and the block has an active Alarm with a priority greater than or equal to 8.	1	0	0	0	1	1	-	-
0x88	Good (Non-Cascade)	Active Advisory Alarm - Set if the value is good and the block has an active Alarm with a priority less than 8.	1	0	0	0	1	0	-	-
0xA8	Good (Non-Cascade)	Maintenance demanded	1	0	1	0	1	0	-	-

代码（十六进制）	质量		Q	Q	S	S	S	S	L	L
0xA4	Good (Non-Cascade)	Maintenance required	1	0	1	0	0	1	-	-
0xBC	Good (Non-Cascade)	Function check - Local override	1	0	1	1	1	1	-	-
0x80	Good (Non-Cascade)	OK - No error or special condition is associated with this value.	1	0	0	0	0	0	-	-

限制

Quality Codes 可通过限制进一步细分。 限制是可选的。

	Q	Q	S	S	S	S	L	L
O.K. - The value is free to move.	-	-	-	-	-	-	0	0
Low limited - The value has acceded its low limits.	-	-	-	-	-	-	0	1
High limited - The value has acceded its high limits.	-	-	-	-	-	-	1	0
Constant (high and low limited) - The value cannot move, no matter what the process does.	-	-	-	-	-	-	1	1

利用 OPC 通讯的质量代码

若是通过"OPC"通信，则会转换 OPC 不支持的 Quality Codes。

WinCC 中的质量代码	OPC 中的质量代码
0x48	0x40
0x4C	0x40
0x5C	0x40
0x60	0x40

WinCC 中的质量代码	OPC 中的质量代码
0x80...0xD4	0xC0
0xD8	0xC0

参见

VariableStateType 属性 (页 1821)

SetTag (页 1706)

GetTag (页 1637)

3.2.1.4 VariableStateType 属性

说明

返回监视属性或事件动态变化的变量监控类型：无监控、质量代码或变量状态。读访问权。

索引	VariableStateType
0	hmiNoVariableState
1	hmiVariableQCState
2	hmiVariableState

示例

“GetVariableStateType()”程序读取当前文档的监控类型。在此示例中，监视类型将在消息中输出：

```
Sub GetVariableStateType()
'VBA819
Dim objDyn As HMIDynamicDialog
Set objDyn =
ActiveDocument.Properties("Width").CreateDynamic(hmiDynamicCreationTypeDynamicDialog,
"'TestVal'")
MsgBox objDyn.VariableStateType
objDyn.Delete
End Sub
```

参见

HMI 变量的质量代码 (页 1814)

结构概览 (页 1812)

常数 (页 1822)

转换例程（控制中心） (页 1831)

SetTag (页 1706)

GetTag (页 1637)

3.2.1.5 常数

常见定义

MAX_DM_OHIO_MACHINES	64	项目中的最大 PC 数
MAX_DM_OHIO_APPLICATIONS	32	最大本地客户端应用程序数量
MAX_DM_UPDATE_CYCLES	15	最大更新周期数量
MAX_DM_SYSTEM_CYCLES	10	在总更新周期中，包括 10 个系统周期
MAX_DM_USER_CYCLES	5	和 5 个用户周期
MAX_DM_VAR_NAME	128	变量名称的最大长度
MAX_DM_TYPE_NAME	128	数据类型名称的最大长度
MAX_DM_GROUP_NAME	64	变量组名称的最大长度
MAX_DM_CYCLE_NAME	64	周期时间文本的最大长度
MAX_DM_FORMAT_NAME	64	格式规范的最大长度
MAX_DM_SCALE_NAME	64	标定规范的最大长度
MAX_DM_SCALE_PARAM_NAME	64	标定参数描述的最大长度
MAX_DM_MEMBER_NAME	32	组合类型成员的最大长度
MAX_DM_INFOTEXT_LEN	255	信息文本的最大长度
MAX_DM_SHIFT_NAME_LEN	32	班次名称的最大长度
MAX_DM_SHIFTS	16	每天排班的最大次数
MAX_DM_SHIFT_HOLYDAYS	30	排班计划中的最大节假日天数
MAX_DM_SHIFT_HOLYNAME	64	节假日名称的最大长度
MAX_DM_SERVICE_NAME	32	服务名称的最大长度
MAX_DM_APP_NAME	32	逻辑应用程序名称的最大长度

MAX_DM_DSN_NAME	32	数据库数据源名称的最大长度
MAX_DM_UNIT_NAME	65	单位的最大长度
MAX_DM_CONNECTION_NAME	32	连接的最大长度
MAX_DM_VAR_SPECIFIC	25	GAPI 中变量特定部分的最大长度
MAX_DM_CON_SPECIFIC	128	GAPI 中连接特定部分的最大长度
MAX_DM_CON_COMMON	128	GAPI 中连接特定部分的最大长度
MAX_DM_VARTYPE_TEXT_LEN	255	文本变量的最大长度

变量类型

DM_VARTYPE_BIT	1	二进制变量
DM_VARTYPE_SBYTE	2	有符号 8 位数
DM_VARTYPE_BYTE	3	无符号 8 位数
DM_VARTYPE_SWORD	4	有符号 16 位数
DM_VARTYPE_WORD	5	无符号 16 位数
DM_VARTYPE_SDWORD	6	有符号 32 位数
DM_VARTYPE_DWORD	7	无符号 32 位数
DM_VARTYPE_FLOAT	8	32 位 IEEE 754 浮点数
DM_VARTYPE_DOUBLE	9	64 位 IEEE 754 浮点数
DM_VARTYPE_TEXT_8	10	文本变量，8 位字符集
DM_VARTYPE_TEXT_16	11	文本变量，16 位字符集
DM_VARTYPE_RAW	12	原始数据类型
DM_VARTYPE_STRUCT	14	结构变量
DM_VARTYPE_TEXTREF	18	文本库中的文本参考

属性标记

DM_SIMULATION_MODE	0x00000001	当前在进行变量仿真（无过程写访问权限）
DM_INTERNAL_VAR	0x00000002	内部变量
DM_EXTERNAL_VAR	0x00000004	外部变量

DM_HAS_MIN_LIMIT	0x00000008	变量有固定下限
DM_HAS_MAX_LIMIT	0x00000010	变量有固定上限
DM_HAS_DEFAULT_VALUE	0x00000020	变量有替换值
DM_HAS_STARTUP_VALUE	0x00000040	变量有初始值
DM_USE_DEFAULT_ON_STARTUP	0x00000080	在系统启动时输入替换值
DM_USE_DEFAULT_ON_MAX	0x00000100	违反上限时输入替换值
DM_USE_DEFAULT_ON_MIN	0x00000200	违反下限时输入替换值
DM_USE_DEFAULT_ON_COMM_ERROR	0x00000400	在出现连接错误时替换值
DM_WRITE_ACCESS_APPLICATION	0x00000800	应用程序可以写变量
DM_WRITE_ACCESS_PROCESS	0x00001000	过程可以写变量
DM_INTERNAL_VAR_LOKAL	0x00002000	内部变量，更新 1 = 本地计算机/0 = 项目范围
DM_INVISIBLE	0x00004000	请勿在控制中心界面上显示变量
DM_EXTERNAL_LOCK	0x00008000	从外部生成变量并且只能从外部删除该变量

记录标记

DM_NOTIFY_MAX_LIMIT	0x00000001	达到上限时记录条目
DM_NOTIFY_MIN_LIMIT	0x00000002	达到下限时记录条目
DM_NOTIFY_FORMAT_ERROR	0x00000004	出现转换错误时记录条目

DM_NOTIFY_ACCESS_FAULT	0x00000008	执行被禁止的写访问时记录条目
DM_NOTIFY_APPLICATION_WRITE	0x00000010	应用程序执行写访问时记录条目
DM_NOTIFY_PROCESS_WRITE	0x00000020	过程执行写访问时记录条目

状态标记

在运行系统中，可以监视各个 WinCC 变量的变量状态。变量状态还包括组态测量范围限制超限以及 WinCC 与自动化系统间的连接状态。

质量代码包括变量质量的相关信息，与在何处形成该代码无关。它将整个数值传送和数值处理过程的状态考虑在内。

例如，如果超出系统中测量范围的下限，则始终报告质量代码"0x55"。这种测量范围超限情况可能会在 WinCC 数据管理器或现场设备中发生。可使用变量状态来确定此类测量范围超限情况是发生在 WinCC 中还是发生在将数值传送到 WinCC 之前。

例如，如果变量状态通过显示代码 0x0010 报告超限，则表明是超过 WinCC 中组态的范围下限。如果变量状态没有报告超限，则表示传送到 WinCC 中的质量代码已经包含超限信息。

	0x0000	无错误
DM_VARSTATE_NOT_ESTABLISHED	0x0001	未与伙伴建立连接
DM_VARSTATE_HANDSHAKE_ERROR	0x0002	协议错误
DM_VARSTATE_HARDWARE_ERROR	0x0004	网络模块存在缺陷
DM_VARSTATE_MAX_LIMIT	0x0008	违反了组态的上限
DM_VARSTATE_MIN_LIMIT	0x0010	违反了组态的下限
DM_VARSTATE_MAX_RANGE	0x0020	超出格式限制
DM_VARSTATE_MIN_RANGE	0x0040	低于格式限制
DM_VARSTATE_CONVERSION_ERROR	0x0080	显示转换错误（与 DM_VARSTATE_..._RANGE 一起显示）
DM_VARSTATE_STARTUP_VALUE	0x0100	变量的初始值
DM_VARSTATE_DEFAULT_VALUE	0x0200	变量的替换值
DM_VARSTATE_ADDRESS_ERROR	0x0400	通道寻址错误
DM_VARSTATE_INVALID_KEY	0x0800	变量未找到/不可用
DM_VARSTATE_ACCESS_FAULT	0x1000	访问变量被拒绝

DM_VARSTATE_TIMEOUT	0x2000	超时/通道无反馈
DM_VARSTATE_SERVERDOWN	0x4000	服务器停机

通知类别

DM_NOTIFY_CLASS_ERROR	0x00000001	通知代码包含错误 ID
DM_NOTIFY_CLASS_WARNING	0x00000002	通知代码包含警告
DM_NOTIFY_CLASS_DATA	0x00000003	通知代码包含数据类型

通知代码

DM_NOTIFY_SHUTDOWN	0x00000001	错误等级：数据管理器正在关闭
DM_NOTIFY_PROCESSNET_ERROR	0x00000002	错误等级：过程总线出错
DM_NOTIFY_SYSNET_ERROR	0x00000003	错误等级：系统总线出错
DM_NOTIFY_QUEUE_50_PERCENT	0x00000001	警告等级：应用程序队列填充水平为 50%
DM_NOTIFY_QUEUE_60_PERCENT	0x00000002	警告等级：应用程序队列填充水平为 60%
DM_NOTIFY_QUEUE_70_PERCENT	0x00000003	警告等级：应用程序队列填充水平为 70%
DM_NOTIFY_QUEUE_80_PERCENT	0x00000004	警告等级：应用程序队列填充水平为 80%
DM_NOTIFY_QUEUE_90_PERCENT	0x00000005	警告等级：应用程序队列填充水平为 90%
DM_NOTIFY_QUEUE_OVERFLOW	0x00000006	警告等级：应用程序队列溢出
DM_NOTIFY_CYCLES_CHANGED	0x00000010	警告等级 => 再次读取循环

DM_NOTIFY_MACHINES_CHANGED	0x00000001	警告等级：计算机列表
DM_NOTIFY_PROJECT_OPENED	0x00000002	警告等级：正在加载项目
DM_NOTIFY_PROJECT_CLOSED	0x00000003	警告等级：正在关闭项目
DM_NOTIFY_SYSTEM_LOCALE	0x00000004	切换语言（资源）
DM_NOTIFY_DATA_LOCALE	0x00000005	切换语言（组态数据）
DM_NOTIFY_PROJECT_RUNTIME	0x00000006	已在运行时模式下激活项目
DM_NOTIFY_PROJECT_EDIT	0x00000007	已取消激活项目的运行时模式
DM_NOTIFY_HOTKEY_CHANGE	0x00000008	热键已更改
DM_NOTIFY_APPLICATION_DATA	0x00000001	数据等级：已通过应用程序发送数据
DM_NOTIFY_VARIABLE_DATA	0x00000002	数据等级：变量数据
DM_NOTIFY_SERVERDOWN	0x00000022	服务器停机
DM_NOTIFY_SERVERUP	0x00000023	服务器运行准备就绪

计算机类型

计算机类型	0	服务器
	1	客户端

已注册消息

DM_OHIOLANGUAGE“WM_OHIOLANGUAGE”

系统内部变量

DM_VAR_CURRENT_LANGUAGE“OHIOCurrentLanguage”

参见

VariableStateType 属性 (页 1821)

3.2.1.6 错误消息

概述

通过 API 函数的 CMN_ERROR 错误结构可返回以下错误消息：

DM_E_SYS_ERROR	0x10000000	如果在错误结构 CMN_ERROR 的 dwError1 中设置该位，则 dwError2 包含系统错误代码。
DM_E_OK	0x00000000	无错误
DM_E_CANCEL	0x00000001	用户已在对话框中选择“取消”(Cancel)
DM_E_FILE	0x00000002	在文件操作过程中出错
DM_E_UPDATE	0x00000003	更新项目
DM_E_NO_RT_PRJ	0x00000004	不存在处于运行时模式的项目
DM_E_NOT_SUPPORTED	0x00000005	请求的服务不可用
DM_E_ALREADY_CONNECTED	0x00000006	已建立与数据管理器的连接
DM_E_NOT_CONNECTED	0x00000007	未连接到数据管理器
DM_E_INVALID_TAID	0x00000008	无效事务 ID
DM_E_INVALID_KEY	0x00000009	未找到变量

DM_E_INVALID_TYPE	0x0000000A	无效变量类型
DM_E_MAX_LIMIT	0x0000000B	违反了变量上限
DM_E_MIN_LIMIT	0x0000000C	违反了变量下限
DM_E_MAX_RANGE	0x0000000D	违反了格式转换上限
DM_E_MIN_RANGE	0x0000000E	违反了格式转换下限
DM_E_ACCESS_FAULT	0x0000000F	已禁止对变量进行写访问
DM_E_TIMEOUT	0x00000010	超时错误
DM_E_ALREADY_EXIST	0x00000011	已存在要创建的对象
DM_E_PARAM	0x00000012	无效参数
DM_E_INV_PRJ	0x00000013	无法找到/加载指定的项目
DM_E_UNKNOWN	0x00000014	未知错误
DM_E_OOM	0x00000015	可用内存不足
DM_E_NOT_CREATED	0x00000016	无法创建项目
DM_E_MACHINE_NOT_FOUND	0x00000017	未找到计算机
DM_E_NO_INFO_FOUND	0x00000018	无法找到启动信息
DM_E_INTERNAL	0x00000019	内部处理错误
DM_E_INVALID_LOCALE	0x0000001A	本地 ID 错误

DM_E_COMMUNICATION	0x0000001B	本地 ID 错误
DM_E_DONT_EXIST	0x0000001C	对象不存在 例如：已尝试创建一个变量，但指定的连接不存在
DM_E_ALREADY_ACTIVATED	0x0000001D	项目正处于运行时模式 例如：已尝试激活处于运行时模式的项目。
DM_E_NO_OPEN_PROJECT	0x0000001E	未打开任何项目。
DM_E_ALREADY_DEACTIVATED	0x0000001F	项目未处于运行时模式
DM_E_NO_RIGHTS	0x00000020	传送了错误的 CreatorID
DM_E_NOT_DELETED	0x00000021	无法删除对象
DM_E_LICENSE	0x00000022	软件保护：无许可证
DM_E_LICENSE_LIMIT	0x00000023	软件保护：已达到/超出限制
DM_E_INVALID_OBJECTTYPE	0x00000024	无效对象类型
DM_E_OP_REQUIERES_PRJEDITMODE	0x00000025	项目正处于运行时模式 例如：操作要求该项目未被激活。
DM_E_INTERFACE	0x00000026	访问接口时出现内部错误
DM_E_UNIT_NOT_FOUND	0x00000027	未找到单位
DM_E_CONNECTION_NOT_FOUND	0x00000028	未找到连接

参见

CMN_ERROR (页 2919)

3.2.1.7 转换例程（控制中心）

概述

编号	“有符号 8 位数”数据类型的转换例程
0	CharToSignedByte
1	CharToUnsignedByte
2	CharToUnsignedWord
3	CharToUnsignedDword
4	CharToSignedWord
5	CharToSignedDword
6	CharToMSBByte
7	CharToMSBWord
8	CharToMSBDword
9	CharToBCDByte
10	CharToBCDWord
11	CharToBCDDword
12	CharToSignedBCDByte
13	CharToSignedBCDWord
14	CharToSignedBCDDword
15	CharToExtSignedBCDByte
16	CharToExtSignedBCDWord
17	CharToExtSignedBCDDword
18	CharToAikenByte
19	CharToAikenWord
20	CharToAikenDword
21	CharToSignedAikenByte
22	CharToSignedAikenWord
23	CharToSignedAikenDword
24	CharToExcessByte
25	CharToExcessWord
26	CharToExcessDword
27	CharToSignedExcessByte

28	CharToSignedExcessWord
29	CharToSignedExcessDword
> 29	(CharToSignedByte)

编号	“无符号 8 位数”数据类型的转换例程
0	ByteToUnsignedByte
1	ByteToUnsignedWord
2	ByteToUnsignedDword
3	ByteToSignedByte
4	ByteToSignedWord
5	ByteToSignedDword
6	ByteToBCDByte
7	ByteToBCDWord
8	ByteToBCDDword
9	ByteToAikenByte
10	ByteToAikenWord
11	ByteToAikenDword
12	ByteToExcessByte
13	ByteToExcessWord
14	ByteToExcessDword
> 14	(ByteToUnsignedByte)

编号	“有符号 16 位数”数据类型的转换例程
0	ShortToSignedWord
1	ShortToUnsignedByte
2	ShortToUnsignedWord
3	ShortToUnsignedDword
4	ShortToSignedByte
5	ShortToSignedDword
6	ShortToMSBByte
7	ShortToMSBWord

8	ShortToMSBDword
9	ShortToBCDByte
10	ShortToBCDWord
11	ShortToBCDDword
12	ShortToSignedBCDByte
13	ShortToSignedBCDWord
14	ShortToSignedBCDDword
15	ShortToExtSignedBCDByte
16	ShortToExtSignedBCDWord
17	ShortToExtSignedBCDDword
18	ShortToAikenByte
19	ShortToAikenWord
20	ShortToAikenDword
21	ShortToSignedAikenByte
22	ShortToSignedAikenWord
23	ShortToSignedAikenDword
24	ShortToExcessByte
25	ShortToExcessWord
26	ShortToExcessDword
27	ShortToSignedExcessByte
28	ShortToSignedExcessWord
29	ShortToSignedExcessDword
> 29	(ShortToSignedWord)

编号	**“无符号 16 位数”数据类型的转换例程**
0	WordToUnsignedWord
1	WordToUnsignedByte
2	WordToUnsignedDword
3	WordToSignedByte
4	WordToSignedWord
5	WordToSignedDword
6	WordToBCDByte

7	WordToBCDWord
8	WordToBCDDword
9	WordToAikenByte
10	WordToAikenWord
11	WordToAikenDword
12	WordToExcessByte
13	WordToExcessWord
14	WordToExcessDword
15	WordToSimaticBCDCounter
16	WordToSimaticCounter
> 16	(WordToUnsignedWord)

编号	“有符号 32 位数”数据类型的转换例程
0	LongToSignedDword
1	LongToUnsignedByte
2	LongToUnsignedWord
3	LongToUnsignedDword
4	LongToSignedByte
5	LongToSignedWord
6	LongToMSBByte
7	LongToMSBWord
8	LongToMSBDword
9	LongToBCDByte
10	LongToBCDWord
11	LongToBCDDword
12	LongToSignedBCDByte
13	LongToSignedBCDWord
14	LongToSignedBCDDword
15	LongToExtSignedBCDByte
16	LongToExtSignedBCDWord
17	LongToExtSignedBCDDword

18	LongToAikenByte
19	LongToAikenWord
20	LongToAikenDword
21	LongToSignedAikenByte
22	LongToSignedAikenWord
23	LongToSignedAikenDword
24	LongToExcessByte
25	LongToExcessWord
26	LongToExcessDword
27	LongToSignedExcessByte
28	LongToSignedExcessWord
29	LongToSignedExcessDword
30	LongToSimaticBCDTimer
31	(LongToSignedDword)
32	(LongToSignedDword)
33	(LongToSignedDword)
34	LongToSimaticTimer
> 34	(LongToSignedDword)

编号	**“无符号 32 位数”数据类型的转换例程**
0	DwordToUnsignedDword
1	DwordToUnsignedByte
2	DwordToUnsignedWord
3	DwordToSignedByte
4	DwordToSignedWord
5	DwordToSignedDword
6	DwordToBCDByte
7	DwordToBCDWord
8	DwordToBCDDword
9	DwordToAikenByte
10	DwordToAikenWord

11	DwordToAikenDword
12	DwordToExcessByte
13	DwordToExcessWord
14	DwordToExcessDword
15	DwordToSimaticBCDTimer
16	(DwordToUnsignedDword)
17	(DwordToUnsignedDword)
18	(DwordToUnsignedDword)
19	DwordToSimaticTimer
> 19	(DwordToUnsignedDword)

编号	“32 位 IEEE 754 浮点数”数据类型的转换例程
0	FloatToFloat
1	FloatToUnsignedByte
2	FloatToUnsignedWord
3	FloatToUnsignedDword
4	FloatToSignedByte
5	FloatToSignedWord
6	FloatToSignedDword
7	FloatToDouble
8	FloatToMSBByte
9	FloatToMSBWord
10	FloatToMSBDword
11	FloatToBCDByte
12	FloatToBCDWord
13	FloatToBCDDword
14	FloatToSignedBCDByte
15	FloatToSignedBCDWord
16	FloatToSignedBCDDword
17	FloatToExtSignedBCDByte

18	FloatToExtSignedBCDWord
19	FloatToExtSignedBCDDword
20	FloatToAikenByte
21	FloatToAikenWord
22	FloatToAikenDword
23	FloatToSignedAikenByte
24	FloatToSignedAikenWord
25	FloatToSignedAikenDword
26	FloatToExcessByte
27	FloatToExcessWord
28	FloatToExcessDword
29	FloatToSignedExcessByte
30	FloatToSignedExcessWord
31	FloatToSignedExcessDword
32	FloatToSimaticBCDTimer
33	(FloatToFloat)
34	(FloatToFloat)
35	(FloatToFloat)
36	FloatToS5Float
37	FloatToSimaticTimer
> 37	(FloatToFloat)

编号	**“64 位 IEEE 754 浮点数”数据类型的转换例程**
0	DoubleToDouble
1	DoubleToUnsignedByte
2	DoubleToUnsignedWord
3	DoubleToUnsignedDword
4	DoubleToSignedByte
5	DoubleToSignedWord
6	DoubleToSignedDword

7	DoubleToFloat
8	DoubleToMSBByte
9	DoubleToMSBWord
10	DoubleToMSBDword
11	DoubleToBCDByte
12	DoubleToBCDWord
13	DoubleToBCDDword
14	DoubleToSignedBCDByte
15	DoubleToSignedBCDWord
16	DoubleToSignedBCDDword
17	DoubleToExtSignedBCDByte
18	DoubleToExtSignedBCDWord
19	DoubleToExtSignedBCDDword
20	DoubleToAikenByte
21	DoubleToAikenWord
22	DoubleToAikenDword
23	DoubleToSignedAikenByte
24	DoubleToSignedAikenWord
25	DoubleToSignedAikenDword
26	DoubleToExcessByte
27	DoubleToExcessWord
28	DoubleToExcessDword
29	DoubleToSignedExcessByte
30	DoubleToSignedExcessWord
31	DoubleToSignedExcessDword
32	DoubleToSimaticBCDTimer
33	(DoubleToDouble)
34	(DoubleToDouble)
35	(DoubleToDouble)
36	DoubleToS5Float
37	DoubleToSimaticTimer
> 37	(DoubleToDouble)

参见

VariableStateType 属性 (页 1821)

函数概览 (页 1813)

3.2.2 结构

3.2.2.1 DM_CONNECTION_DATA

声明

```
typedef struct {
    CHAR          szConnection[
                 MAX_DM_CONNECTION_NAME +3];
    CHAR          szUnitName[MAX_DM_UNIT_NAME +1];
    CHAR          szCommon[MAX_DM_CON_COMMON +1];
    CHAR          szSpecific[MAX_DM_CON_SPECIFIC +1];
    DWORD         dwVarNum;
    }
    DM_CONNECTION_DATA;
```

成员

szConnection

逻辑连接的名称

szUnitName

通道单元的名称

szCommon

该参数预留给将来开发使用。

szSpecific

szSpecific 包含连接的地址参数，例如：以太网地址，插槽号....有关 PLC 的特定详细信息，请参见通信手册。

该值与 WinCC 中变量属性的参数列中显示的值相同。

dwVarNum

已分配变量的数目

所需文件

dmclient.h

API 函数

DM_ENUM_CONNECTION_PROC (页 2091)	列出连接数据（回调）

参见

DM_ENUM_CONNECTION_PROC (页 2091)

3.2.2.2 DM_CONNKEY

声明

```
typedef struct {
    CHAR      szName[ MAX_DM_CONNECTION_NAME + 1 ];
    LPVOID    lpvUserData;
    }
    DM_CONNKEY;
```

成员

szName

逻辑连接的名称

lpvUserData

特定应用数据的指针

所需文件

dmclient.h

API 函数

DMEnumConnectionData (页 2087)	列出连接数据

参见

DMEnumConnectionData (页 2087)

3.2.2.3 DM_CYCLE_INFO

声明

```
typedef struct {
    DWORD    dwCycleTime;
    DWORD    dwCycleIndex;
    char     szDescription[ MAX_DM_CYCLE_NAME + 1 ];
    }
    DM_CYCLE_INFO;
```

成员

dwCycleTime

更新周期的时基

dwCycleIndex

标识更新周期列表内的顺序。

“改变时”	索引：0
“250ms”	索引：1
:	:
“用户周期 5”	索引：15

szDescription

更新周期的描述。

所需文件

dmclient.h

API 函数

DM_ENUM_CYCLES_PROC (页 1923)	列出更新周期（回调）

参见

DM_ENUM_CYCLES_PROC (页 1923)

3.2.2.4 DM_DATA_SERVICE

声明

```
typedef struct {
    DWORD    dwTeleType;
    char     szService[MAX_DM_SERVICE_NAME + 1];
    char     szSendingApp[MAX_DM_APP_NAME + 1];
    DWORD    dwSendingMachine;
    DWORD    dwDataSize;
    BYTE     byData[1];
    }
    DM_DATA_SERVICE;
```

成员

dwTeleType

该参数为将来开发预留，必须预设为 0。

szService

数据传输通道的名称。该名称与安装服务（DMInstallDataService）期间分配的名称一致。

szSendingApp

发送方的逻辑名称。该名称与为 DMConnect 指定的应用程序名称一致。

dwSendingMachine

发送数据包的计算机的索引 (0 - .63)。注册在计算机列表中的所有计算机都可以通过 API 函数的相应索引进行访问。计算机列表中第一个条目的索引为“0”。

dwDataSize

数据包的大小（字节）：Data[0] ... byData[dwDataSize - 1]

byData

数据指针。

所需文件

dmclient.h

API 函数

DM_DATA_SERVICE_PROC (页 1954)	列出数据传输通道（回调），安装数据传输通道（回调）

参见

DM_DATA_SERVICE_PROC (页 1954)

3.2.2.5 DM_DIRECTORY_INFO

声明

```
typedef struct {
    char    szProjectDir[_MAX_PATH + 1 ];
    char    szProjectAppDir[_MAX_PATH + 1 ];
    char    szGlobalLibDir[_MAX_PATH + 1 ];
    char    szProjectLibDir[_MAX_PATH + 1 ];
    char    szLokalProjectAppDir[_MAX_PATH + 1 ];
    }
    DM_DIRECTORY_INFO;
```

成员

szProjectDir

项目目录的完整路径，例如：D:\WinCC\Projekt1

szProjectAppDir

项目目录中应用程序子目录的完整路径，例如：D:\WinCC\Projekt1\GraCS

szGlobalLibDir

交叉项目库目录的完整路径，例如：D:\WinCC\aplib

szProjectLibDir

基于项目的库目录的完整路径，例如：D:\WinCC\Projekt1\Library

szLokalProjectAppDir

本地计算机上项目目录中应用程序子目录的完整路径。

所需文件

dmclient.h

API 函数

DMGetProjectDirectory (页 1941)	获取组态数据的路径和文件名

参见

DMGetProjectDirectory (页 1941)

3.2.2.6 DM_DLGOPTIONS

声明

```
typedef struct {
    DWORD                       dwFlags;
    LPRECT                      lprcPreference;
    DM_TEST_DROP_TARGET_PROC    lpfnTestDropTarget;
    DM_DROP_TARGET_PROC         lpfnDropTarget;
    LPVOID                      lpvUser;
    }
    DM_DLGOPTIONS;
```

成员

dwFlags

dwFlags 用于指定对话或对话框的反应：

DM_DLG_NOT_MODAL	该变量对话框不是模态的。
DM_DLG_NOT_MOVEABLE	该对话框无法移动。
DM_DLG_DRAGDROP	可通过拖放操作编辑对话框中的对象。

lprcPreference

RECT 类型结构的指针，包含对话框大小的相关信息。如果 lprcPreference == NULL，则对话框以预定义大小居中显示。

lpfnTestDropTarget

该参数为将来开发预留，必须预设为 NULL。

lpfnDropTarget

该参数为将来开发预留，必须预设为 NULL。

lpvUser

指向应用程序特定数据的指针。

所需文件

dmclient.h

API 函数

DMShowVarDatabase (页 2052)	打开变量选择对话框
DMShowVarDatabaseMulti (页 2057)	打开变量选择对话框

参见

DMShowVarDatabase (页 2052)

DMShowVarDatabaseMulti (页 2057)

3.2.2.7 DM_FORMAT_INFO

声明

```
typedef struct {
    DWORD    dwID;
    char     szName[MAX_DM_FORMAT_NAME + 1];
    }
    DM_FORMAT_INFO;
```

成员

dwID

要使用的转换例程的数量。更多详细信息，请参见“转换例程”部分。

szName

要使用的转换例程的名称。更多详细信息，请参见“转换例程”部分。

所需文件

dmclient.h

API 函数

DM_ENUM_FORMATS_PROC (页 1919)	列出格式转换（回调）

参见

DM_ENUM_FORMATS_PROC (页 1919)

3.2.2.8 DM_MACHINE_TABLE

声明

```
typedef struct {
    LONG                    nNumMachines;
    LONG                    nLocalMachine;
    DM_SD_TARGET_MACHINE    tm[MAX_DM_OHIO_MACHINES];
    }
    DM_MACHINE_TABLE;
```

成员

nNumMachines

项目中的计算机数（不可超出最大数量 MAX_DM_OHIO_MACHINES）。

nLocalMachine

计算机列表中本地计算机条目的索引。

注册在计算机列表中的所有计算机都可以通过 API 函数的相应索引进行访问。计算机列表中第一个条目的索引为“0”。

tm

该项目中包含的具有相关计算机信息的完全 DM_SD_TARGET_MACHINE (页 1851) 类型数组。但此处只传递这些结构的 nNumMachines 个；因此，最大可用索引 = (nMumMachines - 1)。

所需文件

dmclient.h

API 函数

DMGetMachineTable (页 1934)	查询计算机列表

参见

DM_SD_TARGET_MACHINE (页 1851)

DMGetMachineTable (页 1934)

3.2.2.9 DM_PROJECT_INFO

声明

```
typedef struct {
    char     szProjectFile[ _MAX_PATH + 1 ];
    char     szDSNName[ MAX_DM_DSN_NAME + 1 ];
    DWORD    dwDataLocale;
    }
    DM_PROJECT_INFO;
```

成员

szProjectFile

项目的文件名，包括路径和扩展名。

szDSNName

数据库的数据源名称

dwDataLocale

组态过程中使用的语言代码。

所需文件

dmclient.h

API 函数

DM_ENUM_OPENED_PROJECTS_PROC (页 1940)	列出项目（打开）（回调）
DMEnumOpenedProjects (页 1938)	获取项目信息

参见

DM_ENUM_OPENED_PROJECTS_PROC (页 1940)

DMEnumOpenedProjects (页 1938)

DMGetProjectInformation (页 1943)

3.2.2.10 DM_SEND_DATA_STRUCT

声明

```
typedef struct {
    BOOL                    fHighPriority;
    char                    szService[MAX_DM_SERVICE_NAME + 1];
    DWORD                   dwTargetMachineFlags;
    DWORD                   dwTargetMachines;
    DM_SD_TARGET_MACHINE    dmTargetMachine[
                            MAX_DM_OHIO_MACHINES];
    DWORD                   dwTargetApps;
    DM_SD_TARGET_APP        dmTargetApp[
                            MAX_DM_OHIO_APPLICATIONS];
    DWORD                   dwDataSize;
    BYTE                    byData[1];
    }
    DM_SEND_DATA_STRUCT;
```

成员

fHighPriority

fHighPriority 指示数据通信的优先级：

1	高优先级
0	普通优先级

szService

要使用的服务的名称。为使应用程序接收数据，不仅必须满足 dwTargetMachineFlags 的条件，还必须通过 DMInstallDataService 安装相应的服务。

dwTargetMachineFlags

dwTargetMachineFlags 指示应使用 DMSendApplicationData 函数将数据发送到的应用程序。

DM_SD_LOCAL	仅发送到本地应用程序（排除所有其它标记！） 当前不允许使用所有其它标记，并且这些标记预留给将来开发使用！
DM_SD_ALL_MACHINES	发送到项目中的所有计算机。 （该参数预留给将来开发使用）
DM_SD_ALL_SERVERS	发送到项目中的所有服务器。 （该参数预留给将来开发使用）
DM_SD_ALL_CLIENTS	发送到项目中的所有客户端。 （该参数预留给将来开发使用）
DM_SD_RELATED_MACHINES	发送到与本地主机隶属于同一台服务器的所有客户端 （该参数预留给将来开发使用）
DM_SD_FIRST_SERVER	（该参数预留给将来开发使用）
DM_SD_PRIMARY_SERVER	（该参数预留给将来开发使用）
DM_SD_EXCEPT_LOCAL	发送到满足地址描述的所有计算机，但本地计算机除外。 （该参数预留给将来开发使用）

dwTargetMachines

完整结构 dmTargetMachine 的数目。

该参数预留给将来开发使用，必须用 0 填充。

dmTargetMachine

指向用于指定数据将要发送到的计算机的 DM_SD_TARGET_MACHINE (页 1851) 结构的指针。此处仅 szMachineName 参数相关。

该参数预留给将来开发使用，应使用 0L 进行完全初始化。

dwTargetApps

完整结构 dmTargetApp 的数目。

dmTargetApp

DM_SD_TARGET_APP (页 1852) 类型结构包含数据将被发送到的应用程序的名称。

dwDataSize

要发送的数据量（字节）

byData

包含要发送数据的数组

备注

DMSendApplicationData 函数仅在本地实现。
出现在此结构中涉及远程访问的成员预留给将来开发使用。
dwTargetMachineFlags 只能使用 DM_SD_LOCAL 进行填充，指定任何其它值都将导致错误，并且 dwTargetMachines 必须为 0L。

所需文件

dmclient.h

API 函数

DMSendApplicationData (页 1958)	与应用程序进行数据通信

参见

DMSendApplicationData (页 1958)

DM_SD_TARGET_MACHINE (页 1851)

DM_SD_TARGET_APP (页 1852)

3.2.2.11 DM_SD_TARGET_MACHINE

声明

```
typedef struct {
    BOOL    fServer;
    BOOL    fLocal;
    char    szMachineName[
            MAX_COMPUTERNAME_LENGTH + 1];
    }
    DM_SD_TARGET_MACHINE;
```

成员

fServer

指示当前计算机是服务器还是客户端。

该参数与数据通信 (DM_SEND_DATA_STRUCT) 目的无关。

fLocal

指示查询应用程序当前是本地计算机上还是在网络中组态的其它计算机上运行。

该参数与数据通信 (DM_SEND_DATA_STRUCT) 目的无关。

szMachineName

计算机名称

备注

DM_SD_TARGET_MACHINE 是 DM_MACHINE_TABLE (页 1846) 和 DM_SEND_DATA_STRUCT (页 1848) 结构的一部分。

所需文件

dmclient.h

参见

DM_MACHINE_TABLE (页 1846)

DM_SEND_DATA_STRUCT (页 1848)

3.2.2.12 DM_SD_TARGET_APP

声明

```
typedef struct {
    char    szAppName[MAX_DM_APP_NAME + 1];
    }
    DM_SD_TARGET_APP;
```

成员

szAppName

对于 szAppName，请使用调用 DMConnect 时采用的应用程序的名称。

备注

DM_SD_TARGET_APP 是 DM_SEND_DATA_STRUCT (页 1848) 结构的一部分。

所需文件

dmclient.h

参见

DM_SEND_DATA_STRUCT (页 1848)

3.2.2.13 DM_TYPEREF

声明

```
typedef struct {
    DWORD    dwType;
    DWORD    dwSize;
    char     szTypeName[MAX_DM_TYPE_NAME + 1];
    }
    DM_TYPEREF;
```

成员

dwType

在 dwType 中指定变量类型：

DM_VARTYPE_BIT	二进制变量
DM_VARTYPE_SBYTE	有符号 8 位数
DM_VARTYPE_BYTE	无符号 8 位数
DM_VARTYPE_SWORD	有符号 16 位数
DM_VARTYPE_WORD	无符号 16 位数
DM_VARTYPE_SDWORD	有符号 32 位数
DM_VARTYPE_DWORD	无符号 32 位数
DM_VARTYPE_FLOAT	32 位 IEEE 754 浮点数
DM_VARTYPE_DOUBLE	64 位 IEEE 754 浮点数
DM_VARTYPE_TEXT_8	文本变量，8 位字符集
DM_VARTYPE_TEXT_16	文本变量，16 位字符集
DM_VARTYPE_RAW	原始数据类型
DM_VARTYPE_STRUCT	结构变量
DM_VARTYPE_TEXTREF	文本库中的文本参考
值 >=1024	用户自定义结构类型 在某些调用中需要此类型 ID，例如，过滤器用它来识别特定的结构类型。

dwSize

dwSize 指定 OS 上数据类型的长度（以字节为单位）。

szTypeName

结构变量的结构类型的名称位于 szTypeName 中。

备注

DM_TYPEREF 用于 DM_VAR_UPDATE_STRUCT (页 1854) 和 DM_VARIABLE_DATA (页 1863) 结构。

所需文件

dmclient.h

API 函数

DMGetVarType (页 2017)	获取变量的数据类型

参见

DM_VAR_UPDATE_STRUCT (页 1854)

DMGetVarType (页 2017)

DM_VARIABLE_DATA (页 1863)

DM_VAR_UPDATE_STRUCTEX (页 1856)

DM_VARIABLE_DATA4 (页 1866)

3.2.2.14 DM_VAR_UPDATE_STRUCT

声明

```
typedef struct {
    DM_TYPEREF    dmTypeRef;
    DM_VARKEY     dmVarKey;
    VARIANT       dmValue;
    DWORD         dwState;
    }
    DM_VAR_UPDATE_STRUCT;
```

成员

dmTypeRef

DM_TYPEREF (页 1852) 结构包含变量类型的相关信息。

例外情况：基于性能的考虑，若是循环请求，则并不会向该结构输入任何内容。

dmVarKey

要处理的变量通过 DM_VARKEY (页 1869) 结构进行指定。

dmValue

变量值。

说明

dmValue VARIANT 必须在首次使用前用 VariantInit(&dmValue) 进行初始化，并在使用后用 VariantClear(&dmValue) 进行释放。
只有包含若干 DM_VAR_UPDATE_STRUCT 的数组能够在首次使用前用 ZeroMemory() 或 memset() 进行预初始化。这会影响此处包含的 VARIANT dmValue（如 VariantInit(&dmValue)），因为 0 相当于 VT_EMPTY。

DM_VAR_UPDATE_STRUCT 结构一旦使用后便不可使用 ZeroMemory() 或 memset() 清除。VARIANT 随即包含可能为 VT_BSTR 类型的数据。

变量类型为 VT_BSTR 时，如果在使用 delete[] 删除已分配的 DM_VAR_UPDATE_STRUCT 结构之前未执行 VariantClear(&dmValue)，则可能出现内存管理问题（内存泄漏）。

	(0x0000)	无错误
DM_VARSTATE_NOT_ESTABLISHED	(0x0001)	未与伙伴建立连接
DM_VARSTATE_HANDSHAKE_ERROR	(0x0002)	协议错误
DM_VARSTATE_HARDWARE_ERROR	(0x0004)	网络模块故障
DM_VARSTATE_MAX_LIMIT	(0x0008)	违反了组态的上限
DM_VARSTATE_MIN_LIMIT	(0x0010)	违反了组态的下限
DM_VARSTATE_MAX_RANGE	(0x0020)	超出格式上限
DM_VARSTATE_MIN_RANGE	(0x0040)	超出格式下限
DM_VARSTATE_CONVERSION_ERROR	(0x0080)	显示转换错误（与 DM_VARSTATE_..._RANGE 一起显示）
DM_VARSTATE_STARTUP_VALUE	(0x0100)	变量的初始化值
DM_VARSTATE_DEFAULT_VALUE	(0x0200)	变量的替换值
DM_VARSTATE_ADDRESS_ERROR	(0x0400)	通道寻址错误
DM_VARSTATE_INVALID_KEY	(0x0800)	变量未找到/不可用
DM_VARSTATE_ACCESS_FAULT	(0x1000)	不允许访问变量
DM_VARSTATE_TIMEOUT	(0x2000)	超时/通道无反馈
DM_VARSTATE_SERVERDOWN	(0x4000)	服务器停机

所需文件

dmclient.h

API 函数

DM_NOTIFY_VARIABLE_PROC (页 2107)	获取变量值（回调）
DMGetValue (页 1977)	获取变量值

参见

DM_NOTIFY_VARIABLE_PROC (页 2107)

DM_VARKEY (页 1869)

DMGetValue (页 1977)

DM_TYPEREF (页 1852)

3.2.2.15 DM_VAR_UPDATE_STRUCTEX

声明

```
typedef struct {
    DM_TYPEREF    dmTypeRef;
    DM_VARKEY     dmVarKey;
    VARIANT       dmValue;
    DWORD         dwState;
    DWORD         dwQualityCode;
    }
    DM_VAR_UPDATE_STRUCTEX;
```

成员

dmTypeRef

DM_TYPEREF (页 1852) 结构包含变量类型的相关信息。

例外情况：基于性能的考虑，若是循环请求，则并不会向该结构输入任何内容。

dmVarKey

要处理的变量通过 DM_VARKEY (页 1869) 结构进行指定。

dmValue

变量值。

说明

VARIANT dmValue 必须在首次使用前用 VariantInit(&dmValue) 进行初始化，并在使用后用 VariantClear(&dmValue) 进行释放。
只有包含若干 DM_VAR_UPDATE_STRUCT 的数组能够在首次使用前用 ZeroMemory() 或 memset() 进行预初始化。这会影响此处包含的 VARIANT dmValue（如 VariantInit(&dmValue)），因为 0 相当于 VT_EMPTY。

DM_VAR_UPDATE_STRUCTEX 结构一旦使用后便不可使用 ZeroMemory() 或 memset() 清除。VARIANT 随即包含可能为 VT_BSTR 类型的数据。

变量类型为 VT_BSTR 时，如果在使用 delete[] 删除已分配的 DM_VAR_UPDATE_STRUCTEX 结构之前未执行 VariantClear(&dmValue)，则可能出现内存管理问题（内存泄漏）。

dwState

指示是否成功更改了变量值或是否出现了错误：

DM_VARSTATE_NOT_ESTABLISHED	0x0001	未与伙伴建立连接
DM_VARSTATE_HANDSHAKE_ERROR	0x0002	协议错误
DM_VARSTATE_HARDWARE_ERROR	0x0004	网络模块故障
DM_VARSTATE_MAX_LIMIT	0x0008	违反了组态的上限
DM_VARSTATE_MIN_LIMIT	0x0010	违反了组态的下限
DM_VARSTATE_MAX_RANGE	0x0020	超出格式上限
DM_VARSTATE_MIN_RANGE	0x0040	超出格式下限
DM_VARSTATE_CONVERSION_ERROR	0x0080	显示转换错误（与 DM_VARSTATE_..._RANGE 一起显示）
DM_VARSTATE_STARTUP_VALUE	0x0100	变量的初始化值
DM_VARSTATE_DEFAULT_VALUE	0x0200	变量的替换值
DM_VARSTATE_ADDRESS_ERROR	0x0400	通道寻址错误
DM_VARSTATE_INVALID_KEY	0x0800	变量未找到/不可用
DM_VARSTATE_ACCESS_FAULT	0x1000	不允许访问变量
DM_VARSTATE_TIMEOUT	0x2000	超时/通道无反馈
DM_VARSTATE_SERVERDOWN	0x4000	服务器停机

dwQualityCode

变量值的质量代码。

质量代码的组成方式如下：

	WinCC DM 变量状态	符合 Profibus PA/OPC 的质量代码
DM_VARSTATE_SERVERDOWN (0x4000)	服务器停机	Bad, out of service, 0x1C
DM_VARSTATE_HARDWARE_ERROR (0x0004)	网络模块故障	Bad, device failure, 0x0C
DM_VARSTATE_NOT_ESTABLISHED (0x0001)	未与伙伴建立连接	Bad, not connected, 0x08
DM_VARSTATE_TIMEOUT (0x2000)	超时/通道无反馈	Uncertain, last usable value, 0x44
DM_VARSTATE_HANDSHAKE_ERROR (0x0002)	协议错误	Bad, no communication (no usable value), 0x18
DM_VARSTATE_ADDRESS_ERROR (0x0400)	通道寻址错误	Bad, configuration error, value not accepted, 0x04
DM_VARSTATE_INVALID_KEY (0x0800)	变量未找到/不可用	Bad, configuration error, value not accepted, 0x04
DM_VARSTATE_ACCESS_FAULT (0x1000)	不允许访问变量	Bad, configuration error, value not accepted, 0x04
DM_VARSTATE_MAX_RANGE (0x0020)	超出格式上限	Uncertain, engineering unit range violation, high limit set, 0x56
DM_VARSTATE_MIN_RANGE (0x0040)	超出格式下限	Uncertain, engineering unit range violation, low limit set, 0x55
DM_VARSTATE_CONVERSION_ERROR (0x0080)	显示转换错误	Uncertain, engineering unit range violation, no limits set, 0x54
DM_VARSTATE_MAX_LIMIT (0x0008)	违反了组态的上限	映射为 Limit-Bit HIGH, 0x01，与其它变量状态（例如 good (cascade), ok）进行逻辑或连接，结果得到 0xC2
DM_VARSTATE_MIN_LIMIT (0x0010)	违反了组态的下限	映射为 Limit-Bit LOW, 0x02，与其它变量状态（例如 good (cascade), ok）进行逻辑或连接，结果得到 0xC1
DM_VARSTATE_STARTUP_VALUE (0x0100)	变量的初始化值	Uncertain, initial value, 0x4C
DM_VARSTATE_DEFAULT_VALUE (0x0200)	变量的替换值	Uncertain, substitute-set, 0x48

所需文件

dmclient.h

API 函数

DM_NOTIFY_VARIABLEEX_PROC (页 2119)	获取变量值（回调）
DMGetValueEx (页 1980)	获取变量值

参见

DM_NOTIFY_VARIABLEEX_PROC (页 2119)

DM_TYPEREF (页 1852)

DM_VARKEY (页 1869)

DMGetValueEx (页 1980)

3.2.2.16 DM_VARFILTER

声明

```
typedef struct {
    DWORD      dwFlags;
    DWORD      dwNumTypes;
    LPDWORD    pdwTypes;
    LPSTR      lpszGroup;
    LPSTR      lpszName;
    LPSTR      lpszConn;
    }
    DM_VARFILTER;
```

成员

dwFlags

dwFlags 参数可用于设置以下变量的选择标准：

DM_VARFILTER_TYPE	0x00000001	变量类型 (pdwTypes) 用作选择标准。
DM_VARFILTER_GROUP	0x00000002	组名称 (lpszGroup) 用作选择标准。

DM_VARFILTER_NAME	0x00000004	变量名 (lpszName) 用作选择标准。
DM_VARFILTER_CONNECTION	0x00000008	逻辑连接的名称 (lpszConn) 用作选择标准。
DM_VARFILTER_FAST_CALLBACK	0x00010000	(DMEnumVariables 的标记):如设置此标记，则不再缓冲回调数据。因此，回调操作将直接从数据库返回数据库。 此标记的优点是可以在发送第一个回调之前实现更快的响应。同时，还避免了因数据量大导致内存负担很重。 **注意** 不能在此回调中调用任何其它 DMClient 函数。否则可能会出现堵塞现象。
DM_VARFILTER_LOCAL_ONLY	0x00020000	(DMEnumVariables 的标记):如果设置此标记，则只列举本地变量。不需要项目中所用的来自其它服务器的变量。

dwNumTypes

在 pdwTypes 中指定的变量类型的数目。

pdwTypes

pdwTypes 用于标识要用作选择标准的变量类型。

DM_VARTYPE_BIT	二进制变量
DM_VARTYPE_SBYTE	有符号 8 位数
DM_VARTYPE_BYTE	无符号 8 位数
DM_VARTYPE_SWORD	有符号 16 位数
DM_VARTYPE_WORD	无符号 16 位数
DM_VARTYPE_SDWORD	有符号 32 位数
DM_VARTYPE_DWORD	无符号 32 位数
DM_VARTYPE_FLOAT	32 位 IEEE 754 浮点数
DM_VARTYPE_DOUBLE	64 位 IEEE 754 浮点数
DM_VARTYPE_TEXT_8	文本变量，8 位字符集
DM_VARTYPE_TEXT_16	文本变量，16 位字符集

不支持 DM_VARTYPE_RAW、DM_VARTYPE_STRUCT 和 DM_VARTYPE_TEXTREF 类型。

指定适当的 TypID，以选择具有自定义结构的结构化变量。使用 GAPIEnumTypes 函数来确定 TypID。

lpszGroup

指向变量组名称的指针。此名称应用作选择标准。禁止使用通配符。

lpszName

指向变量名的指针。此名称应用作选择标准。禁止使用通配符。

lpszConn

指向逻辑连接名称的指针。此名称应用作选择标准。禁止使用通配符。

所需文件

dmclient.h

API 函数

DMEnumVariables (页 1974)	列出变量名
DMShowVarDatabase (页 2052)	打开变量选择对话框
DMShowVarDatabaseMulti (页 2057)	打开变量选择对话框

参见

DMEnumVariables (页 1974)

DMShowVarDatabase (页 2052)

DMShowVarDatabaseMulti (页 2057)

3.2.2.17 DM_VARGRP_DATA

声明

```
typedef struct {
    CHAR      szName[ MAX_DM_VAR_NAME + 1 ];
    DWORD     dwCreatorID;
    WORD      dwVarNum;
    LPVOID    lpvUserData;
    }
    DM_VARGRP_DATA;
```

成员

szName

变量组的名称

dwCreatorID

通过创建者标识，可确定对象的创建者。

保留值 0 – 10100 以及 11000 – 11100，供内部或特定系统使用。

dwVarNum

变量组中的变量数目

lpvUserData

指向应用程序特定数据的指针。

所需文件

dmclient.h

API 函数

DM_ENUM_VARGRP_PROC (页 1973)	列出变量组的相关信息（回调）

参见

DM_ENUM_VARGRP_PROC (页 1973)

3.2.2.18 DM_VARGRPKEY

声明

```
typedef struct {
    CHAR      szName[ MAX_DM_VAR_NAME + 1 ];
    LPVOID    lpvUserData;
    }
    DM_VARGRPKEY;
```

成员

szName

变量组的名称

lpvUserData

指向应用程序特定数据的指针。

所需文件

dmclient.h

API 函数

DMEnumVarGrpData (页 1969)	列出变量组的相关信息

参见

DMEnumVarGrpData (页 1969)

3.2.2.19 DM_VARIABLE_DATA

声明

```
typedef struct {
    DM_TYPEREF            dmTypeRef;
    DM_VARLIMIT           dmVarLimit;
    VARIANT               dmStart;
    VARIANT               dmDefault;
    DWORD                 dwNotify;
    DWORD                 dwFlags;
    CHAR                  szSpecific[MAX_DM_VAR_SPECIFIC +1];
    CHAR                  szGroup[MAX_DM_GROUP_NAME +1];
    CHAR                  szConnection[
                          MAX_DM_CONNECTION_NAME +1];
    CHAR                  szChannel[_MAX_PATH +1];
    CHAR                  szUnit[MAX_DM_UNIT_NAME +1];
    }
    DM_VARIABLE_DATA;
```

成员

dmTypeRef

DM_TYPEREF (页 1852) 结构包含有关变量类型的信息。

dmVarLimit

DM_VARLIMIT (页 1871) 结构包含有关变量限制的信息。

dmStart

变量的起始值。使用后必须用 VariantClear(&dmStart) 释放。

dmDefault

变量的替换值。使用后必须用 VariantClear(&dmDefault) 释放。

dwNotify

dwNotify 指定生成报告条目的事件：

DM_NOTIFY_MAX_LIMIT	0x00000001	达到上限时
DM_NOTIFY_MIN_LIMIT	0x00000002	达到下限时
DM_NOTIFY_FORMAT_ERROR	0x00000004	出现转换错误时
DM_NOTIFY_ACCESS_FAULT	0x00000008	执行禁止的写访问时
DM_NOTIFY_APPLICATION_WRITE	0x00000010	应用程序执行写访问时
DM_NOTIFY_PROCESS_WRITE	0x00000020	过程执行写访问时

dwFlags

指示应如何使用替换值：

DM_HAS_MIN_LIMIT	0x00000008	变量有固定下限
DM_HAS_MAX_LIMIT	0x00000010	变量有固定上限
DM_HAS_DEFAULT_VALUE	0x00000020	变量有替换值
DM_HAS_STARTUP_VALUE	0x00000040	变量有初始值
DM_USE_DEFAULT_ON_STARTUP	0x00000080	在系统启动时输入替换值
DM_USE_DEFAULT_ON_MAX	0x00000100	违反上限时输入替换值
DM_USE_DEFAULT_ON_MIN	0x00000200	违反下限时输入替换值
DM_USE_DEFAULT_ON_COMM_ERROR	0x00000400	在出现连接错误时替换值

szSpecific

szSpecific 包含变量的地址关系，例如块中的数据块和字节等。有关 PLC 的特定详细信息，请参见通信手册。

该值与在 WinCC 中变量属性的“参数”列中显示的值相同。

szGroup

变量所属的组的名称。
没有为多客户端上的服务器变量提供该值。

szConnection

变量所链接的逻辑连接的名称。

szChannel

通道驱动程序的文件名。

szUnit

变量所链接的通道单元的名称。

所需文件

dmclient.h

API 函数

DM_ENUM_VARIABLE_PROC (页 1963)	列出变量的相关信息（回调）

参见

DM_TYPEREF (页 1852)

DM_ENUM_VARIABLE_PROC (页 1963)

DM_VARLIMIT (页 1871)

DMEnumVarData (页 1961)

3.2.2.20 DM_VARIABLE_DATA4

声明

```
typedef struct {
    DM_TYPEREF                dmTypeRef;
    DM_VARLIMIT                dmVarLimit;
    VARIANT                    dmStart;
    VARIANT                    dmDefault;
    DWORD                     dwNotify;
    DWORD                     dwFlags;
    CHAR                     szSpecific[MAX_DM_VAR_SPECIFIC +1];
    CHAR                     szGroup[MAX_DM_GROUP_NAME +1];
    CHAR                     szConnection[
                             MAX_DM_CONNECTION_NAME +1];
    CHAR                     szChannel[_MAX_PATH +1];
    CHAR                     szUnit[MAX_DM_UNIT_NAME +1];
    MCP_VARIABLE_SCALES        Scaling;
    DWORD                     dwASDataSize;
    DWORD                     dwOSDataSize;
    DWORD                     dwVarProperty;
    DWORD                     dwFormat;
    }
    DM_VARIABLE_DATA4;
```

成员

dmTypeRef

DM_TYPEREF (页 1852) 结构包含有关变量类型的信息。

dmVarLimit

DM_VARLIMIT (页 1871) 结构包含有关变量限制的信息。

dmStart

变量的起始值。使用后必须用 VariantClear(&dmStart) 释放。

dmDefault

变量的替换值。使用后必须用 VariantClear(&dmDefault) 释放。

dwNotify

dwNotify 指定生成报告条目的事件：

DM_NOTIFY_MAX_LIMIT	0x00000001	达到上限时
DM_NOTIFY_MIN_LIMIT	0x00000002	达到下限时
DM_NOTIFY_FORMAT_ERROR	0x00000004	出现转换错误时
DM_NOTIFY_ACCESS_FAULT	0x00000008	执行禁止的写访问时
DM_NOTIFY_APPLICATION_WRITE	0x00000010	应用程序执行写访问时
DM_NOTIFY_PROCESS_WRITE	0x00000020	过程执行写访问时

dwFlags

指示应如何使用替换值：

DM_HAS_MIN_LIMIT	0x00000008	变量有固定下限
DM_HAS_MAX_LIMIT	0x00000010	变量有固定上限
DM_HAS_DEFAULT_VALUE	0x00000020	变量有替换值
DM_HAS_STARTUP_VALUE	0x00000040	变量有初始值
DM_USE_DEFAULT_ON_STARTUP	0x00000080	在系统启动时输入替换值
DM_USE_DEFAULT_ON_MAX	0x00000100	违反上限时输入替换值
DM_USE_DEFAULT_ON_MIN	0x00000200	违反下限时输入替换值
DM_USE_DEFAULT_ON_COMM_ERROR	0x00000400	在出现连接错误时替换值

szSpecific

szSpecific 包含变量的地址关系，例如块中的数据块和字节等。有关 PLC 的特定详细信息，请参见通信手册。

该值与 WinCC 中变量属性的参数列中显示的值相同。

szGroup

变量所属的组的名称。

szConnection

变量所链接的逻辑连接的名称。

szChannel

通道驱动程序的文件名。

szUnit

变量所链接的通道单元的名称。

Scaling

具有变量标定说明的 MCP_VARIABLE_SCALES (页 1898) 结构。

dwASDataSize

PLC 中变量的长度（位数）

dwOSDataSize

OS 中变量的长度（位数）

dwVarProperty

指示变量是内部变量还是外部变量：

DM_INTERNAL_VAR	0x00000002	内部变量
DM_EXTERNAL_VAR	0x00000004	外部变量

dwFormat

所用转换例程的数目。更多详细信息，请参见“转换例程”部分。

所需文件

dmclient.h

API 函数

DM_ENUM_VARIABLE_PROC4 (页 1967)	列出变量的相关信息（回调）

参见

DMEnumVarData4 (页 1965)

DM_ENUM_VARIABLE_PROC4 (页 1967)

DM_TYPEREF (页 1852)

DM_VARLIMIT (页 1871)

MCP_VARIABLE_SCALES (页 1898)

3.2.2.21 DM_VARKEY

声明

```
typedef struct {
    DWORD     dwKeyType;
    DWORD     dwID;
    char      szName[ MAX_DM_VAR_NAME + 1 ];
    LPVOID    lpvUserData;
    }
    DM_VARKEY;
```

成员

dwKeyType

dwKeyType 定义是通过关键字 ID 还是名称来寻址变量。

DM_VARKEY_ID	说明通过关键字 ID
DM_VARKEY_NAME	说明通过变量名

当 DM_VARKEY 用作返回结构时，如果返回关键字 ID 和变量名，则可同时设置这两种类型。

若 DM_VARKEY 用作源参数，则应优先设置变量名，因为只能在其中指定服务器前缀。如果使用关键字 ID，则操作始终在本地执行。

dwID

如果相应设置了 dwKeyType 类型，则包含变量的关键字 ID。

szName

如果相应设置了 dwKeyType 类型，则包含变量的名称。

lpvUserData

指向应用程序特定数据的指针。

备注

DM_VARKEY 结构是 DM_VAR_UPDATE_STRUCT (页 1854) 结构的一部分。

所需文件

dmclient.h

API 函数

DM_ENUM_TYPEMEMBERS_PROC_EX (页 2079)	列出结构化变量中的变量（回调）
DM_ENUM_TYPEMEMBERS_PROC (页 2074)	列出结构化变量中的变量（回调）
DM_ENUM_VAR_PROC (页 1976)	列出变量名（回调）
DM_ENUM_VARIABLE_PROC (页 1963)	列出变量的相关信息（回调）
DM_NOTIFY_SELECT_VAR_PROC (页 2063)	打开变量选择对话框（回调）
DMEnumVarData (页 1961)	列出关于变量的信息
DMGetValue (页 1977)	获取变量值
DMGetValueWait (页 1991)	获取更新的变量值
DMGetVarLimits (页 2010)	获取变量的限值
DMGetVarInfo (页 1998)	获取变量 ID，获取变量名
DMGetVarType (页 2017)	获取变量的数据类型
DMSetValue (页 2024)	更改变量值
DMSetValueMessage (页 2030)	更改变量值并发出报警
DMSetValueWait (页 2035)	更改变量值并发出通知
DMSetValueWaitMessage (页 2041)	更改变量值并发出通知和报警
DMShowVarDatabase (页 2052)	打开变量选择对话框
DMStartVarUpdate (页 2105)	定义更新变量

参见

DMStartVarUpdate (页 2105)

DM_VAR_UPDATE_STRUCT (页 1854)

DMGetValue (页 1977)

DMGetValueWait (页 1991)

DMGetVarInfo (页 1998)

DMGetVarLimits (页 2010)

DMGetVarType (页 2017)

DMSetValue (页 2024)

DMSetValueMessage (页 2030)

DMSetValueWait (页 2035)

DMSetValueWaitMessage (页 2041)

DMShowVarDatabase (页 2052)

DMEnumVarData (页 1961)

DM_ENUM_VAR_PROC (页 1976)

DM_ENUM_VARIABLE_PROC (页 1963)

DM_ENUM_TYPEMEMBERS_PROC (页 2074)

DM_ENUM_TYPEMEMBERS_PROC_EX (页 2079)

DM_NOTIFY_SELECT_VAR_PROC (页 2063)

DMStartVarUpdateEx (页 2109)

DM_VAR_UPDATE_STRUCTEX (页 1856)

DM_ENUM_VARIABLE_PROC4 (页 1967)

DMGetValueEx (页 1980)

DMGetValueWaitEx (页 1994)

DM_ENUM_TYPEMEMBERS_PROC_EX4 (页 2082)

MSG_CSDATA_STRUCT_PLUS (页 2759)

3.2.2.22 DM_VARLIMIT

声明

```
typedef struct {
    VARIANT    dmMaxRange;
    VARIANT    dmMinRange;
    VARIANT    dmMaxLimit;
    VARIANT    dmMinLimit;
    }
    DM_VARLIMIT;
```

成员

dmMaxRange

格式转换的上限

dmMinRange

格式转换的下限

dmMaxLimit

变量的上限

dmMinLimit

变量的下限

备注

DM_VARLIMIT 是 DM_VARIABLE_DATA (页 1863) 结构的一部分。所有 VARIANT 在使用后都必须用 VariantClear(&dmxxx) 释放。

所需文件

dmclient.h

API 函数

DMGetVarLimits (页 2010)	获取变量的限值

参见

DM_VARIABLE_DATA (页 1863)

DMGetVarLimits (页 2010)

DM_VARIABLE_DATA4 (页 1866)

3.2.2.23 MCP_NEWVARIABLE_DATA

声明

```
typedef struct {
    DWORD                   dwFlags;
    char                    szProjectFile[_MAX_PATH +1];
    char                    szConnection[
                            MAX_DM_CONNECTION_NAME +3];
    char                    szVarName[MAX_DM_VAR_NAME +1];
    char                    szGroupName[MAX_DM_GROUP_NAME +1];
    MCP_VARIABLE_COMMON     Common;
    MCP_VARIABLE_PROTOCOL   Protocol;
    MCP_VARIABLE_LIMITS     Limits;
    char                    szSpecific[MAX_DM_VAR_SPECIFIC +1];
    }
    MCP_NEWVARIABLE_DATA;
```

成员

dwFlags

dwFlags 指示应如何处理变量：

MCP_NVAR_FLAG_CREATE	1	创建变量
MCP_NVAR_FLAG_MODIFY	2	更改变量（不用于运行系统中）
MCP_NVAR_FLAG_TEST	3	检查变量是否存在

标记不能是 ORed。

szProjectFile

项目文件的名称，包括路径和扩展名。

项目文件名称可通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

如果输入了空字符串，则会对当前打开的项目执行内部 DMEnumOpenedProjects。

在运行系统中只能指定当前打开的项目。系统会拒绝任何其他条目，并显示错误消息 (DM_E_NOT_CONNECTED)。

szConnection

已分配给变量的逻辑连接的名称。

szVarName

要处理的变量的名称。

szGroupName

变量所属的组的名称。

如果指定的组名称位于 szConnection 所指定的连接以外的连接中，系统会忽略该组名称且不会显示错误消息，并且直接在连接中创建变量。
如果指定的组名称尚不存在，将隐式创建该组。

通过 MCP_NVAR_FLAG_MODIFY 更改变量时，将忽略组名称中的更改，并且不会显示错误消息。

Common

具有变量说明的 MCP_VARIABLE_COMMON (页 1885) 结构。

Protocol

具有变量如何处理超出限制的说明的 MCP_VARIABLE_PROTOCOL (页 1896) 结构。

Limits

具有变量限制的 MCP_VARIABLE_LIMITS (页 1890) 结构。

szSpecific

szSpecific 包含变量的地址关系，例如块中的数据块和字节等。有关 PLC 的特定详细信息，请参见通信手册。

该值与在 WinCC 中变量属性的“参数”列中显示的值相同。

备注

MCP_NEWVARIABLE_DATA_EX (页 1880) 结构具有扩展功能的类似用途。

所需文件

dmclient.h

API 函数

GAPICreateNewVariable (页 2065)	创建变量

参见

GAPICreateNewVariable (页 2065)

MCP_VARIABLE_PROTOCOL (页 1896)

MCP_VARIABLE_COMMON (页 1885)

MCP_VARIABLE_LIMITS (页 1890)

MCP_NEWVARIABLE_DATA_EX (页 1880)

3.2.2.24 MCP_NEWVARIABLE_DATA_4

声明

```
typedef struct {
    DWORD                   dwFlags;
    char                    szProjectFile[_MAX_PATH +1];
    char                    szConnection[
                            MAX_DM_CONNECTION_NAME +3];
    char                    szVarName[MAX_DM_VAR_NAME +1];
    char                    szGroupName[MAX_DM_GROUP_NAME +1];
    MCP_VARIABLE_COMMON     Common;
    MCP_VARIABLE_PROTOCOL   Protocol;
    MCP_VARIABLE_LIMITS     Limits;
    char                    szSpecific[MAX_DM_VAR_SPECIFIC +1];
    MCP_VARIABLE_SCALES     Scaling;
    }
    MCP_NEWVARIABLE_DATA_4;
```

成员

dwFlags

dwFlags 指示应如何处理变量：

MCP_NVAR_FLAG_CREATE	1	创建变量
MCP_NVAR_FLAG_MODIFY	2	更改变量（如果不可用，则重新创建）（不用于运行系统中）
MCP_NVAR_FLAG_TEST	3	检查变量是否存在

标记不能是 ORed。

szProjectFile

项目文件的名称，包括路径和扩展名。

项目文件名称可通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

如果输入了空字符串，则会对当前打开的项目执行内部 DMEnumOpenedProjects。

在运行系统中只能指定当前打开的项目。系统会拒绝所有其它指定，并显示错误 DM_E_NOT_CONNECTED。

szConnection

已分配给变量的逻辑连接的名称。

szVarName

要处理的变量的名称。

szGroupName

变量所属的组的名称。

如果指定的组名称位于 szConnection 所指定的连接以外的连接中，系统会忽略该组名称且不会显示错误消息，并且直接在连接中创建变量。
如果指定的组名称尚不存在，将隐式创建该组。

通过 MCP_NVAR_FLAG_MODIFY 更改变量时，将忽略组名称中的更改，并且不会显示错误消息。

Common

具有变量说明的 MCP_VARIABLE_COMMON (页 1885) 结构。

Protocol

具有变量如何处理超出限制的说明的 MCP_VARIABLE_PROTOCOL (页 1896) 结构。

Limits

具有变量限制的 MCP_VARIABLE_LIMITS (页 1890) 结构。

szSpecific

szSpecific 包含变量的地址关系，例如块中的数据块和字节等。有关 PLC 的特定详细信息，请参见通信手册。

该值与在 WinCC 中变量属性的“参数”列中显示的值相同。

Scaling

具有变量标定说明的 MCP_VARIABLE_SCALES (页 1898) 结构。

备注

MCP_NEWVARIABLE_DATA_EX4 (页 1883) 结构具有扩展功能的类似用途。

所需文件

dmclient.h

API 函数

GAPICreateNewVariable4 (页 2067)	创建变量
GAPICreateNewVariableEx4 (页 2070)	创建带有创建者 ID 的变量

参见

MCP_VARIABLE_COMMON (页 1885)

MCP_VARIABLE_LIMITS (页 1890)

MCP_VARIABLE_PROTOCOL (页 1896)

GAPICreateNewVariable4 (页 2067)

MCP_VARIABLE_SCALES (页 1898)

MCP_NEWVARIABLE_DATA_EX4 (页 1883)

GAPICreateNewVariableEx4 (页 2070)

3.2.2.25 MCP_NEWVARIABLE_DATA_5

声明

```
typedef struct {
    DWORD                   dwFlags;
    char                    szProjectFile[_MAX_PATH +1];
    char                    szConnection[
                            MAX_DM_CONNECTION_NAME +3];
    char                    szVarName[MAX_DM_VAR_NAME +1];
    char                    szGroupName[MAX_DM_GROUP_NAME +1];
    MCP_VARIABLE_COMMON     Common;
    MCP_VARIABLE_PROTOCOL   Protocol;
    MCP_VARIABLE_LIMITS5    Limits;
    char                    szSpecific[MAX_DM_VAR_SPECIFIC +1];
    MCP_VARIABLE_SCALES     Scaling;
    }
    MCP_NEWVARIABLE_DATA_5;
```

成员

dwFlags

dwFlags 指示应如何处理变量：

MCP_NVAR_FLAG_CREATE	1	创建变量
MCP_NVAR_FLAG_MODIFY	2	更改变量（如果不可用，则重新创建）（不用于运行系统中）
MCP_NVAR_FLAG_TEST	3	检查变量是否存在

标记不能是 ORed。

szProjectFile

项目文件的名称，包括路径和扩展名。

项目文件名称可通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

如果输入了空字符串，则会对当前打开的项目执行内部 DMEnumOpenedProjects。

在运行系统中只能指定当前打开的项目。系统会拒绝所有其它指定，并显示错误 DM_E_NOT_CONNECTED。

szConnection

已分配给变量的逻辑连接的名称。

szVarName

要处理的变量的名称。

szGroupName

变量所属的组的名称。

如果指定的组名称位于 szConnection 所指定的连接以外的连接中，系统会忽略该组名称且不会显示错误消息，并且直接在连接中创建变量。
如果指定的组名称尚不存在，将隐式创建该组。

通过 MCP_NVAR_FLAG_MODIFY 更改变量时，将忽略组名称中的更改，并且不会显示错误消息。

Common

具有变量说明的 MCP_VARIABLE_COMMON (页 1885) 结构。

Protocol

具有变量如何处理超出限制的说明的 MCP_VARIABLE_PROTOCOL (页 1896) 结构。

Limits

具有变量限制的 MCP_VARIABLE_LIMITS5 (页 1892) 结构。

szSpecific

szSpecific 包含变量的地址关系，例如块中的数据块和字节等。有关 PLC 的特定详细信息，请参见通信手册。

该值与在 WinCC 中变量属性的“参数”列中显示的值相同。

Scaling

具有变量标定说明的 MCP_VARIABLE_SCALES (页 1898) 结构。

备注

MCP_NEWVARIABLE_DATA_EX4 (页 1883) 结构具有扩展功能的类似用途。

所需文件

dmclient.h

API 函数

GAPICreateNewVariable5 (页 2069)	创建带有创建者 ID 的变量

参见

GAPICreateNewVariable5 (页 2069)

MCP_VARIABLE_COMMON (页 1885)

MCP_VARIABLE_LIMITS5 (页 1892)

MCP_VARIABLE_PROTOCOL (页 1896)

MCP_NEWVARIABLE_DATA_EX4 (页 1883)

MCP_VARIABLE_SCALES (页 1898)

3.2.2.26 MCP_NEWVARIABLE_DATA_EX

声明

```
typedef struct {
    DWORD                       dwFlags;
    char                        szProjectFile[_MAX_PATH +1];
    char                        szConnection[
                                MAX_DM_CONNECTION_NAME +3];
    char                        szVarName[MAX_DM_VAR_NAME +1];
    char                        szGroupName[MAX_DM_GROUP_NAME +1];
    MCP_VARIABLE_COMMON_EX      Common;
    MCP_VARIABLE_PROTOCOL_EX    Protocol;
    MCP_VARIABLE_LIMITS_EX      Limits;
    char                        szSpecific[MAX_DM_VAR_SPECIFIC +1]
    }
    MCP_NEWVARIABLE_DATA_EX;
```

成员

dwFlags

dwFlags 指示应如何处理变量：

MCP_NVAR_FLAG_CREATE	1	创建变量
MCP_NVAR_FLAG_MODIFY	2	更改变量（不用于运行系统中）
MCP_NVAR_FLAG_TEST	3	检查变量是否存在。

标记不能是 ORed。

szProjectFile

项目文件的名称，包括路径和扩展名。

项目文件名称可通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

如果输入了空字符串，则会对当前打开的项目执行内部 DMEnumOpenedProjects。

在运行系统中只能指定当前打开的项目。系统会拒绝所有其它指定，并显示错误 DM_E_NOT_CONNECTED。

szConnection

已分配给变量的逻辑连接的名称。

szVarName

要处理的变量的名称。

szGroupName

变量所属的组的名称。

如果指定的组名称位于 szConnection 所指定的连接以外的连接中，系统会忽略该组名称且不会显示错误消息，并且直接在连接中创建变量。
如果指定的组名称尚不存在，将隐式创建该组。

通过 MCP_NVAR_FLAG_MODIFY 更改变量时，将忽略组名称中的更改，并且不会显示错误消息。

Common

具有变量说明的 MCP_VARIABLE_COMMON_EX (页 1887) 结构。

Protocol

具有变量如何处理超出限制的说明的 MCP_VARIABLE_PROTOCOL_EX (页 1897) 结构。

Limits

具有变量限制的 MCP_VARIABLE_LIMITS_EX (页 1894) 结构。

szSpecific

szSpecific 包含变量的地址关系，例如块中的数据块和字节等。有关 PLC 的特定详细信息，请参见通信手册。

该值与在 WinCC 中变量属性的“参数”列中显示的值相同。

备注

MCP_NEWVARIABLE_DATA (页 1873) 结构具有类似用途。

所需文件

dmclient.h

API 函数

DM_ENUM_TYPEMEMBERS_PROC_EX (页 2079)	列出结构化变量中的变量（回调）

参见

MCP_NEWVARIABLE_DATA (页 1873)

DM_ENUM_TYPEMEMBERS_PROC_EX (页 2079)

MCP_VARIABLE_COMMON_EX (页 1887)

MCP_VARIABLE_LIMITS_EX (页 1894)

MCP_VARIABLE_PROTOCOL_EX (页 1897)

3.2.2.27 MCP_NEWVARIABLE_DATA_EX4

声明

```
typedef struct {
    DWORD                       dwFlags;
    char                        szProjectFile[_MAX_PATH +1];
    char                        szConnection[
                                MAX_DM_CONNECTION_NAME +3];
    char                        szVarName[MAX_DM_VAR_NAME +1];
    char                        szGroupName[MAX_DM_GROUP_NAME +1];
    MCP_VARIABLE_COMMON_EX      Common;
    MCP_VARIABLE_PROTOCOL_EX    Protocol;
    MCP_VARIABLE_LIMITS_EX      Limits;
    char                        szSpecific[MAX_DM_VAR_SPECIFIC +1]
    MCP_VARIABLE_SCALES         Scaling;
    }
    MCP_NEWVARIABLE_DATA_EX4;
```

成员

dwFlags

dwFlags 指示应如何处理变量：

MCP_NVAR_FLAG_CREATE	1	创建变量
MCP_NVAR_FLAG_MODIFY	2	更改变量（不用于运行系统中）
MCP_NVAR_FLAG_TEST	3	检查变量是否存在。

标记不能是 ORed。

szProjectFile

项目文件的名称，包括路径和扩展名。

项目文件名称可通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

如果输入了空字符串，则会对当前打开的项目执行内部 DMEnumOpenedProjects。

在运行系统中只能指定当前打开的项目。系统会拒绝所有其它指定，并显示错误 DM_E_NOT_CONNECTED。

szConnection

已分配给变量的逻辑连接的名称。

szVarName

要处理的变量的名称。

szGroupName

变量所属的组的名称。

如果指定的组名称位于 szConnection 所指定的连接以外的连接中，系统会忽略该组名称且不会显示错误消息，并且直接在连接中创建变量。
如果指定的组名称尚不存在，将隐式创建该组。

通过 MCP_NVAR_FLAG_MODIFY 更改变量时，将忽略组名称中的更改，并且不会显示错误消息。

Common

具有变量说明的 MCP_VARIABLE_COMMON_EX (页 1887) 结构。

Protocol

具有变量如何处理超出限制的说明的 MCP_VARIABLE_PROTOCOL_EX (页 1897) 结构。

Limits

具有变量限制的 MCP_VARIABLE_LIMITS_EX (页 1894) 结构。

szSpecific

szSpecific 包含变量的地址关系，例如块中的数据块和字节等。有关 PLC 的特定详细信息，请参见通信手册。

该值与在 WinCC 中变量属性的“参数”列中显示的值相同。

Scaling

具有变量标定说明的 MCP_VARIABLE_SCALES (页 1898) 结构。

备注

MCP_NEWVARIABLE_DATA_4 (页 1875) 结构具有类似用途。

所需文件

dmclient.h

API 函数

DM_ENUM_TYPEMEMBERS_PROC_EX4 (页 2082)	列出结构化变量中的变量（回调）

参见

MCP_NEWVARIABLE_DATA_4 (页 1875)

MCP_NEWVARIABLE_DATA_5 (页 1878)

DM_ENUM_TYPEMEMBERS_PROC_EX4 (页 2082)

MCP_VARIABLE_COMMON_EX (页 1887)

MCP_VARIABLE_LIMITS_EX (页 1894)

MCP_VARIABLE_PROTOCOL_EX (页 1897)

MCP_VARIABLE_SCALES (页 1898)

3.2.2.28 MCP_VARIABLE_COMMON

声明

```
typedef struct {
    DWORD    dwVarType;
    DWORD    dwVarLength;
    DWORD    dwVarProperty;
    DWORD    dwFormat;
    }
    MCP_VARIABLE_COMMON;
```

成员

dwVarType

在 dwVarType 中指定变量类型：

DM_VARTYPE_BIT	二进制变量
DM_VARTYPE_SBYTE	有符号 8 位数
DM_VARTYPE_BYTE	无符号 8 位数

DM_VARTYPE_SWORD	有符号 16 位数
DM_VARTYPE_WORD	无符号 16 位数
DM_VARTYPE_SDWORD	有符号 32 位数
DM_VARTYPE_DWORD	无符号 32 位数
DM_VARTYPE_FLOAT	32 位 IEEE 754 浮点数
DM_VARTYPE_DOUBLE	64 位 IEEE 754 浮点数
DM_VARTYPE_TEXT_8	文本变量，8 位字符集
DM_VARTYPE_TEXT_16	文本变量，16 位字符集
DM_VARTYPE_RAW	原始数据类型
DM_VARTYPE_STRUCT	结构变量
DM_VARTYPE_TEXTREF	文本库中的文本参考

dwVarLength

变量长度的指定仅与文本变量 DM_VARTYPE_TEXT_8 和 DM_VARTYPE_TEXT_16 相关。

此处以字符为单位指定文本长度（1 到 255）。

dwVarProperty

指示变量是内部变量还是外部变量：

DM_INTERNAL_VAR	内部变量
DM_EXTERNAL_VAR	外部变量

dwFormat

要使用的转换例程的数量。更多详细信息，请参见“转换例程”部分。

备注

MCP_VARIABLE_COMMON 是 MCP_NEWVARIABLE_DATA (页 1873)
和 MCP_NEWVARIABLE_DATA_4 (页 1875) 结构的一部分。

MCP_VARIABLE_COMMON_EX (页 1887) 结构具有扩展功能的类似用途。

所需文件

dmclient.h

参见

MCP_NEWVARIABLE_DATA (页 1873)

MCP_NEWVARIABLE_DATA_4 (页 1875)

MCP_VARIABLE_COMMON_EX (页 1887)

MCP_NEWVARIABLE_DATA_5 (页 1878)

3.2.2.29 MCP_VARIABLE_COMMON_EX

声明

```
typedef struct {
    DWORD    dwVarType;
    DWORD    dwCreatorID
    DWORD    dwVarLength;
    DWORD    dwVarProperty;
    DWORD    dwFormat;
    DWORD    dwOSOffset;
    DWORD    dwASOffset;
    char     szStructTypeName
    }
    MCP_VARIABLE_COMMON_EX;
```

成员

dwVarType

在 dwVarType 中指定变量类型：

DM_VARTYPE_BIT	二进制变量
DM_VARTYPE_SBYTE	有符号 8 位数
DM_VARTYPE_BYTE	无符号 8 位数
DM_VARTYPE_SWORD	有符号 16 位数
DM_VARTYPE_WORD	无符号 16 位数
DM_VARTYPE_SDWORD	有符号 32 位数
DM_VARTYPE_DWORD	无符号 32 位数
DM_VARTYPE_FLOAT	32 位 IEEE 754 浮点数
DM_VARTYPE_DOUBLE	64 位 IEEE 754 浮点数

DM_VARTYPE_TEXT_8	文本变量，8 位字符集
DM_VARTYPE_TEXT_16	文本变量，16 位字符集
DM_VARTYPE_RAW	原始数据类型
DM_VARTYPE_STRUCT	结构变量
DM_VARTYPE_TEXTREF	文本库中的文本参考

dwCreatorID

通过创建者标识，可确定对象的创建者。

保留值 0 – 10100 以及 11000 – 11100，供内部或特定系统使用。

dwVarLength

变量长度的指定仅与文本变量 DM_VARTYPE_TEXT_8 和 DM_VARTYPE_TEXT_16 相关。此处以字符为单位指定文本长度（1 到 255）。

dwVarProperty

指示变量是内部变量还是外部变量：

DM_INTERNAL_VAR	内部变量
DM_EXTERNAL_VAR	外部变量

dwFormat

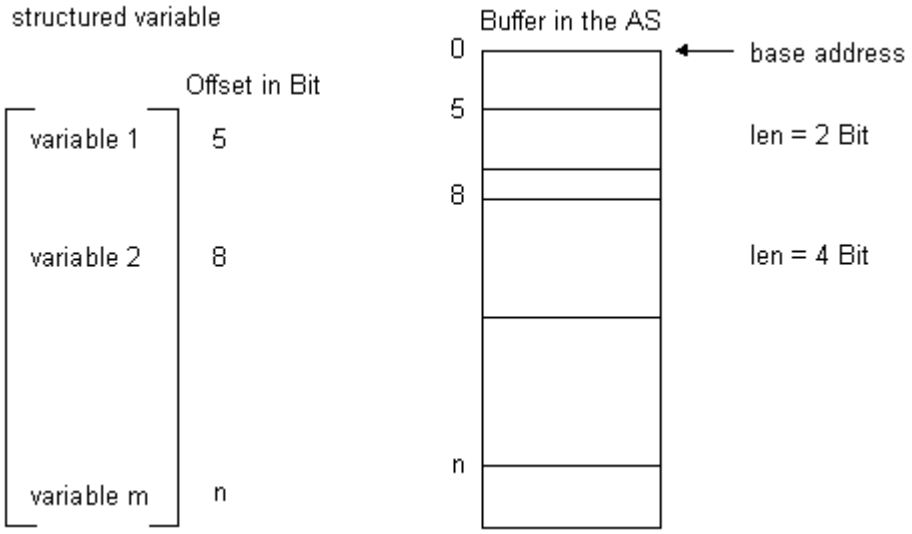
要使用的转换例程的数量。更多详细信息，请参见“转换例程”部分。

dwOSOffset

该参数为将来开发预留，必须预设为 0。

dwASOffset

PLC 的缓冲区中的偏移量



以字节为单位内部处理面向位的 ASOffset。

szStructTypeName

结构类型的名称

备注

MCP_VARIABLE_COMMON_EX 是 MCP_NEWVARIABLE_DATA_EX (页 1880) 结构的一部分。

MCP_VARIABLE_COMMON (页 1885) 结构具有类似用途。

所需文件

dmclient.h

参见

MCP_NEWVARIABLE_DATA (页 1873)

MCP_VARIABLE_COMMON (页 1885)

MCP_NEWVARIABLE_DATA_EX (页 1880)

MCP_NEWVARIABLE_DATA_EX4 (页 1883)

3.2.2.30 MCP_VARIABLE_LIMITS

声明

```
typedef struct {
    double    dTopLimit;
    double    dBottomLimit;
    double    dStartValue;
    double    dSubstituteValue;
    BOOL      bTopLimit;
    BOOL      bBottomLimit;
    BOOL      bStartValue;
    BOOL      bConnectionErr;
    BOOL      bTopLimitValid;
    BOOL      bBottomLimitValid;
    BOOL      bStartValueValid;
    BOOL      bSubstValueValid;
    }
    MCP_VARIABLE_LIMITS;
```

成员

dTopLimit

变量上限值

dBottomLimit

变量下限值

dStartValue

变量起始值

dSubstituteValue

变量的替换值

bTopLimit

如果已设置此参数，当变量值超出 dTopLimit 中所指定的值时，应使用替换值。

bBottomLimit

如果已设置此参数，当变量值低于 dBottomLimit 中所指定的值时，应使用替换值。

bStartValue

如果已设置此参数，替换值应用作起始值。

bConnectionErr

如果已设置此参数，当连接出错时，应使用替换值。

bTopLimitValid

如果已设置此参数，dTopLimit 中所指定的值将应用上限值。

bBottomLimitValid

如果已设置此参数，dBottomLimit 中所指定的值将应用下限值。

bStartValueValid

如果已设置此参数，dStartValue 中所指定的值将应用起始值。

bSubstValueValid

如果已设置此参数，dSubstitudeValue 中所指定的值将应用替换值。

备注

MCP_VARIABLE_Limits 是 MCP_NEWVARIABLE_DATA (页 1873)
和 MCP_NEWVARIABLE_DATA_4 (页 1875) 结构的一部分。

MCP_VARIABLE_LIMITS_EX (页 1894) 结构具有扩展功能的类似用途。

所需文件

dmclient.h

参见

MCP_NEWVARIABLE_DATA (页 1873)

MCP_NEWVARIABLE_DATA_4 (页 1875)

MCP_VARIABLE_LIMITS_EX (页 1894)

3.2.2.31 MCP_VARIABLE_LIMITS5

声明

```
typedef struct {
    VARIANT    varTopLimit;
    VARIANT    varBottomLimit;
    VARIANT    varStartValue;
    VARIANT    varSubstituteValue;
    BOOL       bTopLimit;
    BOOL       bBottomLimit;
    BOOL       bStartValue;
    BOOL       bConnectionErr;
    BOOL       bTopLimitValid;
    BOOL       bBottomLimitValid;
    BOOL       bStartValueValid;
    BOOL       bSubstValueValid;
    }
    MCP_VARIABLE_LIMITS5;
```

成员

varTopLimit

变量上限值

varBottomLimit

变量下限值

varStartValue

变量起始值

varSubstituteValue

变量的替换值

bTopLimit

如果已设置此参数，当变量值超出 dTopLimit 中所指定的值时，应使用替换值。

bBottomLimit

如果已设置此参数，当变量值低于 dBottomLimit 中所指定的值时，应使用替换值。

bStartValue

如果已设置此参数，替换值应用作起始值。

bConnectionErr

如果已设置此参数，当连接出错时，应使用替换值。

bTopLimitValid

如果已设置此参数，dTopLimit 中所指定的值将应用上限值。

bBottomLimitValid

如果已设置此参数，dBottomLimit 中所指定的值将应用下限值。

bStartValueValid

如果已设置此参数，dStartValue 中所指定的值将应用起始值。

bSubstValueValid

如果已设置此参数，dSubstitudeValue 中所指定的值将应用替换值。

备注

MCP_VARIABLE_LIMITS5 是 MCP_NEWVARIABLE_DATA_5 (页 1878) 结构的一部分，需要使用 GAPICreateNewVariable5 (页 2069) 函数指定具有起始和替代值的文本变量。

所需文件

dmclient.h

参见

MCP_NEWVARIABLE_DATA_5 (页 1878)

GAPICreateNewVariable5 (页 2069)

3.2.2.32 MCP_VARIABLE_LIMITS_EX

声明

```
typedef struct {
    double    dTopLimit
    double    dBottomLimit;
    double    dStartValue;
    double    dSubstituteValue;
    DWORD     dwLimitFlags
    DWORD     dwTextBibStartText;
    char      szTextStartText[255];
    DWORD     dwTextBibSubstitude;
    char      szTextSubstitude[255]
    }
    MCP_VARIABLE_LIMITS_EX;
```

成员

dTopLimit

变量上限值

dBottomLimit

变量下限值

dStartValue

变量起始值

dSubstituteValue

变量的替换值

LimitFlags

LimitFlags 参数指定变量的默认值和限值的有效性。

MCP_VARLIM_HAS_MIN_LIMIT	变量有固定下限
MCP_VARLIM_HAS_MAX_LIMIT	变量有固定上限
MCP_VARLIM_HAS_DEFAULT_VALUE	变量有替换值
MCP_VARLIM_HAS_STARTUP_VALUE	变量有起始值
MCP_VARLIM_USE_DEFAULT_ON_STARTUP	系统启动时应使用替换值。
MCP_VARLIM_USE_DEFAULT_ON_MAX	在高于上限时应使用替换值。

MCP_VARLIM_USE_DEFAULT_ON_MIN	在低于下限时应使用替换值。
MCP_VARLIM_USE_DEFAULT_ON_COMM_ERROR	连接出错时应使用替换值。

dwTextBibStartText

该参数仅与文本变量相关。如果要用作起始值的文本应从项目文本中读取，请在此处输入相应文本的 ID。

szTextStartText

该参数仅与文本变量相关。直接在 szTextStartText 下指定要用作起始值的文本。

dwTextBibSubstitude

该参数仅与文本变量相关。如果要用作替换值的文本应从项目文本中读取，请在此处输入相应文本的 ID。

szTextSubstitude

该参数仅与文本变量相关。直接在 szTextSubstitude 下指定要用作替换值的文本。

备注

MCP_VARIABLE_LIMITS_EX 是 MCP_NEWVARIABLE_DATA_EX (页 1880) 结构的一部分。

MCP_VARIABLE_LIMITS (页 1890) 结构具有类似用途。

所需文件

dmclient.h

参见

MCP_VARIABLE_LIMITS (页 1890)

MCP_NEWVARIABLE_DATA_EX (页 1880)

MCP_NEWVARIABLE_DATA_EX4 (页 1883)

3.2.2.33 MCP_VARIABLE_PROTOCOL

声明

```
typedef struct {
    BOOL    bTopLimitErr;
    BOOL    bBottomLimitErr;
    BOOL    bTransformationErr;
    BOOL    bWriteErr;
    BOOL    bWriteErrApplication;
    BOOL    bWriteErrProzess;
    }
    MCP_VARIABLE_PROTOCOL;
```

成员

bTopLimitErr

当变量值超出上限时，会生成报告条目。

bBottomLimitErr

当变量值低于下限时，会生成报告条目。

bTransformationErr

当转换出错时，会生成报告条目。

bWriteErr

每次出现未授权的写访问时，都会生成报告条目。

bWriteErrApplication

每次应用程序进行未授权的写访问时，都会生成报告条目。

bWriteErrProzess

每次进程进行未授权的写访问时，都会生成报告条目。

备注

MCP_VARIABLE_PROTOCOL 是 MCP_NEWVARIABLE_DATA (页 1873)
和 MCP_NEWVARIABLE_DATA_4 (页 1875) 结构的一部分。

MCP_VARIABLE_PROTOCOL_EX (页 1897) 结构具有扩展功能的类似用途。

所需文件

dmclient.h

参见

MCP_NEWVARIABLE_DATA (页 1873)

MCP_NEWVARIABLE_DATA_4 (页 1875)

MCP_VARIABLE_PROTOCOL_EX (页 1897)

MCP_NEWVARIABLE_DATA_5 (页 1878)

3.2.2.34 MCP_VARIABLE_PROTOCOL_EX

声明

```
typedef struct {
    DWORD    dwProtocolFlags
    }
    MCP_VARIABLE_PROTOCOL_EX;
```

成员

dwProtocolFlags

可使用 dwProtocolFlags 参数确定会生成报告条目的事件：

MCP_VARPROT_TOPLIMITERR	当变量值超出上限时，会生成报告条目。
MCP_VARPROT_BOTTEMLIMITERR	当变量值低于下限时，会生成报告条目。
MCP_VARPROT_TRANSFORMATIONERR	当转换出错时，会生成报告条目。
MCP_VARPROT_WRITEERR	每次出现未授权的写访问时，都会生成报告条目。
MCP_VARPROT_WRITEERRAPPLICATION	每次应用程序进行未授权的写访问时，都会生成报告条目。
MCP_VARPROT_WRITEERRPROCESS	每次进程进行未授权的写访问时，都会生成报告条目。

可使用这些常量的逻辑组合。

备注

MCP_VARIABLE_PROTOCOL_EX 是 MCP_NEWVARIABLE_DATA_EX (页 1880) 结构的一部分。

MCP_VARIABLE_PROTOCOL (页 1896) 结构具有类似用途。

所需文件

dmclient.h

参见

MCP_VARIABLE_PROTOCOL (页 1896)

MCP_NEWVARIABLE_DATA_EX (页 1880)

MCP_NEWVARIABLE_DATA_EX4 (页 1883)

3.2.2.35 MCP_VARIABLE_SCALES

声明

```
typedef struct {
    DWORD     dwVarScaleFlags;
    double    doMinProc;
    double    doMaxProc;
    double    doMinVar;
    double    doMaxVar;
    }
    MCP_VARIABLE_SCALES;
```

成员

dwVarScaleFlags

DM_VARSCALE_NOSCALE：不标定

DM_VARSCALE_LINEAR：线性转换

doMinProc

过程中变量的最低值

doMaxProc

过程中变量的最高值

doMinVar

WinCC 中变量的最低值

doMaxVar

WinCC 中变量的最高值

备注

MCP_VARIABLE_SCALES 是 MCP_NEWVARIABLE_DATA_4 (页 1875) 结构的一部分。

所需文件

dmclient.h

参见

MCP_NEWVARIABLE_DATA_4 (页 1875)

MCP_NEWVARIABLE_DATA_5 (页 1878)

DM_VARIABLE_DATA4 (页 1866)

MCP_NEWVARIABLE_DATA_EX4 (页 1883)

3.2.3 常规函数

3.2.3.1 DMActivateRTProject

使用

该函数启用在 WinCC 中打开的项目。

声明

```
BOOL DMActivateRTProject (
    LPCMN_ERROR    lpdmError);
```

参数

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

项目已启用。

FALSE

错误。

注释

在内部对该函数进行异步处理。如果在内部正确转发指令，该函数将始终返回 TRUE。例如，如果项目中服务器计算机的名称未正确指定，则不会报告错误。

通过 DMGetRTProject(..) 检查成功的启动。

如果需要检查计算机名称是否正确，使用 DMGetMachineTable(..) 和 GetComputername(..) 函数。

所需文件

dmclient.h

dmclient.lib

dmclient.dll

参见

DMGetMachineTable (页 1934)

3.2.3.2 DMAddNotify

声明

```
BOOL DMAddNotify (
    DM_NOTIFY_PROC     lpfnNotify,
    LPVOID             lpvUser,
    LPDWORD            lpdwNotifyCookie,
    LPCMN_ERROR        lpdmError);
```

说明

将另一个通知函数调用连接到 DMClient。通知功能对应于可为 DMConnect 指定的功能。

参数

lpfnNotify

指向其它通知函数的指针。

lpvUser

指向应用程序特定数据的指针。此函数不会评估该指针，但会使其在通知函数中重新可用。

lpdwNotifyCookie

指向将 cookie 返回给通知的 DWORD 的指针。

DMRemoveNotify 函数之后还需要利用该 cookie 再次删除通知编辑。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已连接附加通知函数。

FALSE

错误

注释

当 DMConnect 在过程中已执行但无法访问该处指定的通知函数时，需要使用此函数。这样便可将附加通知包含在内，以便评估涉及的通知函数。

例如，当您在全局脚本函数内使用自行创建的具有 DMClient 功能的 DLL 时，将无法再在该处执行 DMConnect。由于已使用脚本执行 DMConnect，因此无法再执行通知。

在极少数情况下，通知在函数调用返回之前就已返回。

此处包含的附加通知函数必须在 DMDisconnect 之前再次使用 DMRemoveNotify 删除。另外，该通知函数不能是可被提前损坏或删除的等级对象的成员函数。最终，DMRemoveNotify 之后会执行通知调用。

错误消息

DM_E_NOT_CONNECTED	尚未执行 DMConnect。尚未初始化内部管理。
DM_E_ALREADY_EXIST	此过程中已存在具有相同地址的 NotifyProc。
DM_E_MAX_LIMIT	已达到附加通知函数的最大数量 (16)。
DM_E_PARAM	参数错误

所需文件

dmclient_exstr.h

dmclient.lib

dmclient.dll

相关函数

DMFireNotifyData	使用 cookie 将数据发送到附加通知函数或所有函数
DMRemoveNotify	使用 cookie 从通知列表中删除附加通知函数

示例

按钮 AddNotify：

```
#include "apdefap.h"
void OnClick(char* lpszPictureName,
             char* lpszObjectName,
```

```
            char* lpszPropertyName)
{
AddNotify();
}
```

项目函数 AddNotify：

```
extern BOOL DM_NotifyProcA(DWORD  dwNotifyClass,
                                   DWORD     dwNotifyCode,
                           LPBYTE lpbyData,
                           DWORD  dwItems,
                           LPVOID lpvUser);

void AddNotify()
{
BOOL bRet = FALSE;
DWORD dwNotifyCookie = 0L;
LPVOID lpvUser = NULL;
CMN_ERRORA err;

memset(&err, 0, sizeof(CMN_ERRORA));

dwNotifyCookie = GetTagDWord("DMdwNotifyCookie");
if (dwNotifyCookie)
{
          printf("\r\nremove first previous Notify Cookie=%08lx",
                     dwNotifyCookie);

            bRet = DMRemoveNotifyA(dwNotifyCookie, &err);

            if (FALSE == bRet)
            {
                printf("\r\nERROR:DMRemoveNotifyA
[%s],%ld,%ld,%ld,%ld,%ld",
                               err.szErrorText,
                               err.dwError1,
                               err.dwError2,
                               err.dwError3,
                               err.dwError4,
                               err.dwError5);
            }
            dwNotifyCookie = 0L;
            SetTagDWord("DMdwNotifyCookie",dwNotifyCookie);
}

memset(&err, 0, sizeof(CMN_ERRORA));
lpvUser = (LPVOID)dwNotifyCookie; //set only for show in notify

bRet = DMAddNotifyA(DM_NotifyProcA,
                    lpvUser,
                    &dwNotifyCookie,
                    &err);
if (bRet)
```

```
{
            printf("\r\nNotify added, Cookie=%08lx!",
dwNotifyCookie);
            SetTagDWord("DMdwNotifyCookie",dwNotifyCookie);
}
else
{
            printf("\r\nERROR:DMAddNotifyA
[%s],%ld,%ld,%ld,%ld,%ld",
                        err.szErrorText,
                        err.dwError1,
                        err.dwError2,
                        err.dwError3,
                        err.dwError4,
                        err.dwError5);
}
}
```

Project function DM_NotifyProcA:

```
extern void RemoveNotify();

BOOL DM_NotifyProcA(DWORD dwNotifyClass,
                    DWORD dwNotifyCode,
                    LPBYTE lpbyData,
                    DWORD dwItems,
                    LPVOID lpvUser)
{
BOOL bShowParams = FALSE;
BOOL bSpecialActionRemoveNotify = FALSE;

printf("\r\n\r\n#### extra added DMNotifyProc [%08lx] entry:
####", (DWORD)lpvUser);
switch (dwNotifyClass)
{
case DM_NOTIFY_CLASS_ERROR:
        switch (dwNotifyCode)
        {
        case DM_NOTIFY_SHUTDOWN:
          printf("\r\nDM_NOTIFY_CLASS_ERROR:DM_NOTIFY_SHUTDOWN");
            bShowParams = TRUE;
            bSpecialActionRemoveNotify = TRUE;
            break;
        case DM_NOTIFY_PROCESSNET_ERROR:

printf("\r\nDM_NOTIFY_CLASS_ERROR:DM_NOTIFY_PROCESSNET_ERROR");
            bShowParams = TRUE;
            break;
        case DM_NOTIFY_SYSNET_ERROR:

printf("\r\nDM_NOTIFY_CLASS_ERROR:DM_NOTIFY_SYSNET_ERROR");
            bShowParams = TRUE;
            break;
```

```
        default:
            printf("\r\nunknown
DM_NOTIFY_CLASS_ERROR:dwNotifyCode[%08lx]", dwNotifyCode);
            bShowParams = TRUE;
        }
        break;
case DM_NOTIFY_CLASS_WARNING:
        switch (dwNotifyCode)
        {
        case DM_NOTIFY_QUEUE_50_PERCENT:

printf("\r\nDM_NOTIFY_CLASS_WARNING:DM_NOTIFY_QUEUE_50_PERCENT");
            break;
        case DM_NOTIFY_QUEUE_60_PERCENT:

printf("\r\nDM_NOTIFY_CLASS_WARNING:DM_NOTIFY_QUEUE_60_PERCENT");
            break;
        case DM_NOTIFY_QUEUE_70_PERCENT:

printf("\r\nDM_NOTIFY_CLASS_WARNING:DM_NOTIFY_QUEUE_70_PERCENT");
            break;
        case DM_NOTIFY_QUEUE_80_PERCENT:

printf("\r\nDM_NOTIFY_CLASS_WARNING:DM_NOTIFY_QUEUE_80_PERCENT");
            break;
        case DM_NOTIFY_QUEUE_90_PERCENT:

printf("\r\nDM_NOTIFY_CLASS_WARNING:DM_NOTIFY_QUEUE_90_PERCENT");
            break;
        case DM_NOTIFY_QUEUE_OVERFLOW:

printf("\r\nDM_NOTIFY_CLASS_WARNING:DM_NOTIFY_QUEUE_OVERFLOW");
            break;
        case DM_NOTIFY_CYCLES_CHANGED:

printf("\r\nDM_NOTIFY_CLASS_WARNING:DM_NOTIFY_CYCLES_CHANGED");
            bShowParams = TRUE;
            break;
        case DM_NOTIFY_MACHINES_CHANGED:

printf("\r\nDM_NOTIFY_CLASS_WARNING:DM_NOTIFY_MACHINES_CHANGED");
            bShowParams = TRUE;
            break;
        case DM_NOTIFY_PROJECT_OPENED:

printf("\r\nDM_NOTIFY_CLASS_WARNING:DM_NOTIFY_PROJECT_OPENED
[%s]",
                                (LPSTR)lpbyData);
            bSpecialActionRemoveNotify = TRUE;
            break;
        case DM_NOTIFY_PROJECT_CLOSED:
```

```
printf("\r\nDM_NOTIFY_CLASS_WARNING:DM_NOTIFY_PROJECT_CLOSE");
            bSpecialActionRemoveNotify = TRUE;
            break;
        case DM_NOTIFY_SYSTEM_LOCALE:

printf("\r\nDM_NOTIFY_CLASS_WARNING:DM_NOTIFY_SYSTEM_LOCALE
[%ld]",
                                *(DWORD*)lpbyData);
            break;
        case DM_NOTIFY_DATA_LOCALE:

printf("\r\nDM_NOTIFY_CLASS_WARNING:DM_NOTIFY_DATA_LOCALE [%ld]",
                                *(DWORD*)lpbyData);
            break;
        case DM_NOTIFY_PROJECT_RUNTIME:

printf("\r\nDM_NOTIFY_CLASS_WARNING:DM_NOTIFY_ROJECT_RUNTIME
[%s]",
                                (LPSTR)lpbyData);
            break;
        case DM_NOTIFY_PROJECT_EDIT:

printf("\r\nDM_NOTIFY_CLASS_WARNING:DM_NOTIFY_PROJECT_EDIT [%s]",
                               (LPSTR)lpbyData);
            bSpecialActionRemoveNotify = TRUE;
            break;
        case DM_NOTIFY_HOTKEY_CHANGE:

printf("\r\nDM_NOTIFY_CLASS_WARNING:DM_NOTIFY_HOTKEY_CHANGE");
            bShowParams = TRUE;
            break;
        case DM_NOTIFY_URSEL:
           printf("\r\nDM_NOTIFY_CLASS_WARNING:DM_NOTIFY_URSEL");
            bShowParams = TRUE;
            break;
        case DM_NOTIFY_BODO:
            printf("\r\nDM_NOTIFY_CLASS_WARNING:DM_NOTIFY_BODO");
            bShowParams = TRUE;
            break;
        case DM_NOTIFY_BEGIN_PROJECT_EDIT:

printf("\r\nDM_NOTIFY_CLASS_WARNING:DM_NOTIFY_BEGIN_PROJECT_EDIT
[%s]",
                               (LPSTR)lpbyData);
            bShowParams = TRUE;
            bSpecialActionRemoveNotify = TRUE;
            break;
        default:
            printf("\r\nunknown
DM_NOTIFY_CLASS_WARNING:dwNotifyCode[%08lx]", dwNotifyCode);
            bShowParams = TRUE;
```

```
        }
        break;
case DM_NOTIFY_CLASS_DATA:
        switch (dwNotifyCode)
        {
        case DM_NOTIFY_APPLICATION_DATA:

printf("\r\nDM_NOTIFY_CLASS_DATA:DM_NOTIFY_APPLICATION_DATA:");
            bShowParams = TRUE;
            break;
        case DM_NOTIFY_VARIABLE_DATA:

printf("\r\nDM_NOTIFY_CLASS_DATA:DM_NOTIFY_VARIABLE_DATA:");
            bShowParams = TRUE;
            break;
        case DM_NOTIFY_FIRE_DATA:
            // the data sended with DMFireNotifyData is char text
           printf("\r\nDM_NOTIFY_CLASS_DATA:DM_NOTIFY_FIRE_DATA:
(data as text: [%s])",
                                  (char*)lpbyData);
            bShowParams = TRUE;
            break;
        default:
            printf("\r\nunknown
DM_NOTIFY_CLASS_DATA:dwNotifyCode[%08lx]", dwNotifyCode);
            bShowParams = TRUE;
        }
        break;
default:
        printf("\r\nunknown dwNotifyClass[%08lx],
dwNotifyCode[%08lx]",
                        dwNotifyClass,
                        dwNotifyCode);
        bShowParams = TRUE;
}

if (bShowParams)
{
        printf("\r\ndwNotifyClass=%08lx,  dwNotifyCode=%08lx,
dwItems=%ld,  lpbyData=%08lx,  lpvUser=%08lx",
                        dwNotifyClass,
                        dwNotifyCode,
                        dwItems,
                        lpbyData,
                        (DWORD)lpvUser);
}

if (bSpecialActionRemoveNotify)
{
printf("\r\nHave to remove notify if RT exits:");
RemoveNotify();
}
```

```
printf("\r\n#### extra added Notify [%08lx] exit ####\r\n",
(DWORD)lpvUser);
return TRUE;
}
```

3.2.3.3 DMChangeDataLocale

使用

向通过 DMConnect 连接的所有应用程序提供运行系统语言的切换通知。

声明

```
BOOL DMChangeDataLocale (
    LPCTSTR        lpszProjectFile,
    DWORD          dwLocaleID,
    LPCMN_ERROR    lpdmError );
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

dwLocaleID

指向新选择语言的代码的指针。可能值为在文本库中组态的所有语言的代码。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

在切换语言时通知应用程序。

FALSE

错误。

注释

该应用程序使用以下参数通过回调函数 DM_NOTIFY_PROC (页 1912) 发出通知：

dwNotifyClass	DM_NOTIFY_CLASS_WARNING
dwNotifyCode	DM_NOTIFY_DATA_LOCALE
lpbyData	指向新组态语言的代码的指针。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_NOT_SUPPORTED	请求的服务不可用

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

AUTOHOTSPOT	连接到数据管理器

参见

DM_NOTIFY_PROC (页 1912)

DMEnumOpenedProjects (页 1938)

DMGetRuntimeProject (页 1945)

3.2.3.4 DMConnect

使用

建立应用程序到数据管理器的连接。在每个应用程序（过程）中仅可执行一次 DMConnect。如果多次调用，将返回错误消息 DM_E_ALREADY_CONNECTED。

声明

```
BOOL DMConnect (
    LPTSTR              lpszAppName,
    DM_NOTIFY_PROC      lpfnNotify,
    LPVOID              lpvUser,
    LPCMN_ERROR         lpdmError );
```

参数

lpszAppName

指向调用应用程序名称的指针。由于该参数用于内部标识，可选择任意名称。

应用程序名称的长度限制为 MAX_DM_APP_NAME （32 个字符）。名称过长会导致嵌入式 OHIOIPC.DLL 出错和函数终止，同时产生错误 DM_E_NOT_CONNECTED。

lpfnNotify

指向管理消息（从数据管理器到应用程序）的通知函数的指针。

当程序报告通知例程时，必须按照指定的时间间隔清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

在极少数情况下，在函数调用返回之前已经传送了“通知”。

“通知”例程在 DMDisConnect 之后应该仍可用，因为最近的调用可能到达。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中该指针会重新可用。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

设置到数据管理器的连接。

FALSE

错误。

注释

在支持多线程的操作系统平台上执行的 WinCC 版本中，通知函数的调用与调用过程异步执行。

您需要在应用程序中组态适当的同步。

如果数据管理器函数分布给程序中的多个线程，则在每个线程中运行 DMConnect，并在结束时运行 DMDisconnet。

首先，使用 DMGetConnectionState 检查是否有其他线程已建立了连接。如果连接已存在，则可能无法执行 DMConnect 或 DMDisConnect。
要避免冲突，可将函数调用与单线程中的数据管理器相结合，并只在此处执行调用。

如果数据管理器函数被换出至具有数据管理器函数的应用程序中的嵌入式 DLL，则检查是否可以通过 DLL 建立和终止连接。在此情况下，应用程序必须始终确保使用 DMGetConnectionState 调用其自身数据管理器函数。但是，线程封装在 DLL 中时，这并不能保证连接不会断开。
因此，在标记中定义自己的连接建立，以便仅在需要的情况下执行 DMDisConnect。

返回 DM_E_ALREADY_CONNECTED 错误消息并具有通知功能的未检查 DMConnect 调用可能会取消先前调用的通知，并导致不符合要求的结果。

指向传递到回调函数的数据范围的指针仅在该函数内有效，也就是说，数据管理器会在回调返回后释放分配的存储区域。如果应用程序要求在调用时间之外访问数据，必须相应地复制数据。

另外，指向仅支持这一点的平台上的存储区的指针仅拥有读取权限，这样对于相关数据的写访问会导致保护故障！

当前未评估回调的返回值，该值仅在未来版本中有意义。默认设置下，应用程序在此返回 TRUE。

API DLL 不可直接用于 (ISS) 服务中，因为此处所需内部资源不可用。

如果在管理环境 (C#) 中使用数据管理器函数（如 wrapper），应确保应用程序中的多个应用程序域在同一过程中运行。因此，在所有应用域中仅允许进行一个 DMConnect。

错误消息

DM_E_ALREADY_CONNECTED	已建立到数据管理器的连接。
DM_E_NOT_CONNECTED	无法建立到数据管理器的连接（例如，应用程序名称过长等）。
DM_E_ACCESS_FAULT	无权建立到数据管理器的连接 （无效用户上下文，例如 ISS 服务等）。

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DMDisconnect (页 1916)	终止到数据管理器的连接
DMGetConnectionState (页 1929)	检查到数据管理器的连接
DM_NOTIFY_PROC (页 1912)	通知函数

示例

连接到 DM (页 2126) "DM01.cpp"

参见

DM_NOTIFY_PROC (页 1912)

DMDisconnect (页 1916)

DMGetConnectionState (页 1929)

连接到 DM (页 2126)

APConnect (页 2272)

3.2.3.5 DM_NOTIFY_PROC

说明

为使应用程序能够提供语言切换通知，您需要提供 DM_NOTIFY_PROC 类型的回调函数。

声明

```
BOOL ( * DM_NOTIFY_PROC) (
        DWORD     dwNotifyClass,
        DWORD     dwNotifyCode,
        LPBYTE    lpbyData,
        DWORD     dwItems,
        LPVOID    lpvUser);
```

参数

dwNotifyClass

标识通知类别：

DM_NOTIFY_CLASS_ERROR	(0x00000001)
DM_NOTIFY_CLASS_WARNING	(0x00000002)
DM_NOTIFY_CLASS_DATA	(0x00000003)

dwNotifyCode

通知代码

对于 DM_NOTIFY_CLASS_ERROR：		
DM_NOTIFY_SHUTDOWN	(0x00000001)	数据管理器已关闭
DM_NOTIFY_PROCESSNET_ERROR	(0x00000002)	过程总线出错
DM_NOTIFY_SYSNET_ERROR	(0x00000003)	系统总线出错
对于 DM_NOTIFY_CLASS_WARNING：		
DM_NOTIFY_QUEUE_50_PERCENT	(0x00000001)	应用程序队列填充水平为 50%
DM_NOTIFY_QUEUE_60_PERCENT	(0x00000002)	应用程序队列填充水平为 60%
DM_NOTIFY_QUEUE_70_PERCENT	(0x00000003)	应用程序队列填充水平为 70%
DM_NOTIFY_QUEUE_80_PERCENT	(0x00000004)	应用程序队列填充水平为 80%
DM_NOTIFY_QUEUE_90_PERCENT	(0x00000005)	应用程序队列填充水平为 90%
DM_NOTIFY_QUEUE_OVERFLOW	(0x00000006)	应用程序队列溢出

DM_NOTIFY_CYCLES_CHANGED	(0x00000010)	重新扫描更新周期
DM_NOTIFY_MACHINES_CHANGED	(0x00000011)	重新扫描计算机列表
DM_NOTIFY_PROJECT_OPENED	(0x00000012)	加载项目
DM_NOTIFY_PROJECT_CLOSE	(0x00000013)	关闭项目
DM_NOTIFY_SYSTEM_LOCALE	(0x00000014)	切换组态语言
DM_NOTIFY_DATA_LOCALE	(0x00000015)	切换 GUI 语言
DM_NOTIFY_PROJECT_RUNTIME	(0x00000016)	启用项目
DM_NOTIFY_PROJECT_EDIT	(0x00000017)	禁用项目
DM_NOTIFY_HOTKEY_CHANGE	(0x00000018)	热键已更改
对于 DM_NOTIFY_CLASS_DATA：		
DM_NOTIFY_APPLICATION_DATA	(0x00000001)	应用程序数据
DM_NOTIFY_VARIABLE_DATA	(0x00000002)	变量数据

有关通知代码中的更多信息，请参见常量说明。

lpbyData

指向在 DM_NOTIFY_CLASS_DATA 类别中提供的数据的指针。

dwItems

lpbyData 中的条目数。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

返回值视执行情况而定。

说明

如有可能，仅在此处复制数据。以下回调中的函数调用类型可导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

当程序报告通知例程时，必须按照指定的时间间隔清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

在极少数情况下，在 DMConnect 函数调用返回之前已经传送了“通知”。

所需文件

dmclient.h

相关函数

DMConnect (页 1909)	连接到数据管理器

示例

AUTOHOTSPOT "DM01.cpp"

参见

DMConnect (页 1909)

3.2.3.6 DMDeactivateRTProject

使用

禁用运行时模式下的项目。

声明

```
BOOL DMDeactivateRTProject (
    LPCMN_ERROR    lpdmError);
```

参数

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

项目已禁用。

FALSE

错误。

错误消息

DM_E_NO_RT_PRJ	不存在处于运行时模式的项目

所需文件

dmclient.h

dmclient.lib

dmclient.dll

示例

OnTestDeactivateRuntimeProject (页 2139) "TESTCDoc.cpp"

参见

OnTestDeactivateRuntimeProject (页 2139)

3.2.3.7 DMDisconnect

使用

应用程序使用此函数来确定到数据管理器的现有连接。

声明

```
BOOL DMDisConnect (
        LPCMN_ERROR    lpdmError );
```

参数

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

连接被终止。

FALSE

错误。

备注

如果未使用 DMConnect 函数建立到数据管理器的连接，则返回值为 FALSE。lpdmError->dwError 错误代码包含 DM_E_NOT_CONNECTED，值，未连接到数据管理器。

说明

不得在应用程序（EXE、DLL、OCX 等）的析构函数中使用该调用。由于 Microsoft 的特定机制，这可能导致调用冻结程序，从而导致程序崩溃。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DMConnect (页 1909)	连接到数据管理器

示例

连接到 DM (页 2126) "DM01.cpp"

参见

DMConnect (页 1909)

连接到 DM (页 2126)

3.2.3.8 DMEnumNumberFormats

使用

该函数列出了 FORMAT.DLL 中所有可用的格式转换。

声明

```
BOOL DMEnumNumberFormats (
    LPDWORD                 lpdwItems,
    DM_ENUM_FORMATS_PROC    lpfnFormat,
    LPVOID                  lpvUser,
    LPCMN_ERROR             lpdmError);
```

参数

lpdwItems

指向应用程序的双字的指针，双字包含调用后枚举格式数据的编号。

lpfnFormat

指向为每个可用数字格式调用的回调函数的指针。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

格式转换已列出。

FALSE

错误。

备注

传送到回调的指针仅在此函数中有效，因为一旦函数返回，系统就会释放所有已分配的存储区。如果应用程序要求在此函数之外访问数据，必须相应地复制数据。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DM_ENUM_FORMATS_PROC (页 1919)	列出格式转换（回调）

参见

DM_ENUM_FORMATS_PROC (页 1919)

3.2.3.9 DM_ENUM_FORMATS_PROC

说明

为了评估系统列出的格式转换，必须提供 DM_ENUM_FORMATS_PROC 类型的回调函数。

声明

```
BOOL ( * DM_ENUM_FORMATS_PROC) (
    LPDM_FORMAT_INFO    lpdmFormat,
    DWORD               dwItem,
    LPVOID              lpvUser );
```

参数

lpdmFormat

指向具有格式转换数据的 DM_FORMAT_INFO (页 1845) 类型结构的指针。

dwItem

连续调用计数器。如果没有提前取消枚举，dwItem 将包含可用格式转换的数量。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

说明

如有可能，仅在此处复制数据。以下回调中的函数调用类型可导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

dmclient.h

相关函数

DMEnumNumberFormats (页 1918)	列出格式转换

参见

DMEnumNumberFormats (页 1918)

DM_FORMAT_INFO (页 1845)

3.2.3.10 DMEnumUpdateCycles

使用

该函数列出了系统中定义的所有更新周期。每次调用回调函数时，该函数为周期传送信息结构。

声明

```
BOOL DMEnumUpdateCycles (
        LPCSTR                  lpszProjectFile,
        LPDWORD                 lpdwItems,
        DM_ENUM_CYCLES_PROC     lpfnCycle,
         LPVOID                  lpvUser,
        LPCMN_ERROR             lpdmError);
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

lpdwItem

指向应用程序的双字的指针，双字包含调用后枚举周期数据的编号。

lpfnCycle

指向为每个可用更新周期调用的回调函数的指针。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中该指针会重新可用。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

更新周期已列出。

FALSE

错误

备注

传送到回调函数的指针仅在此函数中有效，因为一旦函数返回，系统就会释放所有已分配的存储区。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DM_ENUM_CYCLES_PROC (页 1923)	列出更新周期（回调）

示例

OnTestUpdateCycles (页 2150) "TESTCDoc.cpp"

参见

DM_ENUM_CYCLES_PROC (页 1923)

OnTestUpdateCycles (页 2150)

3.2.3.11 DM_ENUM_CYCLES_PROC

说明

为了评估系统列出的更新周期，必须提供 DM_ENUM_CYCLES_PROC 类型的回调函数。

声明

```
BOOL ( * DM_ENUM_CYCLES_PROC) (
    LPDM_CYCLE_INFO    lpdmCycle,
    DWORD              dwItem,
    LPVOID             lpvUser);
```

参数

lpdmCycle

指向具有更新周期数据的 DM_CYCLE_INFO (页 1841) 类型结构的指针。

dwItem

连续调用计数器。如果没有提前取消枚举，dwItem 将包含可用更新周期的数量。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

说明

如有可能，仅在此处复制数据。以下回调中的函数调用类型可导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

dmclient.h

相关函数

DMEnumUpdateCycles (页 1921)	列出更新周期

参见

DMEnumUpdateCycles (页 1921)

DM_CYCLE_INFO (页 1841)

3.2.3.12 DMExitWinCC

使用

可以使用该函数关闭 WinCC。

声明

```
BOOL DMExitWinCC (
    VOID);
```

参数

无

备注

也可以使用 DMExitWinCCEx 指定操作系统应该如何响应 WinCC 的关闭。

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DMExitWinCCEx (页 1925)	通过默认设置关闭 WinCC

示例

OnTestWinCCShutdown (页 2168) "TESTCDoc.cpp"

参见

OnTestWinCCShutdown (页 2168)

DMExitWinCCEx (页 1925)

3.2.3.13 DMExitWinCCEx

使用

可以使用该函数关闭 WinCC。也可以指定 WinCC 关闭后操作系统如何响应。

声明

```
BOOL DMExitWinCCEx (
    DWORD    dwMode );
```

参数

dwMode

可以使用 dwMode 指定在 WinCC 关闭后操作系统的各种响应方式。

DM_SDMODE_WINCC	WinCC 已关闭
DM_SDMODE_LOGOFF	当 WinCC 关闭时，用户也从操作系统注销。
DM_SDMODE_SYSTEM	当 WinCC 关闭时，操作系统关闭。
DM_SDMODE_REBOOT	当 WinCC 关闭时，操作系统重新启动。

备注

使用 DM_SDMODE_WINCC 参数时，该函数与 DMExitWinCC 具有同样的效果。

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DMExitWinCC (页 1924)	关闭 WinCC

参见

DMExitWinCC (页 1924)

3.2.3.14 DMFireNotifyData

声明

```
BOOL DMFireNotifyData (
        DWORD            dwNotifyCookie,
        DWORD            dwByteCount,
        LPBYTE           lpbyData,
        LPCMN_ERROR      lpdmError);
```

说明

使用特定的 cookie 标识符将附加通知发送到已连接的通知。

参数

dwNotifyCookie

使用 DMAddNotify 连接的附加通知函数的 Cookie。

如果指定“0”，则会将通知数据发送给所有活动通知（广播）。

dwByteCount

要发送的数据缓冲区大小（字节）。

lpbyData

包含要发送的通知数据的字节缓冲区指针。

缓冲区的大小由 dwByteCount 指定。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

通知数据已发送。

FALSE

错误

注释

已发送通知在 dwNotifyClass=DM_NOTIFY_CLASS_DATA 和 dwNotifyCode=DM_NOTIFY_FIRE_DATA 下所述的通知例程中编辑。

如果未预先删除附加通知函数就执行 DMDisconnect，则会再次给所有这些通知函数发出最终 DM_NOTIFY_SHUTDOWN，随后使用管理功能从内部删除这些函数。当这些函数不再有效后，便可丢弃相应连接和 cookie。

错误消息

DM_E_NOT_CONNECTED	尚未执行 DMConnect 或者已执行 DMDisconnect
DM_E_DONT_EXIST	指定 cookie 不存在任何通知
DM_E_PARAM	参数错误

所需文件

dmclient_exstr.h

dmclient.lib

dmclient.dll

相关函数

DMAddNotify	将附加通知函数链接到“DMClient”
DMFireNotify	使用 cookie 从通知列表中删除附加通知函数

示例

按钮 DMFireNotifyData：

```
#include "apdefap.h"
void OnClick(char* lpszPictureName,
             char* lpszObjectName,
             char* lpszPropertyName)
{
DWORD dwNotifyCookie;
CHAR* pszNotifyText;
CMN_ERROR err;
BOOL bRet;

dwNotifyCookie = GetTagDWord("DMdwNotifyCookie");
```

```
pszNotifyText = GetTagChar("DMNotifySendText");
memset(&err, 0, sizeof(err));
bRet = FALSE;

bRet = DMFireNotifyData(dwNotifyCookie,
                        strlen(pszNotifyText)*sizeof(CHAR)+1,
                        (LPBYTE)pszNotifyText, &err);
if (FALSE == bRet)
{
printf("\r\nERROR:DMFireNotifyData [%s],%ld,%ld,%ld,%ld,%ld",
       err.szErrorText,
       err.dwError1,
       err.dwError2,
       err.dwError3,
       err.dwError4,
       err.dwError5);
}

}
```

3.2.3.15 DMGetConnectionState

使用

可使用此函数查询到数据管理器的连接，例如，检查是否正确执行了 DMConnect 函数调用。

声明

```
BOOL DMGetConnectionState (
    LPCMN_ERROR    lpdmError);
```

参数

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

连接已建立。

FALSE

错误或无连接。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DMConnect (页 1909)	连接到数据管理器

示例

连接到 DM (页 2126) "DM01.cpp"

参见

DMConnect (页 1909)

连接到 DM (页 2126)

3.2.3.16 DMGetDataLocale

使用

该函数返回当前选择的运行系统语言的 ID。

声明

```
BOOL DMGetDataLocale (
    LPDWORD        lpdwLocaleID,
    LPCMN_ERROR    lpdmError);
```

参数

lpdwLocaleID

指向当前组态的语言代码的指针。返回值为在文本库中组态的所有语言的代码。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

获取语言 ID。

FALSE

错误。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器

所需文件

dmclient.h

dmclient.lib

dmclient.dll

3.2.3.17 DMGetHotkey

使用

该函数确定与热键关联的操作。

声明

```
BOOL DMGetHotKey (
        DWORD          dwHotKeyAction,
        LPDWORD        lpdwHotKey,
        LPCMN_ERROR    lpdmError);
```

参数

dwHotKeyAction

动作

lpdwHotKey

指向存储热键的存储单元的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

获取热键 ID。

FALSE

错误。

所需文件

dmclient.h

dmclient.lib

dmclient.dll

3.2.3.18 DMGetMachineInfo

使用

应用程序使用该函数确定特定计算机的启动参数。要实现该功能，应用程序必须具有事先通过项目管理函数创建的计算机属性的相关条目。

声明

```
BOOL DMGetMachineInfo (
    LPCSTR          lpszLogicalName,
    LPVOID          lpvData,
    LPDWORD         lpdwSize,
    LPCMN_ERROR     lpdmError);
```

参数

lpszLogicalName

指向 Explorer DLL（查询其启动参数）的逻辑名称的指针。

lpvData

指向传送到应用程序的启动参数数据的数据范围的指针。

lpdwSize

如果 lpvData = 0，则在 lpdwSize 中指定数据范围的大小。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

获取关于本地 PC 的信息。

FALSE

错误。

错误消息

DM_E_NO_RT_PRJ	不存在处于运行时模式的项目
DM_E_MACHINE_NOT_FOUND	无法找到 PC
DM_E_NO_INFO_FOUND	无法找到启动信息

所需文件

dmclient.h

dmclient.lib

dmclient.dll

3.2.3.19 DMGetMachineTable

使用

除项目中涉及的 PC 数量之外，还可以使用该函数确定有关单个 PC 的信息。

声明

```
BOOL DMGetMachineTable (
    LPCSTR                lpszProjectFile,
    LPDM_MACHINE_TABLE    lpdmMachineTable,
    LPCMN_ERROR           lpdmError);
```

参数

lpdwProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

lpdmMachineTable

指向其中存储了计算机列表数据的 DM_MACHINE_TABLE (页 1846) 结构的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已确定计算机列表的数据。

FALSE

错误。

所需文件

dmclient.h

dmclient.lib

dmclient.dll

示例

OnTestMachines (页 2143) "TESTCDoc.cpp"

参见

DM_MACHINE_TABLE (页 1846)

OnTestMachines (页 2143)

DMActivateRTProject (页 1899)

3.2.3.20 DMRemoveNotify

声明

```
BOOL DMRemoveNotify (
    DWORD            dwNotifyCookie,
    LPCMN_ERROR      lpdmError);
```

说明

使用相关 cookie 标识符从附加通知列表中删除已连接的附加通知函数。

此函数仅在 WinCC V7.2 或更高版本中可用。

参数

dwNotifyCookie

使用 DMAddNotify 连接的附加通知函数的 Cookie。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已删除附加通知函数。

FALSE

错误

注释

如果未预先删除附加通知函数就执行 DMDisconnect，则会再次给所有这些通知函数发出最终 DM_NOTIFY_SHUTDOWN，随后使用管理功能从内部删除这些函数。当这些函数不再有效后，便可丢弃相应连接和 cookie。

错误消息

DM_E_NOT_CONNECTED	尚未执行 DMConnect 或者已执行 DMDisconnect
DM_E_DONT_EXIST	指定 cookie 不存在任何通知
DM_E_PARAM	参数错误

所需文件

dmclient_exstr.h

dmclient.lib

dmclient.dll

相关函数

DMFireNotifyData	使用 cookie 将数据发送到附加通知函数或所有函数
DMAddNotify	将附加通知函数链接到“DMClient”

示例

按钮 RemoveNotify:

```
#include "apdefap.h"
void OnClick(char* lpszPictureName,
             char* lpszObjectName,
             char* lpszPropertyName)
{
RemoveNotify();
}
```

项目函数 RemoveNotify:

```
void RemoveNotify()
{
DWORD dwNotifyCookie = 0L;
CMN_ERRORA err;
BOOL bRet = FALSE;

memset(&err, 0, sizeof(CMN_ERRORA));
dwNotifyCookie = GetTagDWord("DMdwNotifyCookie");

if (dwNotifyCookie)
{
            bRet = DMRemoveNotify(dwNotifyCookie, &err);
            if (bRet)
            {
            printf("\r\nNotify [%08lx] removed", dwNotifyCookie);
            dwNotifyCookie = 0L;
            SetTagDWord("DMdwNotifyCookie",dwNotifyCookie);
            }
            else
            {
                printf("\r\nERROR:DMRemoveNotifyA
[%s],%ld,%ld,%ld,%ld,%ld, Cookie=%08lx",
                                err.szErrorText,
                 err.dwError1,
                 err.dwError2,
                 err.dwError3,
                 err.dwError4,
                 err.dwError5,
                 dwNotifyCookie);
            }
}
```

```
else
{
            printf("\r\nNo Notify present to remove!");
}
}
```

3.2.4 项目管理函数

3.2.4.1 DMEnumOpenedProjects

使用

为在 WinCC 中打开的每个项目调用传送的回调函数。

声明

```
BOOL DMEnumOpenedProjects (
    LPDWORD                          lpdwItems,
    DM_ENUM_OPENED_PROJECTS_PROC     lpfnEnum,
    LPVOID                           lpvUser,
    LPCMN_ERROR                      lpdmError);
```

参数

lpdwItems

指向双字的指针，双字包含完成枚举时打开项目的数量。

lpfnEnum

指向回调函数的指针，回调函数接收项目数据。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

列出项目。

FALSE

错误。

备注

当前，在 WinCC 中仅可以打开一个项目。因此仅能获取关于该项目的信息。

错误消息

DM_E_FILE	文件操作错误

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DM_ENUM_OPENED_PROJECTS_PROC (页 1940)	列出打开的项目（回调）

示例

枚举打开项目 (页 2132) "DM01.cpp"

OnTestOpenProjects (页 2147) "TESTCDoc.cpp"

参见

DM_ENUM_OPENED_PROJECTS_PROC (页 1940)

DM_PROJECT_INFO (页 1847)

枚举打开项目 (页 2132)

OnTestOpenProjects (页 2147)

DMChangeDataLocale (页 1908)

3.2.4.2 DM_ENUM_OPENED_PROJECTS_PROC

说明

为能够评估系统列出的项目，必须提供 DM_ENUM_OPENED_PROJECTS_PROC 类型的一个回调函数。

声明

```
BOOL ( * DM_ENUM_OPENED_PROJECTS_PROC) (
    LPDM_PROJECT_INFO    lpInfo,
    LPVOID                lpvUser);
```

参数

lpInfo

指向具有开放式项目相关信息的 DM_PROJECT_INFO (页 1847) 类型结构的指针。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

备注

当前，在 WinCC 中仅可以打开一个项目。因此仅能获取关于该项目的信息。

说明

如有可能，仅在此处复制数据。以下回调中的函数调用类型可导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

dmclient.h

相关函数

DMEnumOpenedProjects (页 1938)	列出项目（打开）

示例

Enum open projects "DM01.cpp"

参见

DMEnumOpenedProjects (页 1938)

DM_PROJECT_INFO (页 1847)

枚举打开项目 (页 2132)

3.2.4.3 DMGetProjectDirectory

使用

返回适用于应用程序的给定项目的组态数据的路径。

声明

```
BOOL DMGetProjectDirectory (
    LPCSTR                  lpszAppName,
    LPCSTR                  lpszProjectFile,
    LPDM_DIRECTORY_INFO     lpdmDirInfo,
    LPCMN_ERROR             lpdmError );
```

参数

lpszAppName

指向应用程序（将为其确定路径）的名称的指针。

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

可以通过 DMEnumOpenedProjects 确定项目文件的名称。

lpdmDirInfo

指向其中存储了组态数据路径的 DM_DIRECTORY_INFO (页 1843) 结构的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

获取路径信息。

FALSE

错误。

错误消息

DM_E_FILE	文件操作错误

所需文件

dmclient.h

dmclient.lib

dmclient.dll

示例

OnTestProjectPaths (页 2145) "TESTCDoc.cpp"

参见

DM_DIRECTORY_INFO (页 1843)

OnTestProjectPaths (页 2145)

3.2.4.4 DMGetProjectInformation

使用

确定给定项目的相关信息，例如路径、数据源名称、计算机组态等。

声明

```
BOOL DMGetProjectInformation (
    LPCSTR               lpszProjectFile,
    LPDM_PROJECT_INFO    lpProjectInfo,
    LPCMN_ERROR          lpdmError);
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件的名称可使用 DMEnumOpenedProjects 或在 RT 中通过 DMGetRuntimeProject 确定。

lpProjectInfo

指向其中存储了项目信息的 DM_PROJECT_INFO (页 1847) 结构的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

获取项目信息

FALSE

错误

错误消息

DM_E_FILE	文件操作错误

所需文件

dmclient.h

dmclient.lib

dmclient.dll

示例

查询项目信息 (页 2138) "DM01.cpp"

OnTestProjectInfo (页 2144) "TESTCDoc.cpp"

参见

DM_PROJECT_INFO (页 1847)

查询项目信息 (页 2138)

OnTestProjectInfo (页 2144)

3.2.4.5 DMGetRuntimeProject

使用

返回处于在线模式的项目的文件名称。要在成功执行后确定所有其他项目数据，调用应用程序应记录在 lpszProjectFile 输入的项目文件名称。

声明

```
BOOL DMGetRuntimeProject (
    LPSTR          lpszProjectFile,
    DWORD          dwBufSize,
    LPCMN_ERROR    lpdmError);
```

参数

lpszProjectFile

用于存储项目文件名称（包括路径和扩展名）的缓冲区。缓冲区的大小必须至少为 _MAX_PATH 个字符。

dwBufSize

给定缓冲区的大小，以字符为单位。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已启用项目的名称已确定。

FALSE

错误。

备注

即使当运行系统已经启动但并未启动所有组件时，也可成功执行调用。

错误消息

DM_E_FILE	文件操作错误
DM_E_NO_RT_PRJ	不存在处于运行时模式的项目

所需文件

dmclient.h

dmclient.lib

dmclient.dll

示例

读取变量 (页 2172) "DM02.cpp"

写入变量 (页 2177) "DM02.cpp"

OnTestRuntimeProject (页 2148) "TESTCDoc.cpp"

参见

读取变量 (页 2172)

写入变量 (页 2177)

OnTestRuntimeProject (页 2148)

DMChangeDataLocale (页 1908)

3.2.4.6 DMOpenProjectDocPlus

使用

打开在 lpszProjectFile 中指定的项目。

声明

```
BOOL DMOpenProjectDocPlus (
    LPSTR          lpszProjectFile,
    LPCMN_ERROR    lpdmError);
```

参数

lpszProjectFile

指向要打开的项目文件名称（包括路径和扩展名）的指针。

WinCC 隐式生成的发布名称（例如 WinCC 50_Project_Odk）不能用作路径，因为它仅在此项目打开时存在。

项目文件名称通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

项目打开。

FALSE

错误。

所需文件

dmclient.h

dmclient.lib

dmclient.dll

3.2.4.7 DMOpenProjectPlus

使用

通过系统的标准选择对话框选择项目文件。如果该项目尚未在 WinCC 中打开，则加载该项目的所需信息。

当执行成功时，调用应用程序应记录在 lpszProjectFile 中输入的项目文件名称，因为该名称可用来确定关于项目的所有其他信息。

可以以后通过 DMEnumOpenedProjects 确定项目文件的名称。

声明

```
BOOL DMOpenProjectPlus (
    HWND           hwndParent,
    LPSTR          lpszProjectFile,
    DWORD          dwBufSize,
    LPCMN_ERROR    lpdmError );
```

参数

hwndParent

用作该对话框父窗口的窗口的句柄。

lpszProjectFile

指向用于存储项目文件名称（包括路径和扩展名）的缓冲区的指针。缓冲区的大小应至少为 _MAX_PATH 个字符。

当调用该函数时，lpszProjectFile 必须是空字符串或现有项目的有效路径。

如果 lpszProjectFile 是空字符串，则调用系统的默认选择对话框（参见上文）。

如果 lpszProjectFile 不为空，WinCC 将 lpszProjectFile 中包含的字符串解释为项目路径，并尝试在不打开选择对话框的情况下将其打开。

WinCC 隐式生成的发布名称（例如 WinCC 50_Project_Odk）不能用作项目路径，因为它仅在此项目打开时存在。

dwBufSize

给定缓冲区的大小，以字符为单位。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

单击“确定”(OK) 关闭对话框。

FALSE

出错或单击“取消”(CANCEL) 关闭对话框

备注

打开项目的过程可能需要几分钟时间。

错误消息

DM_E_CANCEL	用户已在对话框中选择“取消”(Cancel)
DM_E_FILE	文件操作错误
DM_E_UPDATE	更新项目

所需文件

dmclient.h

dmclient.lib

dmclient.dll

示例

以对话框方式打开项目 (页 2170) "DM01.cpp"

OnTestOpenProject (页 2146) "TESTCDoc.cpp"

参见

以对话框方式打开项目 (页 2170)

OnTestOpenProject (页 2146)

3.2.5 数据传输通道

3.2.5.1 DMClearBlockQueue

使用

删除应用程序的通知队列中的所有条目。

声明

```
BOOL DMClearBlockQueue (
    LPCMN_ERROR    lpdmError );
```

参数

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

通知队列已删除。

FALSE

错误。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器

所需文件

dmclient.h

dmclient.lib

dmclient.dll

3.2.5.2 DMEnumDataServices

使用

DMEnumDataServices 为每个已安装服务调用传送的回调函数，只要回调函数返回 TRUE。

声明

```
BOOL DMEnumDataServices (
    LPDWORD                     lpdwItems,
    DM_ENUM_DATA_SERVICE_PROC   lpfnEnum,
    LPVOID                      lpvUser,
    LPCMN_ERROR                 lpdmError);
```

参数

lpdwItems

指向应用程序的双字的指针，双字包含调用后列出的服务的数量。

lpfnEnum

指向为每个已安装服务调用的回调函数的指针。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

列出服务。

FALSE

错误。

注释

传送到回调的指针仅在此函数中有效，因为一旦函数返回，系统就会释放所有已分配的存储区。如果应用程序要求在此函数之外访问数据，必须相应地复制数据。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DM_ENUM_DATA_SERVICE_PROC (页 1952)	列出数据传输通道（回调）
DMInstallDataService (页 1957)	安装数据传输通道

参见

DM_ENUM_DATA_SERVICE_PROC (页 1952)

DMInstallDataService (页 1957)

3.2.5.3 DM_ENUM_DATA_SERVICE_PROC

说明

为了评估系统列出的已安装的数据传输通道，必须提供 DM_ENUM_DATA_SERVICE_PROC 类型的回调函数。

声明

```
BOOL ( * DM_ENUM_DATA_SERVICE_PROC) (
    LPCSTR                  pszService,
    DM_DATA_SERVICE_PROC    pfnService,
    LPVOID                  lpvUser );
```

参数

pszService

指向数据传输通道的逻辑名称的指针。

pfnService

指向 DM_DATA_SERVICE_PROC (页 1954) 类型回调函数的指针，可通过该类型回调函数提供服务的附加数据。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

说明

如有可能，仅在此处复制数据。以下回调中的函数调用类型可导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

dmclient.h

相关函数

DMEnumDataServices (页 1950)	列出数据传输通道

参见

DMEnumDataServices (页 1950)

DM_DATA_SERVICE_PROC (页 1954)

3.2.5.4 DM_DATA_SERVICE_PROC

说明

要使应用程序能接收数据，必须已通过 DMInstallDataService 安装数据传输通道并提供 DM_DATA_SERVICE_PROC 类型的回调函数。

如果要评估系统列出的传输通道（服务），也需要此类型的回调函数。

声明

```
BOOL ( * DM_DATA_SERVICE_PROC) (
    LPDM_DATA_SERVICE    lpds,
    LPVOID               lpvUser);
```

参数

lpds

指向具有安装数据传输通道（服务）相关信息的 DM_DATA_SERVICE (页 1842) 类型结构的指针。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

取决于回调函数的使用。

FALSE

错误

说明

如有可能，仅在此处复制数据。以下回调中的函数调用类型可导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

如果程序声明了“通知”例程，必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

在极少数情况下，在函数调用返回之前已经传送了“通知”。

所需文件

dmclient.h

相关函数

DMInstallDataService (页 1957)	安装数据传输通道
DM_ENUM_DATA_SERVICE_PROC (页 1952)	列出数据传输通道（回调）

参见

DM_ENUM_DATA_SERVICE_PROC (页 1952)

DM_DATA_SERVICE (页 1842)

DMInstallDataService (页 1957)

3.2.5.5 DMGetNumPendingBlocks

使用

应用程序使用该函数确定队列中的条目。

声明

```
BOOL DMGetNumPendingBlocks (
    LONG           *plEntries,
    LPCMN_ERROR    lpdmError);
```

参数

plEntries

成功执行后，该函数包含应用程序的队列中数据包的数量。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

队列中的条目数已确定。

FALSE

错误。

DM_E_NOT_CONNECTED	未连接到数据管理器

所需文件

dmclient.h

dmclient.lib

dmclient.dll

3.2.5.6 DMInstallDataService

使用

安装用于在应用程序之间传输数据的传输通道（服务）。该函数发出信号，指示应用程序已预备就绪，可接收由任何其他应用程序以指定的服务名称发送的数据。如果已经以指定的名称安装服务，则为此安装的回调函数会替换为传递的任何值。如果 lpfnService == NULL，则安装的服务 lpszService 被移除。

声明

```
BOOL DMInstallDataService (
    LPCSTR                  lpszService,
    DM_DATA_SERVICE_PROC    lpfnService,
    LPVOID                  lpvUser,
    LPCMN_ERROR             lpdmError);
```

参数

lpszService

指向服务名称的指针。通过该名称标识传输通道，名称是自由分配的。

lpfnService

指向回调函数的指针，通过该服务将数据发送到应用程序时将调用此函数。

当程序报告通知例程时，必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

在极少数情况下，在函数调用返回之前已经传送了“通知”。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中该指针会重新可用。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

数据传输通道已安装。

FALSE

错误。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DM_DATA_SERVICE_PROC (页 1954)	列出数据传输通道（回调），安装数据传输通道（回调）

参见

DMEnumDataServices (页 1950)

DM_DATA_SERVICE_PROC (页 1954)

3.2.5.7 DMSendApplicationData

使用

将 lpdmSendData 中传送的数据发送到所有应用程序，这些应用程序必须满足在 DM_SEND_DATA_STRUCT 结构中指定的条件，并且已经使用 DMInstallDataService 安装了 szService 下描述的服务。

声明

```
BOOL DMSendApplicationData (
    LPDM_SEND_DATA_STRUCT    lpdmSendData,
    LPCMN_ERROR              lpdmError);
```

参数

lpdmSendData

指向 DM_SEND_DATA_STRUCT (页 1848) 结构的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

数据已发送到应用程序。

FALSE

错误。

注释

该函数仅本地执行。
在 DM_SEND_DATA_STRUCT 结构中预留了用于远程访问"TargetMachine"的成员，以供将来使用。
只能通过 DM_SD_LOCAL 指定 dwTargetMachineFlags，任何其他指定将导致错误并且 dwTargetMachines 必须为 0L。

所需文件

dmclient.h

dmclient.lib

dmclient.dll

参见

DM_SEND_DATA_STRUCT (页 1848)

3.2.5.8 DMSetBlockQueueSize

使用

该函数指定在应用程序的通知队列中可保留的最大消息数。

声明

```
BOOL DMSetBlockQueueSize (
    LONG           nCount,
    LPCMN_ERROR    lpdmError);
```

参数

nCount

最大消息数。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

队列集的大小。

FALSE

错误。

注释

32 位平台的默认值是 1000 个条目，这通常足够了。

每条消息占用应用程序的地址空间。增加消息数量之前请仔细考虑，因为这将降低应用程序的处理速度。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器

所需文件

dmclient.h

dmclient.lib

dmclient.dll

3.2.6 用于处理变量的函数

3.2.6.1 DMEnumVarData

使用

该函数用于确定与变量关联的对象的信息，例如，关联的变量组、通道单元等。

声明

```
BOOL DMEnumVarData (
    LPCSTR                  lpszProjectFile,
    LPDM_VARKEY             lpdmVarKey,
    DWORD                   dwItems,
    DM_ENUM_VARIABLE_PROC   lpfnEnum,
    LPVOID                  lpvUser,
    LPCMN_ERROR             lpdmError);
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。
项目文件名称通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

lpdmVarKey

指向 DM_VARKEY (页 1869) 类型结构的字段起始位置的指针，可通过该类型结构指定要列出的变量。从 V5.0 SP2 开始，如果不指定数组并且为该参数提供 NULL，则枚举所有变量。

如果为 DM_VARKEY 指定错误名称或错误 ID，将不会显示错误，因为列表中一些值可能有效，而一些值可能错误。为使用户能够识别出错误的 DM_VARKEY，会在该回调中针对这一点传递 DM_VARIABLE_DATA (页 1863) 类型结构（使用 0 填充）。

dwItems

列出其数据的变量的数量（= 在 lpdmVarKey 中指定的变量数量）。如果将该值设为 0，则枚举所有变量。

lpfnEnum

指向回调的指针，该回调接收关于变量的信息。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

关于变量的信息已列出。

FALSE

错误

注释

有一个高级函数 DMEnumVarData4。

说明

DMEnumVariables 多级枚举的现有实现（通过为每个元素调用回调中的 DMEnumVarDataX）应该替换为 DMEnumVarData4 的单个调用。应该在回调本身中执行任何必要的过滤。此更改可显著地提高性能。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_INVALID_KEY	未找到变量

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DM_ENUM_VARIABLE_PROC (页 1963)	列出变量的相关信息（回调）

参见

DM_VARKEY (页 1869)

DM_VARIABLE_DATA (页 1863)

DM_ENUM_VARIABLE_PROC (页 1963)

3.2.6.2 DM_ENUM_VARIABLE_PROC

说明

为了评估系统列出的关于变量的信息，必须提供 DM_ENUM_VARIABLE_PROC 类型的回调函数。

声明

```
BOOL ( * DM_ENUM_VARIABLE_PROC) (
    LPDM_VARKEY           lpdmVarKey,
    LPDM_VARIABLE_DATA    lpdmVarData,
    LPVOID                lpvUser
```

参数

lpdmVarKey

指向用于指定将列出变量的 DM_VARKEY (页 1869) 类型结构的指针。

lpdmVarData

指向具有变量相关信息的 DM_VARIABLE_DATA (页 1863) 类型结构的指针。

当该结构完全用 0 填充时，在 DM_VARKEY 中会给出错误名称或错误 ID。 例如，可通过 lpdmVarData->dmTypeRef->dwType 进行查询。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举

说明

如有可能，仅在此处复制数据。以下回调中的函数调用类型可导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

dmclient.h

相关函数

DMEnumVarData (页 1961)	列出关于变量的信息

参见

DM_VARKEY (页 1869)

DM_VARIABLE_DATA (页 1863)

DMEnumVarData (页 1961)

3.2.6.3 DMEnumVarData4

使用

该函数用于确定与变量相关的对象的信息，例如，相关变量组和通道单元等。
该函数额外指定了标定信息，因此与 DMEnumVarData 不同。

声明

```
BOOL DMEnumVarData4 (
    LPCSTR                    lpszProjectFile,
    LPDM_VARKEY               lpdmVarKey,
    DWORD                     dwItems,
    DM_ENUM_VARIABLE_PROC4    lpfnEnum,
    LPVOID                    lpvUser,
    LPCMN_ERROR               lpdmError);
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

lpdmVarKey

指向 DM_VARKEY (页 1869) 类型结构的字段起始位置的指针，可通过该类型结构指定要列出的变量。如果将该参数设为 NULL，则枚举所有变量。

如果为 DM_VARKEY 指定错误名称或错误 ID，将不会显示错误，因为列表中一些值可能有效，而一些值可能错误。为使用户能够识别出错误的 DM_VARKEY，会在该回调中针对这一点通过 DM_VARIABLE_DATA4 (页 1866) 类型结构（使用 0 填充）。

dwItems

应列出其数据的变量的数量（= 在 lpdmVarKey 中指定的变量数量）。如果将该值设为 0，则枚举所有变量。

lpfnEnum

指向回调的指针，该回调接收关于变量的信息。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

关于变量的信息已列出。

FALSE

错误。

注释

说明

DMEnumVariables 多级枚举的现有实现（通过为每个元素调用回调中的 DMEnumVarDataX）应该替换为 DMEnumVarData4 的单个调用。应该在回调本身中执行任何必要的过滤。此更改可显著地提高性能。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_INVALID_KEY	未找到变量

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DM_ENUM_VARIABLE_PROC4 (页 1967)	列出变量的相关信息（回调）

示例

变量的枚举数据 (页 2130) "DM01.cpp"

参见

DM_VARKEY (页 1869)

DM_VARIABLE_DATA4 (页 1866)

DM_ENUM_VARIABLE_PROC4 (页 1967)

变量的枚举数据 (页 2130)

3.2.6.4 DM_ENUM_VARIABLE_PROC4

说明

为了评估系统列出的关于变量的信息，必须提供 DM_ENUM_VARIABLE_PROC4 类型的回调函数。该回调函数额外指定了标定信息，因此与 DM_ENUM_VARIABLE_PROC 不同。

声明

```
BOOL ( * DM_ENUM_VARIABLE_PROC4) (
    LPDM_VARKEY             lpdmVarKey,
    LPDM_VARIABLE_DATA4     lpdmVarData,
    LPVOID                  lpvUser
```

参数

lpdmVarKey

指向用于指定将列出变量的 DM_VARKEY (页 1869) 类型结构的指针。

lpdmVarData

指向具有变量相关信息的 DM_VARIABLE_DATA4 (页 1866) 类型结构的指针。

当该结构完全用 0 填充时，在 DM_VARKEY 中会给出错误名称或错误 ID。 例如，可通过 lpdmVarData->dmTypeRef->dwType 进行查询。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

dmclient.h

相关函数

DMEnumVarData4 (页 1965)	列出关于变量的信息

示例

变量的枚举数据 (页 2130) "DM01.cpp"

参见

DMEnumVarData4 (页 1965)

DM_VARKEY (页 1869)

DM_VARIABLE_DATA4 (页 1866)

变量的枚举数据 (页 2130)

3.2.6.5 DMEnumVarGrpData

使用

该函数通过 DM_ENUM_VARGRP_PROC 回调函数返回有关变量组的信息，例如，名称、创建者 ID 和组中变量的数量。

声明

```
BOOL DMEnumVarGrpData (
    LPSTR                   lpszProjectFile,
    LPDM_VARGRPKEY          lpdmVarGrpKey,
    DWORD                   dwItems,
    DM_ENUM_VARGRP_PROC     lpfnEnum,
    LPVOID                  lpvUser,
    LPCMN_ERROR             lpdmError);
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

lpdmVarGrpKey

指向第一个 DM_VARGRPKEY (页 1862) 类型结构的指针，可通过该类型结构指定变量组。

dwItems

要列出其数据的变量的数量。

lpfnEnum

指向回调的指针，该回调接收关于变量组的信息。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

关于变量组的信息已列出。

FALSE

错误。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_INVALID_KEY	未找到变量

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DM_ENUM_VARGRP_PROC (页 1973)	列出变量组的相关信息（回调）

示例

OnTestEnumGroupsAll (页 2140) "TESTCDoc.cpp"

参见

DM_VARGRPKEY (页 1862)

DM_ENUM_VARGRP_PROC (页 1973)

OnTestEnumGroupsAll (页 2140)

3.2.6.6 DMEnumVarGrpDataExStr

声明

```
BOOL DMEnumVarGrpDataExStr (
    LPTSTR                      lpszProjectFile,
    DWORD                       dwFlags,
    LPVARIANT                   lpvdmVarGrpKey,
    LPDWORD                     lpdwVarGrpCount,
    DM_ENUM_VARGRP_PROC_EXSTR   lpfnEnum,
    LPVOID                      lpvUser,
    LPCMN_ERROR                 lpdmError);
```

说明

该函数通过 DM_ENUM_VARGRP_PROC_EXSTR 回调函数返回变量组信息，例如，名称、ID 和组中变量的数量。

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称可在运行系统中通过 DMEnumOpenedProjects 或 DMGetRuntimeProject 检索。

dwFlags

保留供以后使用。用 0L 填充。

lpvdmVarGrpKey

指向指定变量组的第一个"DM_VARGRPKEY"类型结构的指针。

指向此变量组列表的 VARIANT 的指针。必须创建 VT_ARRAY | VT_VARIANT 类型的列表，因为每个变量可能采用不同的数据类型，例如 VT_I4 和 VT_BSTR，特定条件下还包括 VT_LPSTR。另外，如果只指定了一个关键字，也可以选择创建单个 VARIANT。

VT_EMPTY 或 NULL 指针用于枚举所有组。

lpdwVarGrpCount

指向返回变量组数量的 DWORD 指针。

lpfnCallback = NULL 用于确定变量组最初所需的存储区大小。

lpfnEnum

指向接收变量组信息的回调函数的指针。

lpvUser

指向应用程序特定数据的指针。此函数不会评估该指针，但会使其在回调函数中重新可用。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已枚举变量组信息

FALSE

错误

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_INVALID_KEY	未找到变量

所需文件

dmclient_exstr.h

dmclient.lib

dmclient.dll

相关函数

DM_ENUM_VARGRP_PROC_EXSTR	枚举变量组信息（回调）

3.2.6.7 DM_ENUM_VARGRP_PROC

说明

为了评估系统列出的关于变量组的信息，必须提供 DM_ENUM_VARGRP_PROC 类型的回调函数。

声明

```
BOOL ( * DM_ENUM_VARGRP_PROC) (
    LPDM_VARGRP_DATA    lpdmVarGrpData,
    LPVOID              lpvUser );
```

参数

lpdmVarGrpData

指向具有变量组相关信息的 DM_VARGRP_DATA (页 1861) 类型结构的指针。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

dmclient.h

相关函数

DMEnumVarGrpData (页 1969)	列出变量组的相关信息

参见

DMEnumVarGrpData (页 1969)

DM_VARGRP_DATA (页 1861)

3.2.6.8 DMEnumVariables

使用

该函数用于列出与特定选择标准相符的所有变量的名称。通过 DM_VARFILTER 结构确定选择标准。

声明

```
BOOL DMEnumVariables (
    LPCSTR              lpszProjectFile,
    LPDM_VARFILTER      lpdmVarFilter,
    DM_ENUM_VAR_PROC    lpfnEnum,
    LPVOID              lpvUser,
    LPCMN_ERROR         lpdmError);
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。
项目文件名称通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

lpdmVarFilter

指向其中指定了选择标准的 DM_VARFILTER (页 1859) 结构的指针。

lpfnEnum

指向回调函数的指针，回调函数接收变量名称。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

变量名已列出。

FALSE

错误。

注释

说明

DMEnumVariables 多级枚举的现有实现（通过为每个元素调用回调中的 DMEnumVarDataX）应该替换为 DMEnumVarData4 的单个调用。应该在回调本身中执行任何必要的过滤。此更改可显著地提高性能。

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DM_ENUM_VAR_PROC (页 1976)	列出变量名（回调）

示例

OnTestEnumVariables (页 2141) "TESTCDoc.cpp"

参见

DM_VARFILTER (页 1859)

DM_ENUM_VAR_PROC (页 1976)

OnTestEnumVariables (页 2141)

3.2.6.9 DM_ENUM_VAR_PROC

说明

为了评估系统列出的变量的名称，必须提供 DM_ENUM_VAR_PROC 类型的回调函数。

声明

```
BOOL ( * DM_ENUM_VAR_PROC) (
    LPDM_VARKEY    lpdmVarKey,
    LPVOID                lpvUser
```

参数

lpdmVarKey

指向具有变量名称的 DM_VARKEY (页 1869) 类型结构的指针。

lpvUser

指向应用程序特定数据的指针。 该指针在回调函数中重新可用。

返回值

TRUE

附加回调调用很适合。

FALSE

回调系列应该取消。

注释

注意
不能在此回调中调用任何其它 DMClient 函数。 否则可能会出现堵塞现象。

说明

仅应在这里复制数据（如可能）。 回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如： GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

dmclient.h

相关函数

DMEnumVariables (页 1974)	列出变量名

参见

DM_VARKEY (页 1869)

DMEnumVariables (页 1974)

3.2.6.10 DMGetValue

使用

从数据管理器的过程映像中读取一个或更多变量的值。读出值为最近更新时存在的一个值。

更高级的 DMGetValueEx 函数还另外返回 DM_VAR_UPDATE_STRUCTEX 结构中的质量代码。

声明

```
BOOL DMGetValue (
    LPDM_VARKEY                lpdmVarKey,
    DWORD                      dwItems,
    LPDM_VAR_UPDATE_STRUCT     lpdmvus,
    LPCMN_ERROR                lpdmError);
```

参数

lpdmVarKey

指向可识别要读取的变量值的 DM_VARKEY (页 1869) 结构指针。

dwItems

所传递结构的数量（与将要读取的变量值的数量相对应）。

lpdmvus

指向包含变量值的第一个 DM_VAR_UPDATE_STRUCT (页 1854) 结构的指针（函数返回后）。

lpdmError

指向第一个具有 CMN_ERROR 类型的 dwItems 错误结构的指针。在写入变量过程中出现错误时，系统会将错误信息写入适当的结构中。因此，请记得要为这些结构预留空间。

返回值

TRUE

值已确定。

FALSE

错误。

注释

要使应用程序能够执行单独的周期管理，可指定 ZERO 指针作为 DMStartVarUpdate 的回调函数。之后，数据管理器会根据要求的周期更新变量。不过，应用程序仍会负责对从过程映像中读取的值进行计时。

由于变量的更新请求可能由多个应用程序使用不同周期发出，因此数据管理器始终会基于最短请求周期来更新其过程映像。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_PARAM	无效参数
DM_E_OOM	内存不足

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DMGetValueWait (页 1991)	获取更新的变量值
DMSetValue (页 2024)	更改变量值

示例

读取变量 (页 2172) "DM02.cpp"

OnTestVariablenGetvalue (页 2153) "TESTCDoc.cpp"

参见

DM_VARKEY (页 1869)

DM_VAR_UPDATE_STRUCT (页 1854)

DMGetValueWait (页 1991)

DMSetValue (页 2024)

读取变量 (页 2172)

OnTestVariablenGetvalue (页 2153)

3.2.6.11 DMGetValueEx

使用

从数据管理器的过程映像中读取一个或更多变量的值。读出值为最近更新时存在的一个值。

相较 DMGetValue 而言，该函数还会返回 DM_VAR_UPDATE_STRUCTEX 结构中的质量代码。

声明

```
BOOL DMGetValueEx (
    LPDM_VARKEY                lpdmVarKey,
    DWORD                      dwItems,
    LPDM_VAR_UPDATE_STRUCTEX   lpdmvus,
    LPCMN_ERROR                lpdmError);
```

参数

lpdmVarKey

指向可识别要读取的变量值的 DM_VARKEY (页 1869) 结构指针。

dwItems

所传递结构的数量（与将要读取的变量值的数量相对应）。

lpdmvus

指向包含变量值的第一个 DM_VAR_UPDATE_STRUCTEX (页 1856) 结构的指针（函数返回后）。

lpdmError

指向第一个具有 CMN_ERROR 类型的 dwItems 错误结构的指针。在写入变量过程中出现错误时，系统会将错误信息写入适当的结构中。因此，请记得要为这些结构预留空间。

返回值

TRUE

值已确定。

FALSE

错误。

注释

要使应用程序能够执行单独的周期管理，可指定 NULL 指针作为 DMStartVarUpdate 的回调函数。之后，数据管理器会根据要求的周期更新变量。不过，应用程序仍会负责对从过程映像中读取的值进行计时。

由于变量的更新请求可能由多个应用程序使用不同周期发出，因此数据管理器始终会基于最短请求周期来更新其过程映像。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_PARAM	无效参数
DM_E_OOM	内存不足

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DMGetValueWait (页 1991)	获取更新的变量值
DMSetValue (页 2024)	更改变量值

参见

DM_NOTIFY_VARIABLEEX_PROC (页 2119)

DM_VAR_UPDATE_STRUCTEX (页 1856)

DM_VARKEY (页 1869)

DMGetValueWait (页 1991)

DMSetValue (页 2024)

DMGetValueWaitEx (页 1994)

3.2.6.12 DMGetValueExStr

声明

```
BOOL DMGetValueExStr (
    DWORD                           dwFlags,
    LPVARIANT                       lpdmVarKey,
    LPDM_VAR_UPDATE_STRUCT_EXSTR    lpdmvus,
    DWORD                           dwdmvusCount,
    LPCMN_ERROR                     lpdmError);
```

说明

从数据管理器的过程映像中读取一个或更多变量的值。读出值为最近更新时存在的一个值。

与 DMGetValueEx 不同，将不再使用 DM_VARKEY 结构；因此，变量名称没有长度限制。

参数

dmFlags

如果需要以 VT_LPSTR 格式在 DM_NOTIFY_VARIABLE_PROC_EXSTR 的 DM_VAR_UPDATE_STRUCT_EXSTR 结构的 VARIANT 中返回变量名称，则可在此指定 DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR 标志。

lpvdmVarKey

指向变量列表的 VARIANT 的指针。必须创建 VT_ARRAY | VT_VARIANT 类型的列表，因为可能存在不同的数据类型，例如 VT_I4 和 VT_BSTR，某些情况下还包括 VT_LPSTR。这也适用于单个关键字。

lpdmvus

指向包含函数返回后的变量值的第一个 DM_VAR_UPDATE_STRUCT_EXSTR 结构的指针。

dwdmvusCount

传递的 DM_VAR_UPDATE_STRUCT_EXSTR 结构数（与要读取的变量值数相同）。该数量必须与 lpvdmVarKey 中的变体数组的大小一致。

lpdmError

指向第一个 CMN_ERROR 类型的"dwdmvusCount"错误结构的指针。写入变量出错时，系统会将错误信息写入相应的结构。因此，切记要为这些结构预留空间。

返回值

TRUE

值已确定

FALSE

错误

注释

要使应用程序能够管理自身的周期管理，可指定 NULL 指针作为 DMStartVarUpdate 的回调函数。之后，数据管理器会根据要求的周期更新变量。不过，应用程序仍会负责对从过程映像中读取的值进行计时。

由于变量的更新请求可能由多个应用程序使用不同周期发出，因此数据管理器始终会基于最短周期时间来更新其过程映像。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_PARAM	无效参数
DM_E_OOM	内存不足

只有第一个错误结构才会返回这些错误。

所需文件

dmclient_exstr.h

dmclient.lib

dmclient.dll

相关函数

DMGetValueWaitExStr	获取更新的变量值
DMSetValueExStr	更改变量值

示例

脚本示例“按钮 DMGetValueExStr/DMSetValueExStr|DMGetValueWaitExStr/DMSetValueWaitExStr”：

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
#pragma code("kernel32.dll")
#define CP_ACP 0
int MultiByteToWideChar(UINT CodePage, DWORD dwFlags, LPCSTR
lpMultiByteStr, int cbMultiByte, LPWSTR lpWideCharStr, int
cchWideChar);
int WideCharToMultiByte(UINT CodePage, DWORD dwFlags, LPCWSTR
lpWideCharStr, int cchWideChar, LPSTR lpMultiByteStr, int
cbMultiByte, LPCSTR lpDefaultChar, LPBOOL lpUsedDefaultChar);
#pragma code()

#pragma code ("OleAut32.dll")
//#include "OleAuto.h"
SAFEARRAY * SafeArrayCreateVector(VARTYPE vt, long lLbound,
unsigned int cElements );
HRESULT SafeArrayPtrOfIndex(SAFEARRAY FAR* psa, long FAR*
rgIndices, void HUGEP* FAR* ppvData );
HRESULT SafeArrayLock(SAFEARRAY FAR* psa);
HRESULT SafeArrayLock(SAFEARRAY FAR* psa);
HRESULT VariantChangeType( VARIANTARG FAR* pvargDest, VARIANTARG
FAR* pvarSrc, unsigned short wFlags, VARTYPE vt);
typedef WCHAR OLECHAR;
BSTR SysAllocString(OLECHAR FAR* sz);
HRESULT SysFreeString(BSTR bstr);
#pragma code()

extern BOOL DM_NotifyVariableProcExStr_GetValueWait(DWORD
dwTAID, LPDM_VAR_UPDATE_STRUCT_EXSTR lpdmvus, DWORD dwItems,
LPVOID lpvUser);
extern BOOL DM_CompletitionProc(DWORD dwTAID, LPDWORD
lpdmVarState, DWORD dwItems, LPVOID lpvUser);

VARIANT vVarKey;
VARIANT* pvElem;
VARIANT vdmValue;
SAFEARRAY* parrayKeys;
SAFEARRAY* parrayValues;
DWORD dwVal[4];
DWORD dwState[4];
DWORD dwMerk;
CMN_ERROR err;
CMN_ERROR errArray[4];
DM_VAR_UPDATE_STRUCT_EXSTR dmvus[4];
HRESULT hr;
long lInx;
```

```
BOOL bRet;
DWORD dwFlags;
DWORD* pdwVarState;
DWORD dwTAID;

memset(&err, 0, sizeof(err));
memset(dmvus, 0, sizeof(dmvus));
dwVal[0] = 1;
dwVal[1] = 2;
dwVal[2] = 3;
dwVal[3] = 4;
dwMerk = 0L;
memset(dwState, 0, sizeof(dwState));
pvElem = NULL;
parrayKeys = NULL;
parrayValues = NULL;
hr = 0L; //S_OK
lInx = 0L;
bRet = FALSE;
dwFlags = DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR;
pdwVarState = dwState;
dwTAID = 0L;

printf("\r\n\r\n########## enter Test with DMGetValue(Wait)ExStr/
DMSetValue(Wait)ExStr ##########");

VariantInit(&vVarKey);
parrayKeys = SafeArrayCreateVector(VT_VARIANT, 0L, 4);
vVarKey.vt = VT_ARRAY | VT_VARIANT;
vVarKey.u.parray = parrayKeys;

SafeArrayLock(parrayKeys);
lInx = 0L;
hr = SafeArrayPtrOfIndex(parrayKeys, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = "dwVal_1";
lInx = 1L;
hr = SafeArrayPtrOfIndex(parrayKeys, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = "dwVal_2";
lInx = 2L;
hr = SafeArrayPtrOfIndex(parrayKeys, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = "dwVal_3";
lInx = 3L;
hr = SafeArrayPtrOfIndex(parrayKeys, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = "dwVal_4";
SafeArrayUnlock(parrayKeys);

printf("\r\ncall DMGetValueExStr:");
bRet = DMGetValueExStr(dwFlags, &vVarKey, dmvus, 4L, &err);
```

```
if (bRet == FALSE)
{
    printf("\r\n  error DMGetValueExStr, err=%ld,%ld,%ld,%ld,%ld,
[%s]",
              err.dwError1, err.dwError2, err.dwError3,
err.dwError4, err.dwError5, err.szErrorText);
}
else
{
    int i = 0;
    for (i = 0; i < 4; i++)
    {
       hr = VariantChangeType((VARIANTARG*)&(dmvus[i].vdmValue),
(VARIANTARG*)&(dmvus[i].vdmValue), 0, VT_I4);
        if (hr)
        {
           printf("\r\n    error VariantChangeType[%d] hr=%08lx",
i, hr);
        }
        else
        {
            if (dmvus[i].vdmValue.vt == VT_I4)
            {
                dwVal[i] = dmvus[i].vdmValue.u.lVal;
                printf("\r\n    Var[%d]:[%s] = %ld",
                       i, dmvus[i].vdmVarKey.u.pbVal,
dmvus[i].vdmValue.u.lVal);
            }
            else
            {
            printf("\r\n    wrong datatype %ld, VT_I4 expected
after VariantChange",
                   dmvus[i].vdmValue.vt == VT_I4);
            }
        }
    }

    //clear the internal alloced Data, that no memoryleaks
    //the vdmValues no need to clear here, while no PROPVARIANT
here given (double used)
    //only the dmVarkey of dmvus have to free here
    for (i = 0; i < 4; i++)
    {
        free(dmvus[i].vdmVarKey.u.pbVal);
        dmvus[i].vdmVarKey.u.pbVal = NULL;
        dmvus[i].vdmVarKey.vt = VT_EMPTY;
    }
}

// change the values before set again
dwMerk = dwVal[0];
```

```
dwVal[0] = dwVal[3];
dwVal[3] = dwMerk;
dwMerk = dwVal[1];
dwVal[1] = dwVal[2];
dwVal[2] = dwMerk;

VariantInit(&vdmValue);
parrayValues = SafeArrayCreateVector(VT_VARIANT, 0L, 4);
vdmValue.vt = VT_ARRAY | VT_VARIANT;
vdmValue.u.parray = parrayValues;

SafeArrayLock(parrayValues);
lInx = 0L;
hr = SafeArrayPtrOfIndex(parrayValues, &lInx, &pvElem);
pvElem->vt = VT_I4;
pvElem->u.pbVal = dwVal[0];
lInx = 1L;
hr = SafeArrayPtrOfIndex(parrayValues, &lInx, &pvElem);
pvElem->vt = VT_I4;
pvElem->u.pbVal = dwVal[1];
lInx = 2L;
hr = SafeArrayPtrOfIndex(parrayValues, &lInx, &pvElem);
pvElem->vt = VT_I4;
pvElem->u.pbVal = dwVal[2];
lInx = 3L;
hr = SafeArrayPtrOfIndex(parrayValues, &lInx, &pvElem);
pvElem->vt = VT_I4;
pvElem->u.pbVal = dwVal[3];
SafeArrayUnlock(parrayValues);

printf("\r\ncall DMSetValueExStr:");
memset(&err, 0, sizeof(err));
bRet = DMSetValueExStr(&vVarKey, &vdmValue, pdwVarState, &err);

if (bRet == FALSE)
{
    printf("\r\n  error DMSetValueExStr, err=%ld,%ld,%ld,%ld,%ld,
[%s]",
          err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}
else
{
    int i = 0;
    for (i = 0; i < 4; i++)
    {
        printf("\r\n    dwVarState[%d] = %ld", i, dwState[i]);
    }
}

// clear the values before read back again with DMGetValueWait
(timeout 2sec.)
```

```
dwVal[0] = 0;
dwVal[1] = 0;
dwVal[2] = 0;
dwVal[3] = 0;

printf("\r\ncall DMGetValueWaitExStr:");
memset(&err, 0, sizeof(err));
bRet = DMGetValueWaitExStr(&dwTAID, dwFlags, &vVarKey, TRUE,
2000, DM_NotifyVariableProcExStr_GetValueWait, dwVal, &err);
if (bRet == FALSE)
{
    printf("\r\n  error DMGetValueWaitExStr,
err=%ld,%ld,%ld,%ld,%ld,[%s]",
          err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}
else
{
    int i = 0;
    for (i = 0; i < 4; i++)
    {
        printf("\r\n    dwValue[%d] = %ld", i, dwVal[i]);
    }
}

//increment values for identify execute of following
DMSetValueWait
dwVal[0] = dwVal[0] + 1;
if (dwVal[0] >= 100)
{
    dwVal[0] = dwVal[0] - 100;
}
dwVal[1] = dwVal[1] + 1;
if (dwVal[1] >= 100)
{
    dwVal[1] = dwVal[1] - 100;
}
dwVal[2] = dwVal[2] + 1;
if (dwVal[2] >= 100)
{
    dwVal[2] = dwVal[2] - 100;
}
dwVal[3] = dwVal[3] + 1;
if (dwVal[3] >= 100)
{
    dwVal[3] = dwVal[3] - 100;
}
VariantClear(&vdmValue);
parrayValues = SafeArrayCreateVector(VT_VARIANT, 0L, 4);
vdmValue.vt = VT_ARRAY | VT_VARIANT;
vdmValue.u.parray = parrayValues;
SafeArrayLock(parrayValues);
```

```
lInx = 0L;
hr = SafeArrayPtrOfIndex(parrayValues, &lInx, &pvElem);
pvElem->vt = VT_I4;
pvElem->u.pbVal = dwVal[0];
lInx = 1L;
hr = SafeArrayPtrOfIndex(parrayValues, &lInx, &pvElem);
pvElem->vt = VT_I4;
pvElem->u.pbVal = dwVal[1];
lInx = 2L;
hr = SafeArrayPtrOfIndex(parrayValues, &lInx, &pvElem);
pvElem->vt = VT_I4;
pvElem->u.pbVal = dwVal[2];
lInx = 3L;
hr = SafeArrayPtrOfIndex(parrayValues, &lInx, &pvElem);
pvElem->vt = VT_I4;
pvElem->u.pbVal = dwVal[3];
SafeArrayUnlock(parrayValues);

printf("\r\ncall DMSetValueWaitExStr:");
memset(errArray, 0, sizeof(errArray));
bRet = DMSetValueWaitExStr(&dwTAID, &vVarKey, 4, &vdmValue,
pdwVarState, 2000, DM_CompletitionProc, NULL, errArray);
if (bRet == FALSE)
{
    int i = 0;
    for(i=0; i < 4; i++)
    {
        printf("\r\n    error DMSetValueWaitExStr,
errArray[%d]=%ld,%ld,%ld,%ld,%ld,[%s]",
               i, errArray[i].dwError1, errArray[i].dwError2,
errArray[i].dwError3,
               errArray[i].dwError4, errArray[i].dwError5,
errArray[i].szErrorText);
    }
}
else
{
}

printf("\r\n########## exit Test with DMGetValue(Wait)ExStr/
DMSetValue(Wait)ExStr ##########\r\n");

}
```

脚本示例“项目函数 DM_NotifyVariableProcExStr_GetValueWait”：

```
#pragma code ("OleAut32.dll")
//#include "OleAuto.h"
HRESULT VariantChangeType( VARIANTARG FAR* pvargDest, VARIANTARG
FAR* pvarSrc, unsigned short wFlags, VARTYPE vt);
#pragma code()
```

```
BOOL DM_NotifyVariableProcExStr_GetValueWait(DWORD dwTAID,
LPDM_VAR_UPDATE_STRUCT_EXSTR lpdmvus, DWORD dwItems, LPVOID
lpvUser)
{
BOOL bRet = FALSE;
int i = 0;
DWORD* pdwVal = NULL;

pdwVal = (DWORD*)lpvUser;
// the DMGetValueWait used PROPVARIANT and the lpvUser is ptr to
OutDataArray (DWORDS)
printf("\r\n*** DM_NotifyVariableProcExStr_GetValueWait entry
***");

for (i = 0; i < dwItems; i++)
{
    VariantChangeType((VARIANTARG*)&(lpdmvus[i].vdmValue),
(VARIANTARG*)&(lpdmvus[i].vdmValue), 0, VT_I4);
    if (0 == strcmp((LPCSTR)lpdmvus[i].vdmVarKey.u.pbVal,
"dwVal_1"))
   {
        pdwVal[0] = lpdmvus[i].vdmValue.u.lVal;
        printf("\r\n  set dwVal_1=%ld to dwVal[0]", pdwVal[0]);
        bRet = TRUE;
   }
    if (0 == strcmp((LPCSTR)lpdmvus[i].vdmVarKey.u.pbVal,
"dwVal_2"))
   {
        pdwVal[1] = lpdmvus[i].vdmValue.u.lVal;
        printf("\r\n  set dwVal_2=%ld to dwVal[1]", pdwVal[1]);
        bRet = TRUE;
   }
    if (0 == strcmp((LPCSTR)lpdmvus[i].vdmVarKey.u.pbVal,
"dwVal_3"))
   {
        pdwVal[2] = lpdmvus[i].vdmValue.u.lVal;
        printf("\r\n  set dwVal_3=%ld to dwVal[2]", pdwVal[2]);
        bRet = TRUE;
   }
    if (0 == strcmp((LPCSTR)lpdmvus[i].vdmVarKey.u.pbVal,
"dwVal_4"))
   {
        pdwVal[3] = lpdmvus[i].vdmValue.u.lVal;
        printf("\r\n  set dwVal_4=%ld to dwVal[3]", pdwVal[3]);
        bRet = TRUE;
   }
}

printf("\r\n*** DM_NotifyVariableProcExStr_GetValueWait exit
***");
return bRet;
}
```

```
Project-Function DM_CompletitionProc:

BOOL DM_CompletitionProc(DWORD dwTAID, LPDWORD lpdmVarState,
DWORD dwItems, LPVOID lpvUser)
{
BOOL bRet = FALSE;
int i = 0;

printf("\r\n*** DM_CompletitionProc entry ***");

for (i = 0; i < dwItems; i++)
{
    printf("\r\n    VarState[%d] = %ld", i, lpdmVarState[i]);
}

printf("\r\n*** DM_CompletitionProc exit ***");

return bRet;
}
```

3.2.6.13 DMGetValueWait

使用

从数据管理器的过程映像中读取一个或更多变量的值。读出值为最近更新时存在的一个值。

与 DMGetValue 相比，该函数强制基于值来更新变量的值。

更高级的 DMGetValueWaitEx 函数还会在 DM_NOTIFY_VARIABLEEX_PROC 回调中另外在 DM_VAR_UPDATE_STRUCTEX 结构中返回质量代码。

声明

```
BOOL DMGetValueWait (
    LPDWORD                   pdwTAID,
    LPDM_VARKEY               lpdmVarKey,
    DWORD                     dwItems,
    BOOL                      fWaitForCompletition,
    DWORD                     dwTimeOut,
    DM_NOTIFY_VARIABLE_PROC   lpfnVariable,
    LPVOID                    lpvUser,
    LPCMN_ERROR               lpdmError);
```

参数

pdwTAID

指向包含由数据管理器在成功调用函数之后所分配事务 ID 的变量的指针。

lpdmVarKey

指向可识别要读取的变量值的 DM_VARKEY (页 1869) 结构指针。

dwItems

所传递结构的数量（与将要读取的变量值的数量相对应）。

fWaitForCompletition

如果设置了 fWaitForCompletition 标志，则仅当所有请求的变量都已更新或超出了指定的超时时间时，才会调用传递的回调函数。

如果未设置 fWaitForCompletition，则立即调用传递的回调函数。该时间特性与函数 DMGetValue 相对应。

dwTimeout

应用程序的最长等待时间（毫秒）。如果等待时间过期后未更新完所有变量，则调用回调函数并返回相应的错误代码。

如果设置了 fWaitForCompletition == FALSE 标志，则不评估传递的值。

lpfnVariable

指向 DM_NOTIFY_VARIABLE_PROC 回调函数的指针，在所有请求的变量更新后或在等待时间过后调用该回调函数。

如果 fWaitForCompletition == FALSE，则立即用当前设置在数据管理器的过程映像中的值调用该回调函数。

当程序报告通知例程时，必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

在极少数情况下，在函数调用返回之前已经传送了“通知”。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

值已确定。

FALSE

错误。

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DM_NOTIFY_VARIABLE_PROC (页 2107)	获取更新的变量值（回调）
DMGetValue (页 1977)	获取变量值
DMGetValueWaitEx (页 1994)	获取更新的变量值
DMSetValueWait (页 2035)	更改变量值并发出通知

示例

OnTestVariablenGetvaluewait (页 2156) "TESTCDoc.cpp"

参见

DM_NOTIFY_VARIABLE_PROC (页 2107)

DM_VARKEY (页 1869)

DMGetValue (页 1977)

DMGetValueEx (页 1980)

DMSetValueWait (页 2035)

OnTestVariablenGetvaluewait (页 2156)

DMGetValueWaitEx (页 1994)

3.2.6.14 DMGetValueWaitEx

使用

从数据管理器的过程映像中读取一个或更多变量的值。读出值为最近更新时存在的一个值。

与 DMGetValueEx 相比，该函数强制基于值来更新变量的值。

声明

```
BOOL DMGetValueWaitEx (
    LPDWORD                     pdwTAID,
    LPDM_VARKEY                 lpdmVarKey,
    DWORD                       dwItems,
    BOOL                        fWaitForCompletition,
    DWORD                       dwTimeOut,
    DM_NOTIFY_VARIABLEEX_PROC   lpfnVariable,
    LPVOID                      lpvUser,
    LPCMN_ERROR                 lpdmError);
```

参数

pdwTAID

指向包含由数据管理器在成功调用函数之后所分配事务 ID 的变量的指针。

lpdmVarKey

指向可识别要读取的变量值的 DM_VARKEY (页 1869) 结构指针。

dwItems

所传递结构的数量（与将要读取的变量值的数量相对应）。

fWaitForCompletition

如果设置了 fWaitForCompletition 标志，则仅当所有请求的变量都已更新或超出了指定的超时时间时，才会调用传递的回调函数。

如果未设置 fWaitForCompletition，则立即调用传递的回调函数。该时间特性与函数 DMGetValue 相对应。

dwTimeout

应用程序的最长等待时间（毫秒）。如果等待时间过期后未更新完所有变量，则调用回调函数并返回相应的错误代码。

如果设置了 fWaitForCompletition == FALSE 标志，则不评估传递的值。

lpfnVariable

指向 DM_NOTIFY_VARIABLEEX_PROC 回调函数的指针，在所有请求的变量更新后或在等待时间过后调用该回调函数。

如果 fWaitForCompletition == FALSE，则立即用当前设置在数据管理器的过程映像中的值调用该回调函数。

当程序报告通知例程时，必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

在极少数情况下，在函数调用返回之前已经传送了“通知”。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

值已确定。

FALSE

错误。

相关函数

DM_NOTIFY_VARIABLEEX_PROC (页 2119)	获取通知的变量值（回调）
DMGetValueEx (页 1980)	获取变量值
DMSetValueWait (页 2035)	更改变量值并发出通知

参见

DM_NOTIFY_VARIABLEEX_PROC (页 2119)

DMGetValueWait (页 1991)

DM_VARKEY (页 1869)

DMGetValueEx (页 1980)

DMSetValueWait (页 2035)

3.2.6.15 DMGetValueWaitExStr

声明

```
BOOL DMGetValueWaitExStr (
    LPDWORD                          pdwTAID,
    DWORD                            dwFlags
    LPVARIANT                        lpvdmVarKey,
    LPVARIANT                        lpvCookie,
    BOOL                             fWaitForCompletition,
    DWORD                            dwTimeOut,
    DM_NOTIFY_VARIABLE_PROC_EXSTR    lpfnVariable,
    LPVOID                           lpvUser,
    LPCMN_ERROR                      lpdmError);
```

说明

从数据管理器的过程映像中读取一个或更多变量的值。读出值为最近更新时存在的一个值。

与 DMGetValueExStr 不同，该函数可用来强制基于值来更新变量值。

与 DMGetValueEx 不同，将不再使用 DM_VARKEY 结构；因此，变量名称没有长度限制。

参数

pdwTAID

指向包含由数据管理器在成功调用函数之后所分配事务 ID 的变量的指针。

dwFlags

如果需要以 VT_LPSTR 格式在 DM_NOTIFY_VARIABLE_PROC_EXSTR 的 DM_VAR_UPDATE_STRUCT_EXSTR 结构的 VARIANT 中返回变量名称，则可在此指定 DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR 标志。

lpvdmVarKey

指向变量列表的 VARIANT 的指针。必须创建 VT_ARRAY | VT_VARIANT 类型的列表，因为可能存在不同的数据类型，例如 VT_I4 和 VT_BSTR，某些情况下还包括 VT_LPSTR。

lpvCookie

指向各变量用户相关数据附加列表的 VARIANT 的指针。

该指针可用来替代之前 DM_VARKEY 的 lpvUserData，并且也会返回到每个变量的 DM_VAR_UPDATE_STRUCT_EXSTR 结构中。

fWaitForCompletition

如果设置了 fWaitForCompletion 标志，则仅当所有请求的变量都已更新或超出了指定的超时时间后，才会调用传递的回调函数。

如果未设置 fWaitForCompletition 标志，则立即调用传递的回调函数。该时间与 DMGetValueExStr 函数的时间完全相同。

dwTimeout

应用程序的最长等待时间（毫秒）。如果等待时间过期后未更新完所有变量，则调用回调函数并返回相应的错误代码。

如果 fWaitForCompletition 标志 = FALSE，则不评估传递的值。

lpfnVariable

指向 DM_NOTIFY_VARIABLEEX_PROC_EXSTR 回调函数的指针，该函数在所有请求的变量更新后或在等待时间超时后调用。

如果 fWaitForCompletition = FALSE，则立即用当前存储在数据管理器的过程映像中的值调用该回调函数。

如果程序请求通知例程，则必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

在极少数情况下，通知在函数调用返回之前就已返回。

lpvUser

指向应用程序特定数据的指针。此函数不会评估该指针，但会使其在回调函数中重新可用。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

值已确定

FALSE

错误

所需文件

dmclient_exstr.h

dmclient.lib

dmclient.dll

相关函数

DMGetValueExStr	获取变量值
DMSetValueWaitExStr	更改变量值并发出通知

示例

可在 DMGetValueExStr 页面找到示例。

3.2.6.16 DMGetVarInfo

使用

DMGetVarInfo 确定变量的完整变量关键字。该命令可以用于获取与变量 ID 相关的变量名称，反之亦然。

声明

```
BOOL DMGetVarInfo (
    LPCSTR          lpszProjectFile,
    LPDM_VARKEY     lpdmVarKey,
    DWORD           dwItems,
    LPCMN_ERROR     lpdmError);
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

lpdmVarKey

指向要使用变量密钥完成的 DM_VARKEY (页 1869) 结构指针。

dwItems

完整结构的数目。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

获取变量关键字。

FALSE

错误。

注释

不要和创建变量的调用交替使用该调用。如果在创建变量后第一次使用 DMGetVarInfo，将对搜索列表重新排序，从而会花费更多的执行时间。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器

所需文件

dmclient.h

dmclient.lib

dmclient.dll

示例

OnTestVariablenGetVarInfo (页 2160) "TESTCDoc.cpp"

参见

DM_VARKEY (页 1869)

OnTestVariablenGetVarInfo (页 2160)

3.2.6.17 DMGetVarInfoExStr

声明

```
BOOL DMGetVarInfoExStr (
    LPCTSTR         lpszProjectFile,
    DWORD           dwFlags,
    LPVARIANT       lpvdmVarKeyIn,
    LPVARIANT       lpvdmVarKeyOut,
    LPCMN_ERROR     lpdmError);
```

说明

确定变量的完整变量关键字。该命令可以用于获取变量 ID 的变量名称，反之亦然。

与 DMGetVarInfo 不同，变量列表将作为指向类型为 VT_ARRAY | VT_VARIANT 的 VARIANT 的指针传递。它会在另一个 VARIANT 中返回。因此，变量名没有长度限制。

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称可在运行系统中通过 DMEnumOpenedProjects 或 DMGetRuntimeProject 检索。

dwFlags

如果需要以 VT_LPSTR 格式在 VARIANT lpvdmVarKeyOut 中返回变量名称，则可在此指定 DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR。

lpvdmVarKeyIn

指向类型为 VT_ARRAY | VT_VARIANT 的 VARIANT 的指针，用于传递变量列表或在仅指定了一个变量的情况下传递单个 VARIANT。

在 VT_I4 类型的相关列表元素和 VT_BSTR 类型的变量名称中输入 TagID。此外，还可输入 VT_LPSTR (PROPVARIANT) 作为已分配的 ASCII 字符串进行传递，例如，用以传递来自脚本的常量名称。VT_LPSTR 随后会在内部转换为所需的 VT_BSTR 类型。

lpvdmVarKeyOut

指向用于返回 lpvdmVarKeyIn 的相应值的 VARIANT 的指针。

之后，lpvdmVarKeyIn 中会返回给定变量名称的变量 ID，反之亦然。

如果 VARIANT 使用 VT_EMPTY 进行初始化，则会创建一个相应的数组，并使用与 lpvdmVarKeyIn 中大小相同的 VT_ARRAY | VT_VARIANT 进行填充。

如果 VARIANT 非空，则需要首先调用 VariantClear，之后才能创建新数组。未经过初始化的 VARIANT 可能会导致异常。

如果变量不存在或者无法访问，则相应条目的类型保持为 VT_EMPTY。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

变量关键字已确定

FALSE

错误

注释

此调用不能与创建变量的调用交替使用。在创建变量后第一次使用 DMGetVarInfo 或 DMGetVarInfoExStr 函数时，必须对搜索列表进行重新排序。这需要额外的运行时间。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器

所需文件

dmclient_exstr.h

dmclient.lib

dmclient.dll

示例

脚本示例按钮 DMGetVarInfoExStr：

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
#pragma code("kernel32.dll")
#define CP_ACP 0
int MultiByteToWideChar(UINT CodePage,
                        DWORD dwFlags,
                        LPCSTR lpMultiByteStr,
                        int cbMultiByte,
                        LPWSTR lpWideCharStr,
                       int cchWideChar);
int WideCharToMultiByte(UINT CodePage,
                        DWORD dwFlags,
                        LPCWSTR lpWideCharStr,
                        int cchWideChar,
                        LPSTR lpMultiByteStr,
                        int cbMultiByte,
                        LPCSTR lpDefaultChar,
                        LPBOOL lpUsedDefaultChar);
#pragma code()

#pragma code ("OleAut32.dll")
//#include "OleAuto.h"
SAFEARRAY * SafeArrayCreateVector(VARTYPE vt, long lLbound,
unsigned int cElements );
HRESULT SafeArrayPutElement(SAFEARRAY FAR* psa, long FAR*
rgIndices, void FAR* pv );
HRESULT SafeArrayGetElement(SAFEARRAY FAR* psa, long FAR*
rgIndices, void FAR* pv );
HRESULT SafeArrayPtrOfIndex(SAFEARRAY FAR* psa, long FAR*
rgIndices, void HUGEP* FAR* ppvData );
HRESULT SafeArrayLock(SAFEARRAY FAR* psa);
HRESULT SafeArrayLock(SAFEARRAY FAR* psa);
typedef WCHAR OLECHAR;
BSTR SysAllocString(OLECHAR FAR* sz);
HRESULT SysFreeString(BSTR bstr);
#pragma code()
VARIANT vVarkeyIn;
VARIANT vVarkeyOut;
VARIANT vElem;
VARIANT* pvElem;
HRESULT hr;
CHAR szProjectName[256];
```

```
CMN_ERROR err;
DWORD dwFlags;
BOOL bRet;
DWORD dwVarID1, dwVarID2, dwVarID3, dwVarID4;
CHAR szVarNam1[256], szVarNam2[256], szVarNam3[256],
szVarNam4[265];
SAFEARRAY* parrayIN;
SAFEARRAY* parrayOUT;
long lInx;
WCHAR wszBuffer[256];
int nRet;
memset(&err, 0, sizeof(err));
bRet = FALSE;
szProjectName[0] = 0;
dwFlags = 0;
VariantInit(&vVarkeyIn);
VariantInit(&vVarkeyOut);
VariantInit(&vElem);
parrayIN = NULL;
parrayOUT = NULL;
//hr = E_FAIL;
nRet = 0;
printf("\r\n\r\n########## enter Test with DMGetVarInfoExStr
##########");

bRet = DMGetRuntimeProject(szProjectName, 256, &err);
if (!bRet)
{
    printf("\r\n  error DMGetRuntimeProject =>[%s],
err=%ld,%ld,%ld,%ld,%ld,[%s]", szProjectName,
             err.dwError1, err.dwError2, err.dwError3,
err.dwError4, err.dwError5, err.szErrorText);
}
dwVarID1 = GetTagDWord("dwVarKeyID_1");
dwVarID2 = GetTagDWord("dwVarKeyID_2");
dwVarID3 = GetTagDWord("dwVarKeyID_3");
dwVarID4 = GetTagDWord("dwVarKeyID_4");
strncpy(szVarNam1, GetTagChar("szVarKeyName_1"), 256);
strncpy(szVarNam2, GetTagChar("szVarKeyName_2"), 256);
strncpy(szVarNam3, GetTagChar("szVarKeyName_3"), 256);
strncpy(szVarNam4, GetTagChar("szVarKeyName_4"), 256);
printf("\r\n  set vVarkeyIn:");
printf("\r\n    dwVarID1=%ld, szVarNam1=[%s] (use name as
VT_BSTR)", dwVarID1, szVarNam1);
printf("\r\n    dwVarID2=%ld, szVarNam2=[%s] (use name as
VT_LPSTR)", dwVarID2, szVarNam2);
printf("\r\n    dwVarID3=%ld, szVarNam3=[%s] (use ID)", dwVarID3,
szVarNam3);
printf("\r\n    dwVarID4=%ld, szVarNam4=[%s] (useID)", dwVarID4,
szVarNam4);
//preseting input Variant:name1, name2, id3, id4
// will deliver on output Variant:id1, id2, name3, name4
```

```
parrayIN = SafeArrayCreateVector(VT_VARIANT, 0L, 4);
vVarkeyIn.vt = (VT_ARRAY | VT_VARIANT);
vVarkeyIn.u.parray = parrayIN;
// name1 as VT_BSTR
vElem.vt = VT_BSTR;
nRet =  MultiByteToWideChar(CP_ACP, 0, szVarNam1, -1, wszBuffer,
256);
vElem.u.bstrVal = SysAllocString(wszBuffer);
lInx = 0;
hr = SafeArrayPutElement(parrayIN, &lInx, &vElem);
VariantClear(&vElem);
// name2 als VT_LPSTR
SafeArrayLock(parrayIN);
lInx = 1;
hr = SafeArrayPtrOfIndex(parrayIN, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = szVarNam2;
SafeArrayUnlock(parrayIN);
if (hr)
{
    printf("\r\nerror SafeArrayPutElement:hr = %08lx", hr);
}
VariantClear(&vElem);
vElem.vt = VT_I4;
vElem.u.lVal = dwVarID3;
lInx = 2;
hr = SafeArrayPutElement(parrayIN, &lInx, &vElem);
vElem.vt = VT_I4;
vElem.u.lVal = dwVarID4;
lInx = 3;
hr = SafeArrayPutElement(parrayIN, &lInx, &vElem);

memset(&err, 0, sizeof(err));
bRet = DMGetVarInfoExStr(szProjectName, dwFlags, &vVarkeyIn,
&vVarkeyOut, &err);
if (!bRet)
{
    printf("\r\n  error DMGetVarInfoExStr[%s],
err=%ld,%ld,%ld,%ld,%ld,[%s]", szProjectName,
              err.dwError1, err.dwError2, err.dwError3,
err.dwError4, err.dwError5, err.szErrorText);
    if (err.dwError1 == DM_E_DONT_EXIST)
    {
        //reset error, elements will be checked each for error in
type (VT_EMPTY == not exist)
        bRet = TRUE;
    }
}
else
{
    printf("\r\n  DMGetVarInfoExStr return OK");
}
```

```
if (bRet)
{
    if (vVarkeyOut.vt == VT_EMPTY)
    {
        printf("\r\n  vVarkeyOut is VT_EMPTY");
    }
    else
    {
        parrayOUT = vVarkeyOut.u.parray;
        lInx = 0;
        SafeArrayGetElement(parrayOUT, &lInx, &vElem);
        if (vElem.vt != VT_I4)
        {
            printf("\r\n  vVarKeyOut[%ld].vt = [%ld] != VT_I4",
lInx, vElem.vt);
            dwVarID1 = 0L;
        }
        else
        {
            dwVarID1 = vElem.u.lVal;
         }
        SetTagDWord("dwVarKeyID_1",dwVarID1);
        lInx = 1;
        SafeArrayGetElement(parrayOUT, &lInx, &vElem);
        if (vElem.vt != VT_I4)
        {
            printf("\r\n  vVarKeyOut[%ld].vt = [%ld] != VT_I4",
lInx, vElem.vt);
            dwVarID2 = 0L;
        }
        else
        {
            dwVarID2 = vElem.u.lVal;
        }
        SetTagDWord("dwVarKeyID_2",dwVarID2);
        lInx = 2;
        SafeArrayGetElement(parrayOUT, &lInx, &vElem);
        if (vElem.vt != VT_BSTR)
        {
           printf("\r\n  vVarKeyOut[%ld].vt = [%ld] != VT_BSTR",
lInx, vElem.vt);
            strcpy(szVarNam3, "error!!!");
        }
        else
        {
            nRet = WideCharToMultiByte(CP_ACP, 0L,
(WCHAR*)vElem.u.bstrVal, -1,
                                       szVarNam3, 256, NULL, NULL);
        }
        SetTagChar("szVarKeyName_3",szVarNam3);
        lInx = 3;
        SafeArrayGetElement(parrayOUT, &lInx, &vElem);
```

```
        if (vElem.vt != VT_BSTR)
        {
           printf("\r\n  vVarKeyOut[%ld].vt = [%ld] != VT_BSTR",
lInx, vElem.vt);
            strcpy(szVarNam4, "error!!!");
        }
        else
        {
            nRet = WideCharToMultiByte(CP_ACP, 0L,
(WCHAR*)vElem.u.bstrVal, -1,
                                       szVarNam4, 256, NULL, NULL);
        }
        SetTagChar("szVarKeyName_4",szVarNam4);
        printf("\r\n  get vVarkeyOut:");
        printf("\r\n    dwVarID1=%ld, szVarNam1=[%s] (delivered
ID)", dwVarID1, szVarNam1);
        printf("\r\n    dwVarID2=%ld, szVarNam2=[%s] (delivered
ID)", dwVarID2, szVarNam2);
        printf("\r\n    dwVarID3=%ld, szVarNam3=[%s] (delivered
name VT_BSTR)", dwVarID3, szVarNam3);
        printf("\r\n    dwVarID4=%ld, szVarNam4=[%s] (delivered
name VT_BSTR)", dwVarID4, szVarNam4);
    }
}
printf("\r\n  set flag DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR and
repeat DMGetVarInfoExStr");
dwFlags = DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR;
memset(&err, 0, sizeof(err));
hr = VariantClear(&vVarkeyOut);
if (hr)
{
    printf("\r\nerror VariantClear(vVarKeyOut:hr = %08lx", hr);
}
bRet = DMGetVarInfoExStr(szProjectName, dwFlags, &vVarkeyIn,
&vVarkeyOut, &err);
if (!bRet)
{
    printf("\r\n  error DMGetVarInfoExStr[%s],
err=%ld,%ld,%ld,%ld,%ld,[%s]", szProjectName,
              err.dwError1, err.dwError2, err.dwError3,
err.dwError4, err.dwError5, err.szErrorText);
    if (err.dwError1 == DM_E_DONT_EXIST)
    {
        //reset error, elements will be checked each for error in
type (VT_EMPTY == not exist)
        bRet = TRUE;
    }
}
else
{
    printf("\r\n  DMGetVarInfoExStr return OK");
}
```

```
if (bRet)
{
    if (vVarkeyOut.vt == VT_EMPTY)
    {
        printf("\r\n  vVarkeyOut is VT_EMPTY");
    }
    else
    {
        parrayOUT = vVarkeyOut.u.parray;
        lInx = 0;
        SafeArrayGetElement(parrayOUT, &lInx, &vElem);
        if (vElem.vt != VT_I4)
        {
            printf("\r\n  vVarKeyOut[%ld].vt = [%ld] != VT_I4",
lInx, vElem.vt);
            dwVarID1 = 0L;
        }
        else
        {
            dwVarID1 = vElem.u.lVal;
        }
        SetTagDWord("dwVarKeyID_1",dwVarID1);
        lInx = 1;
        SafeArrayGetElement(parrayOUT, &lInx, &vElem);
        if (vElem.vt != VT_I4)
        {
            printf("\r\n  vVarKeyOut[%ld].vt = [%ld] != VT_I4",
lInx, vElem.vt);
            dwVarID2 = 0L;
        }
        else
        {
            dwVarID2 = vElem.u.lVal;
        }
        SetTagDWord("dwVarKeyID_2",dwVarID2);

        SafeArrayLock(parrayOUT);
        lInx = 2;
        SafeArrayPtrOfIndex(parrayOUT, &lInx, &pvElem);
        if (pvElem->vt != VT_LPSTR)
        {
           printf("\r\n  vVarKeyOut[%ld].vt = [%ld] != VT_LPSTR",
lInx, pvElem->vt);
            strcpy(szVarNam3, "error!!!");
        }
        else
        {
            strncpy(szVarNam3, (LPSTR)pvElem->u.pbVal, 255);
            szVarNam3[255] = 0;
        }
        SetTagChar("szVarKeyName_3",szVarNam3);
        lInx = 3;
```

```
        SafeArrayPtrOfIndex(parrayOUT, &lInx, &pvElem);
        if (pvElem->vt != VT_LPSTR)
        {
           printf("\r\n  vVarKeyOut[%ld].vt = [%ld] != VT_LPSTR",
lInx, pvElem->vt);
            strcpy(szVarNam4, "error!!!");
        }
        else
        {
            strncpy(szVarNam4, (LPSTR)pvElem->u.pbVal, 255);
            szVarNam4[255] = 0;
        }
        SetTagChar("szVarKeyName_4",szVarNam4);
        SafeArrayUnlock(parrayOUT);

        printf("\r\n  get vVarkeyOut:");
        printf("\r\n    dwVarID1=%ld, szVarNam1=[%s] (delivered
ID)", dwVarID1, szVarNam1);
        printf("\r\n    dwVarID2=%ld, szVarNam2=[%s] (delivered
ID)", dwVarID2, szVarNam2);
        printf("\r\n    dwVarID3=%ld, szVarNam3=[%s] (delivered
name VT_LPSTR)", dwVarID3, szVarNam3);
        printf("\r\n    dwVarID4=%ld, szVarNam4=[%s] (delivered
name VT_LPSTR)", dwVarID4, szVarNam4);
    }
}

printf("\r\n  only 1 Input (ID = VT_I4 and dwVarID1) without
VT_ARRAY, set flag DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR and repeat
DMGetVarInfoExStr");
dwFlags = DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR;
memset(&err, 0, sizeof(err));
hr = VariantClear(&vVarkeyIn);
if (hr)
{
    printf("\r\n  error VariantClear(vVarKeyIn):hr = %08lx", hr);
    // know that error, have to free the PROPVARIANT in Array
manualy
   //this can be set to VT_EMPTY direct, while no alloc of string
was used
    parrayIN = vVarkeyIn.u.parray;
    SafeArrayLock(parrayIN);
    lInx = 1;
    hr = SafeArrayPtrOfIndex(parrayIN, &lInx, &pvElem);
    pvElem->vt = VT_EMPTY;
    pvElem->u.pbVal = NULL;
    SafeArrayUnlock(parrayIN);
    hr = VariantClear(&vVarkeyIn);
    if (hr)
    {
        printf("\r\n  error VariantClear(vVarkeyIn) again after
manualy reset the PROPVARIANT in 2. elem:hr = %08lx", hr);
```

```
    }
    else
    {
        printf("\r\n  VariantClear(vVarkeyIn) OK after manualy
reset the PROPVARIANT in 2. elem");
    }
}
hr = VariantClear(&vVarkeyOut);
if (hr)
{
    printf("\r\n  error VariantClear(vVarkeyOut):hr = %08lx", hr);
}
vVarkeyIn.vt = VT_I4;
vVarkeyIn.u.lVal = dwVarID1;
bRet = DMGetVarInfoExStr(szProjectName, dwFlags, &vVarkeyIn,
&vVarkeyOut, &err);
if (!bRet)
{
    printf("\r\n  error DMGetVarInfoExStr[%s],
err=%ld,%ld,%ld,%ld,%ld,[%s]", szProjectName,
              err.dwError1, err.dwError2, err.dwError3,
err.dwError4, err.dwError5, err.szErrorText);
    if (err.dwError1 == DM_E_DONT_EXIST)
    {
        //reset error, elements will be checked each for error in
type (VT_EMPTY == not exist)
        bRet = TRUE;
    }
}
else
{
    printf("\r\n  DMGetVarInfoExStr return OK");
}
if (bRet)
{
    if (vVarkeyOut.vt == VT_EMPTY)
    {
        printf("\r\n  vVarkeyOut is VT_EMPTY");
    }
    else
    {
        if (vVarkeyOut.vt != VT_LPSTR)
        {
            printf("\r\n  vVarKeyOut.vt = [%ld] != VT_LPSTR",
vVarkeyOut.vt);
            strcpy(szVarNam1, "error!!!");
        }
        else
        {
            strncpy(szVarNam1, (LPSTR)vVarkeyOut.u.pbVal, 255);
            szVarNam1[255] = 0;
        }
```

```
            SetTagChar("szVarKeyName_1",szVarNam1);
        }
    }

    printf("\r\n########## exit Test with DMGetVarInfoExStr
    ##########\r\n");

    }
```

3.2.6.18 DMGetVarLimits

使用

DMGetVarLimits 确定指定变量的限值，在此限制范围内允许 DMSetValue 函数进行写操作。

声明

```
BOOL DMGetVarLimits (
    LPCSTR          lpszProjectFile,
    LPDM_VARKEY     lpdmVarKey,
    DWORD           dwItems,
    LPDM_VARLIMIT   lpdmVarLimit,
    LPCMN_ERROR     lpdmError );
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

lpdmVarKey

指向 DM_VARKEY (页 1869) 类型结构的指针，该结构用于指定变量。

dwItems

完整结构的数目。

lpdmVarLimit

指向其中存储了限值的 DM_VARLIMIT (页 1871) 类型结构的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

限值已确定。

FALSE

错误。

注释

除组态中定义的变量上下限外，通过指定的格式对允许的数字范围进行进一步的限制。

以下示例显示了 DM_VARTYPE_BYTE 类型变量的关系，该变量以 BCD 码格式的字节形式存储在自动化系统中。

错误消息

DM_E_MAX_RANGE	超出格式转换上限
DM_E_MIN_RANGE	超出格式转换下限
DM_E_MAX_LIMIT	超出变量上限
DM_E_MIN_LIMIT	超出变量下限
DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_INVALID_KEY	未找到变量
DM_E_INVALID_TYPE	无效变量类型

所需文件

dmclient.h

dmclient.lib

dmclient.dll

示例

OnTestVariablenGetvarlimits (页 2161) "TESTCDoc.cpp"

参见

DM_VARKEY (页 1869)

DM_VARLIMIT (页 1871)

OnTestVariablenGetvarlimits (页 2161)

3.2.6.19 DMGetVarLimitsExStr

声明

```
BOOL DMGetVarLimitsExStr (
    LPCTSTR           lpszProjectFile,
    LPVARIANT         lpvdmVarKey,
    LPDM_VARLIMIT     lpdmVarLimit,
    LPCMN_ERROR       lpdmError );
```

说明

对于指定的变量，DMGetVarLimitsExStr 确定了可使用 DMSetValueExStr 函数执行写操作的限值范围。

与 DMGetVarLimits 不同，变量列表将作为指向类型为 VT_ARRAY | VT_VARIANT 的 VARIANT 的指针传递。因此，变量名没有长度限制。

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称可在运行系统中通过 DMEnumOpenedProjects 或 DMGetRuntimeProject 检索。

lpvdmVarKey

指向类型为 VT_ARRAY | VT_VARIANT 的 VARIANT 的指针，用于传递变量列表或在仅指定了一个变量的情况下传递单个 VARIANT。

在 VT_I4 类型的相关列表元素和 VT_BSTR 类型的变量名称中输入 TagID。此外，还可输入 VT_LPSTR (PROPVARIANT) 作为已分配的 ASCII 字符串进行传递，例如，用以传递来自脚本的常量名称。VT_LPSTR 随后会在内部转换为所需的 VT_BSTR 类型。

lpdmVarLimit

指向用于存储限值的 DM_VARLIMIT 类型结构的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

限值已确定

FALSE

错误

注释

除了已组态的变量上限和下限外，存储的格式指令也进一步限制了有效的数值范围。

以下示例以采用 BCD 字节格式存储在 AS 中的 DM_VARTYPE_BYTE 类型的变量为例，说明了这种情况。

错误消息

DM_E_MAX_RANGE	超出格式转换上限
DM_E_MIN_RANGE	超出格式转换下限
DM_E_MAX_LIMIT	超出变量上限
DM_E_MIN_LIMIT	超出变量下限
DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_INVALID_KEY	未找到变量
DM_E_INVALID_TYPE	无效变量类型

所需文件

dmclient_exstr.h

dmclient.lib

dmclient.dll

示例

脚本示例按钮 DMGetVarLimitsExStr：

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
#pragma code ("OleAut32.dll")
SAFEARRAY * SafeArrayCreateVector(VARTYPE vt, long lLbound,
unsigned int cElements );
HRESULT SafeArrayPtrOfIndex(SAFEARRAY FAR* psa, long FAR*
rgIndices, void HUGEP* FAR* ppvData );
HRESULT SafeArrayLock(SAFEARRAY FAR* psa);
HRESULT SafeArrayLock(SAFEARRAY FAR* psa);
HRESULT VariantChangeType( VARIANTARG FAR* pvargDest, VARIANTARG
FAR* pvarSrc, unsigned short wFlags, VARTYPE vt);
#pragma code()

VARIANT vdmVarkey;
VARIANT* pvElem;
VARIANT vElem;
HRESULT hr;
CHAR szProjectName[256];
CMN_ERROR err;
BOOL bRet;
CHAR szVarNam1[256], szVarNam2[256], szVarNam3[256],
szVarNam4[265];
SAFEARRAY* parray;
long lInx;
int nRet;
DM_VARLIMIT dmLimits[4];
int i;

memset(&err, 0, sizeof(err));
bRet = FALSE;
szProjectName[0] = 0;
VariantInit(&vElem);
VariantInit(&vdmVarkey);
parray = NULL;
nRet = 0;
i = 0;
```

```
printf("\r\n\r\n########## enter Test with DMGetVarLimitsExStr
##########");

bRet = DMGetRuntimeProject(szProjectName, 256, &err);
if (!bRet)
{
    printf("\r\n  error DMGetRuntimeProject =>[%s],
err=%ld,%ld,%ld,%ld,%ld,[%s]", szProjectName,
          err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}

strncpy(szVarNam1, GetTagChar("szVarKeyName_1"), 256);
strncpy(szVarNam2, GetTagChar("szVarKeyName_2"), 256);
strncpy(szVarNam3, GetTagChar("szVarKeyName_3"), 256);
strncpy(szVarNam4, GetTagChar("szVarKeyName_4"), 256);
printf("\r\n  set the vdmVarkey with names in VT_LPSTR:");
parray = SafeArrayCreateVector(VT_VARIANT, 0L, 4);
vdmVarkey.vt = (VT_ARRAY | VT_VARIANT);
vdmVarkey.u.parray = parray;
SafeArrayLock(parray);
lInx = 0;
hr = SafeArrayPtrOfIndex(parray, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = szVarNam1;
lInx = 1;
hr = SafeArrayPtrOfIndex(parray, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = szVarNam2;
lInx = 2;
hr = SafeArrayPtrOfIndex(parray, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = szVarNam3;
lInx = 3;
hr = SafeArrayPtrOfIndex(parray, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = szVarNam4;
SafeArrayLock(parray);

memset(&err, 0, sizeof(err));
memset(dmLimits, 0, sizeof(DM_VARLIMIT)*4);
bRet = DMGetVarLimitsExStr(szProjectName, &vdmVarkey, dmLimits,
&err);
if (!bRet)
{
    printf("\r\n  error DMGetVarLimitsExStr[%s],
err=%ld,%ld,%ld,%ld,%ld,[%s]", szProjectName,
          err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
    bRet = TRUE; //clear error flag, the defect data is set to
VT_EMPTY
}
```

```
else
{
    printf("\r\n  DMGetVarLimitsExStr return OK");
}
if (bRet)
{
    for (i = 0; i < 4; i++)
    {
        VariantClear(&vElem);
        printf("\r\n    dmLimits[%d]:{",i);
        hr = VariantChangeType((VARIANTARG*)&vElem,
(VARIANTARG*)&dmLimits[i].dmMaxRange, 0, VT_R8);
        if ((0L == hr) && (VT_R8 == vElem.vt) && (VT_EMPTY !=
dmLimits[i].dmMaxRange.vt))
        {
            printf("dmMaxRange=[%g], ", vElem.u.dblVal);
        }
        else
        {
            printf("dmMaxRange{VariantChangeType error hr =
0x%08lx, vt=0x%04x}",
                    hr, dmLimits[i].dmMaxRange.vt);
        }
        VariantClear(&vElem);
        hr = VariantChangeType((VARIANTARG*)&vElem,
(VARIANTARG*)&dmLimits[i].dmMinRange, 0, VT_R8);
        if ((0L == hr) && (VT_R8 == vElem.vt) && (VT_EMPTY !=
dmLimits[i].dmMinRange.vt))
        {
            printf("dmMinRange=[%g], ", vElem.u.dblVal);
        }
        else
        {
            printf("dmMinRange{VariantChangeType error hr =
0x%08lx, vt=0x%04x}",
                    hr, dmLimits[i].dmMinRange.vt);
        }
        VariantClear(&vElem);
        hr = VariantChangeType((VARIANTARG*)&vElem,
(VARIANTARG*)&dmLimits[i].dmMaxLimit, 0, VT_R8);
        if ((0L == hr) && (VT_R8 == vElem.vt) && (VT_EMPTY !=
dmLimits[i].dmMaxLimit.vt))
        {
            printf("dmMaxLimit=[%g], ", vElem.u.dblVal);
        }
        else
        {
          printf("dmMaxLimitVariantChangeType error hr = 0x%08lx,
vt=0x%04x}",
                    hr, dmLimits[i].dmMaxLimit.vt);
        }
        VariantClear(&vElem);
```

```
        hr = VariantChangeType((VARIANTARG*)&vElem,
(VARIANTARG*)&dmLimits[i].dmMinLimit, 0, VT_R8);
        if ((0L == hr) && (VT_R8 == vElem.vt) && (VT_EMPTY !=
dmLimits[i].dmMinLimit.vt))
        {
            printf("dmMinLimit=[%g]", vElem.u.dblVal);
        }
        else
        {
          printf("dmMinLimitVariantChangeType error hr = 0x%08lx,
vt=0x%04x}",
                    hr, dmLimits[i].dmMinLimit.vt);
        }
        printf("}");
    }
}

printf("\r\n########## exit Test with DMGetVarLimitsExStr
##########\r\n");

}
```

3.2.6.20 DMGetVarType

使用

DMGetVarType 用于确定所传递变量的类型参考信息。

声明

```
BOOL DMGetVarType (
    LPCSTR          lpszProjectFile,
    LPDM_VARKEY     lpdmVarKey,
    DWORD           dwItems,
    LPDM_TYPEREF    lpdmTypeRef,
    LPCMN_ERROR     lpdmError);
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

lpdmVarKey

指向变量密钥用于决定类型的 DM_VARKEY (页 1869) 结构的指针。

dwItems

完整结构的数目。

lpdmTypeRef

指向用于返回有关变量类型详细信息的 DM_TYPEREF (页 1852) 结构的列表的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

数据类型已确定。

FALSE

错误。

注释

用于创建变量的组态函数将完整的 DM_VARREF 和 DM_TYPEREF 结构传递到包含变量数据类型详细信息的调用应用程序。

要在应用程序内节省存储空间，不必记住每个变量本身的类型引用，可通过数据管理器使用 DMGetVarType 随时查询该信息。

不要和创建变量的调用交替使用该调用。如果在创建变量后第一次使用 DMGetVarType，将对搜索列表重新排序，从而会花费更多的执行时间。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_INVALID_KEY	未找到变量

所需文件

dmclient.h

dmclient.lib

dmclient.dll

示例

OnTestVariablenGetvartype (页 2162) "TESTCDoc.cpp"

参见

DM_VARKEY (页 1869)

DM_TYPEREF (页 1852)

OnTestVariablenGetvartype (页 2162)

3.2.6.21 DMGetVarTypeExStr

声明

```
BOOL DMGetVarTypeExStr (
    LPCTSTR                lpszProjectFile,
    LPVARIANT              lpvdmVarKey,
    LPDM_TYPEREF_EXSTR     lpdmTypeRef,
    LPCMN_ERROR            lpdmError);
```

说明

用于确定所传递变量的类型参考信息。

与 DMGetVarType 不同，变量列表作为指向采用 VT_ARRAY | VT_ VARIANT 形式的 VARIANT 的指针传递。因此，变量名没有长度限制。这同样适用于 DM_TYPEREF_EXSTR 结构中的类型名称。

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称可在运行系统中通过 DMEnumOpenedProjects 或 DMGetRuntimeProject 检索。

lpvdmVarKey

指向类型为 VT_ARRAY | VT_VARIANT 的 VARIANT 的指针，用于传递变量列表。

在 VT_I4 类型的相关列表元素和 VT_BSTR 类型的变量名称中输入 TagID。此外，还可输入 VT_LPSTR (PROPVARIANT) 作为已分配的 ASCII 字符串进行传递，例如，用以传递来自脚本的常量名称。VT_LPSTR 随后会在内部转换为所需的 VT_BSTR 类型。

lpdmTypeRef

指向用于返回变量类型信息的 DM_TYPEREF_EXSTR 结构列表的指针。

该列表至少要达到 lpvdmVarKey 中变体数组的大小。指针 lpszTypeName 和 dwBufferCount 必须进行初始化，并指向单独的字符串数组。

如果 lpszTypeName 为 NULL，请检查返回值。已分配的指针需要再次启用它。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已确认数据类型

FALSE

错误

注释

为有效利用应用程序中的内存，将不强制应用程序存储每个变量的类型参考信息。采用的做法是，应用程序可随时通过 DMGetVarType 从数据管理器获取此信息。

此调用不能与创建变量的调用交替使用。在创建变量之后第一次使用“DMGetVarType”或“DMGetVarTypeExStr”函数时，必须对搜索列表进行重新排序。这需要额外的运行时间。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_INVALID_KEY	未找到变量

所需文件

dmclient_exstr.h

dmclient.lib

dmclient.dll

示例

脚本示例按钮 DMGetVarTypeExStr：

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
#pragma code ("OleAut32.dll")
SAFEARRAY * SafeArrayCreateVector(VARTYPE vt, long lLbound,
unsigned int cElements );
HRESULT SafeArrayPtrOfIndex(SAFEARRAY FAR* psa, long FAR*
rgIndices, void HUGEP* FAR* ppvData );
HRESULT SafeArrayLock(SAFEARRAY FAR* psa);
HRESULT SafeArrayLock(SAFEARRAY FAR* psa);
#pragma code()

VARIANT vdmVarkey;
VARIANT* pvElem;
HRESULT hr;
CHAR szProjectName[256];
CMN_ERROR err;
BOOL bRet;
CHAR szVarNam1[256], szVarNam2[256], szVarNam3[256],
szVarNam4[256];
SAFEARRAY* parray;
long lInx;
int nRet;
DM_TYPEREF_EXSTR dmTypeRef[4];
CHAR szTypeName[4][256];
int i;

memset(&err, 0, sizeof(err));
memset(dmTypeRef, 0, sizeof(DM_TYPEREF_EXSTR)*4);
szTypeName[0][0] = 0;
szTypeName[1][0] = 0;
szTypeName[2][0] = 0;
szTypeName[3][0] = 0;
bRet = FALSE;
szProjectName[0] = 0;
VariantInit(&vdmVarkey);
parray = NULL;
nRet = 0;
i = 0;
```

```
printf("\r\n\r\n########## enter Test with DMGetVarTypeExStr
##########");

bRet = DMGetRuntimeProject(szProjectName, 256, &err);
if (!bRet)
{
    printf("\r\n  error DMGetRuntimeProject =>[%s],
err=%ld,%ld,%ld,%ld,%ld,[%s]", szProjectName,
          err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}

strncpy(szVarNam1, GetTagChar("szVarKeyName_1"), 256);
strncpy(szVarNam2, GetTagChar("szVarKeyName_2"), 256);
strncpy(szVarNam3, GetTagChar("szVarKeyName_3"), 256);
strncpy(szVarNam4, GetTagChar("szVarKeyName_4"), 256);
printf("\r\n  set the vdmVarkey with names in VT_LPSTR:");
parray = SafeArrayCreateVector(VT_VARIANT, 0L, 4);
vdmVarkey.vt = (VT_ARRAY | VT_VARIANT);
vdmVarkey.u.parray = parray;
SafeArrayLock(parray);
lInx = 0;
hr = SafeArrayPtrOfIndex(parray, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = szVarNam1;
lInx = 1;
hr = SafeArrayPtrOfIndex(parray, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = szVarNam2;
lInx = 2;
hr = SafeArrayPtrOfIndex(parray, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = szVarNam3;
lInx = 3;
hr = SafeArrayPtrOfIndex(parray, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = szVarNam4;
SafeArrayLock(parray);

printf("\r\n  call DMGetVarTypeExStr with NULL for names");
memset(&err, 0, sizeof(err));
bRet = DMGetVarTypeExStr(szProjectName, &vdmVarkey, dmTypeRef,
&err);
if (!bRet)
{
    printf("\r\n  error DMGetVarTypeExStr[%s],
err=%ld,%ld,%ld,%ld,%ld,[%s]", szProjectName,
          err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}
else
```

```
{
    printf("\r\n  DMGetVarTypeExStr return OK");
}
if (bRet)
{
    for (i = 0; i < 4; i++)
    {
        printf("\r\n    dmTypeRef[%d]:{",i);
        printf("dwType=[%ld]", dmTypeRef[i].dwType);
        printf(", dwSize=[%ld]", dmTypeRef[i].dwSize);
        if (dmTypeRef[i].lpszTypeName)
        {
            printf(", lpszTypeName=[%s]",
dmTypeRef[i].lpszTypeName);
            printf(", dwNameCharCount=[%ld]",
dmTypeRef[i].dwNameCharCount);
            //clear the delivered strings that no memoryleak
            free(dmTypeRef[i].lpszTypeName);
            dmTypeRef[i].lpszTypeName = NULL;
            dmTypeRef[i].dwNameCharCount = 0L;
            //set name store for next read (no need to free after
call)
            dmTypeRef[i].lpszTypeName = szTypeName[i];
            dmTypeRef[i].dwNameCharCount = 255;
        }
        printf("}");
    }
}
printf("\r\n  call DMGetVarTypeExStr with pointers and size for
name storage");
memset(&err, 0, sizeof(err));
bRet = DMGetVarTypeExStr(szProjectName, &vdmVarkey, dmTypeRef,
&err);
if (!bRet)
{
    printf("\r\n  error DMGetVarTypeExStr[%s],
err=%ld,%ld,%ld,%ld,%ld,[%s]", szProjectName,
          err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}
else
{
    printf("\r\n  DMGetVarTypeExStr return OK");
}
if (bRet)
{
    for (i = 0; i < 4; i++)
    {
        printf("\r\n    dmTypeRef[%d]:{",i);
        printf("dwType=[%ld]", dmTypeRef[i].dwType);
        printf(", dwSize=[%ld]", dmTypeRef[i].dwSize);
        printf(", lpszTypeName=[%s]", dmTypeRef[i].lpszTypeName);
```

```
        printf(", dwNameCharCount=[%ld]",
dmTypeRef[i].dwNameCharCount);
        printf("}");
    }
}
printf("\r\n########## exit Test with DMGetVarTypeExStr
##########\r\n");

}
```

3.2.6.22 DMSetValue

使用

该函数将由 lpdmVarKey 描述的变量的值更改为在 lpdmValue 中指定的值。

按照“过后不管”的原则对变量执行不同步更改。

声明

```
BOOL DMSetValue (
        LPDM_VARKEY    lpdmVarKey,
        DWORD          dwItems,
        LPVARIANT      lpdmValue,
        LPDWORD        lpdmVarState,
        LPCMN_ERROR    lpdmError);
```

参数

lpdmVarKey

指向识别要更改的变量值的第一个 DM_VARKEY (页 1869) 结构的指针。

dwItems

所传递结构的数量（与将要更改的变量值的数量相对应）。

lpdmValue

指向要更改的变量的第一个新值的指针。

lpdmVarState

指向第一个存储单元的指针，该单元中存储的信息指示是否可成功更改变量值或是否出现了错误。

值 0 (OK) 意味着已成功完成传输/修改，且至少存在一个指定变量。仅当不存在指定变量时，才出现具有 DM_VARSTATE_INVALID_KEY 状态的错误。

必须使用 DMSetValueWait 来检查是否已经应用该值。

DM_VARSTATE_NOT_ESTABLISHED	(0x0001)	未与伙伴建立连接
DM_VARSTATE_HANDSHAKE_ERROR	(0x0002)	协议错误
DM_VARSTATE_HARDWARE_ERROR	(0x0004)	网络模块故障
DM_VARSTATE_MAX_LIMIT	(0x0008)	违反了组态的上限
DM_VARSTATE_MIN_LIMIT	(0x0010)	违反了组态的下限
DM_VARSTATE_MAX_RANGE	(0x0020)	超出格式上限
DM_VARSTATE_MIN_RANGE	(0x0040)	超出格式下限
DM_VARSTATE_CONVERSION_ERROR	(0x0080)	显示转换错误（与 DM_VARSTATE_..._RANGE 一起显示）
DM_VARSTATE_STARTUP_VALUE	(0x0100)	变量的初始化值
DM_VARSTATE_DEFAULT_VALUE	(0x0200)	变量的替换值
DM_VARSTATE_ADDRESS_ERROR	(0x0400)	通道寻址错误
DM_VARSTATE_INVALID_KEY	(0x0800)	变量未找到/不可用
DM_VARSTATE_ACCESS_FAULT	(0x1000)	不允许访问变量
DM_VARSTATE_TIMEOUT	(0x2000)	超时/通道无反馈
DM_VARSTATE_SERVERDOWN	(0x4000)	服务器停机

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

变量已更改。

FALSE

错误。

注释

由于数据管理器可基于传递的变量关键字映射每个类型，因而无需指定对 lpdmValue 的访问规则。

在过程中执行对外部变量的写访问。对于 WinCC 客户端 PC，该请求会转发到相应的服务器。

通常，不同步执行应用程序的写请求。所有写入请求都列在服务器上数据管理器的队列中，并顺序传递至正确的通道 DLL。

如果通道 DLL 正面确认该写请求，则负责该请求的服务器的过程映像中的新值即会生效。随后，在通过系统总线互连的 WinCC 站内执行更新。

因此，变量在不同的站上可能临时有不同的值。

错误消息

DM_E_MAX_LIMIT	超出变量上限
DM_E_MIN_LIMIT	超出变量下限
DM_E_MAX_RANGE	超出格式转换上限
DM_E_MIN_RANGE	超出格式转换下限
DM_E_ACCESS_FAULT	已禁止对变量进行写访问
DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_PARAM	无效参数
DM_E_OOM	内存不足

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DMGetValue (页 1977)	获取变量值
DMSetValueMessage (页 2030)	更改变量值并发出报警
DMSetValueWait (页 2035)	更改变量值并发出通知
DMSetValueWaitMessage (页 2041)	更改变量值并发出通知和报警

示例

写入变量 (页 2177) "DM02.cpp"

OnTestVariablenSetvalue (页 2164) "TESTCDoc.cpp"

参见

DM_VARKEY (页 1869)

DMGetValue (页 1977)

DMGetValueEx (页 1980)

DMSetValueMessage (页 2030)

DMSetValueWait (页 2035)

DMSetValueWaitMessage (页 2041)

写入变量 (页 2177)

OnTestVariablenSetvalue (页 2164)

3.2.6.23 DMSetValueExStr

声明

```
BOOL DMSetValueExStr (
    LPVARIANT      lpvdmVarKey,
    LPVARIANT      lpvdmValue,
    LPDWORD        lpdmVarState,
    LPCMN_ERROR    lpdmError);
```

说明

该函数将由 lpvdmVarKey 描述的变量的值更改为在 lpvdmValue 中指定的值。

根据“发后即忘”原则，变量修改以异步模式传送。

与 DMSetValue 不同，变量列表将作为指向类型为 VT_ARRAY | VT_VARIANT 的 VARIANT 的指针传递。因此，变量名没有长度限制。

参数

lpvdmVarKey

指向类型为 VT_ARRAY | VT_VARIANT 的 VARIANT 的指针，用于传递变量列表或在仅指定了一个变量的情况下传递单个 VARIANT。

在 VT_I4 类型的相关列表元素和 VT_BSTR 类型的变量名称中输入 TagID。此外，还可输入 VT_LPSTR (PROPVARIANT) 作为已分配的 ASCII 字符串进行传递，例如，用以传递来自脚本的常量名称。VT_LPSTR 随后会在内部转换为所需的 VT_BSTR 类型。

对于 WinCC 版本 5.0 或更高版本中的多客户端项目，此时可能需要为每个名称添加服务器前缀（参见“项目类型和版本”）。

lpvdmValue

指向要更改的变量的第一个新值的指针。

lpdmVarState

指向存储单元的指针，该单元中存储的信息指示是否可成功更改变量值或是否出现了错误。

值“0”(OK) 意味着已成功完成传输/修改，且至少存在一个指定变量。仅在不存在任何指定变量时，才会返回 DM_VARSTATE_INVALID_KEY 错误状态。

要检查 PLC 是否也应用了该值，必须使用 DMSetValueWaitExStr。

DM_VARSTATE_NOT_ESTABLISHED	(0x0001)	未与伙伴建立连接
DM_VARSTATE_HANDSHAKE_ERROR	(0x0002)	协议错误
DM_VARSTATE_HARDWARE_ERROR	(0x0004)	网络模块故障
DM_VARSTATE_MAX_LIMIT	(0x0008)	超出组态的上限
DM_VARSTATE_MIN_LIMIT	(0x0010)	超出组态的下限
DM_VARSTATE_MAX_RANGE	(0x0020)	超出格式上限
DM_VARSTATE_MIN_RANGE	(0x0040)	超出格式下限
DM_VARSTATE_CONVERSION_ERROR	(0x0080)	转换错误显示（与 DM_VARSTATE_..._RANGE 一起显示）
DM_VARSTATE_STARTUP_VALUE	(0x0100)	变量的初始化值
DM_VARSTATE_DEFAULT_VALUE	(0x0200)	变量的替换值
DM_VARSTATE_ADDRESS_ERROR	(0x0400)	通道寻址错误
DM_VARSTATE_INVALID_KEY	(0x0800)	变量未找到/不可用
DM_VARSTATE_ACCESS_FAULT	(0x1000)	不允许访问变量

DM_VARSTATE_TIMEOUT	(0x2000)	超时/通道无反馈
DM_VARSTATE_SERVERDOWN	(0x4000)	服务器停机

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

变量已更改。

FALSE

错误

注释

无需为 lpdmValue 指定访问指令，因为传递的变量关键字会启用数据管理器以正确分配信息。

对于外部变量，在 WinCC 客户端计算机将作业传递到相应的服务器时，将获得对过程的写访问权限。

应用程序的写作业通常以异步模式执行，即将所有写作业排列到由数据管理器保留在服务器上的队列中，然后按顺序传递到合适的通道 DLL。

如果通道 DLL 肯定确认该写作业，则负责该作业的服务器的过程映像中的新值即会生效。随后，在通过系统总线互连的 WinCC 站内执行更新。

因此，变量在不同的站上可能临时有不同的值。

错误消息

DM_E_MAX_LIMIT	超出变量上限
DM_E_MIN_LIMIT	超出变量下限
DM_E_MAX_RANGE	超出格式转换上限
DM_E_MIN_RANGE	超出格式转换下限
DM_E_ACCESS_FAULT	禁止对变量进行写访问
DM_E_NOT_CONNECTED	未连接到数据管理器

DM_E_PARAM	无效参数
DM_E_OOM	内存不足

所需文件

dmclient_exstr.h

dmclient.lib

dmclient.dll

相关函数

DMGetValueExStr	获取变量值
DMSetValueMessageExStr	更改变量值并输出报警
DMSetValueWaitExStr	更改变量值并发出通知
DMSetValueWaitMessageExStr	更改变量值并发出通知和输出报警

示例

可在 DMGetValueExStr 页面找到示例。

3.2.6.24 DMSetValueMessage

使用

该函数将由 lpdmVarKey 描述的变量的值更改为在 lpdmValue 中指定的值。

与 DMSetValue 函数不同，该函数仅更改一个变量。成功更新变量值之后，生成可自由定义的报警文本。

声明

```
BOOL DMSetValueMessage (
    LPDM_VARKEY    lpdmVarKey,
    LPVARIANT      lpdmValue,
    DWORD          fFlags,
    LPSTR          lpszMessage,
    LPCMN_ERROR    lpdmError);
```

参数

lpdmVarKey

指向可识别要更改的变量的 DM_VARKEY (页 1869) 结构指针。

从 WinCC 版本 5.0 起，也可以在 lpdmVarKey->szName 中为多客户端项目指定服务器前缀（参见项目类型和版本）。

lpdmValue

指向将要更改变量的新值的指针。

fFlags

fFlags 确定如何处理报警文本：

DMSVM_OPERATIONREPORT	将报警存储在操作员输入日志中。
DMSVM_OPERATIONMESSAGE	将报警输出为操作员输入报警。

lpszMessage

要传送的报警文本。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

变量已更改。

FALSE

错误。

注释

可以在 DMSetValue 函数中找到有关更改变量值的更多信息。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_PARAM	无效参数
DM_E_OOM	内存不足

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DMSetValue (页 2024)	更改变量值
DMSetValueWait (页 2035)	更改变量值并发出通知
DMSetValueWaitMessage (页 2041)	更改变量值并发出通知和报警

参见

DM_VARKEY (页 1869)

DMSetValue (页 2024)

DMSetValueWait (页 2035)

DMSetValueWaitMessage (页 2041)

3.2.6.25 DMSetValueMessageExStr

声明

```
BOOL DMSetValueMessageExStr (
    LPVARIANT       lpvdmVarKey,
    LPVARIANT       lpvdmValue,
    DWORD           fFlags,
    LPTSTR          lpszMessage,
    LPCMN_ERROR     lpdmError);
```

说明

该函数将由 lpvdmVarKey 描述的变量的值更改为在 lpdmValue 中指定的值。

与 DMSetValueExStr 函数不同，该函数一次只能更改一个变量。如果已成功更改变量值，将传送所选的报警文本。

参数

lpvdmVarKey

指向 VARIANT 的指针，用于传递变量名称、TagID 或单个 VARIANT（如果仅指定了一个用以识别要更改的变量的变量）。

在 VT_I4 类型的相关列表元素和 VT_BSTR 类型的变量名称中输入 TagID。此外，还可输入 VT_LPSTR (PROPVARIANT) 作为已分配的 ASCII 字符串进行传递，例如，用以传递来自脚本的常量名称。VT_LPSTR 随后会在内部转换为所需的 VT_BSTR 类型。

如果使用 VT_ARRAY | VT_VARIANT 指定数组，则只会评估第一个元素。

对于 WinCC 版本 5.0 或更高版本中的多客户端项目，此时可能需要为每个名称添加服务器前缀（参见“项目类型和版本”）。

lpvdmValue

指向将要更改变量的新值的指针。

fFlags

fFlags 指定如何处理报警文本：

DMSVM_OPERATIONREPORT	将报警存储在操作员输入日志中。
DMSVM_OPERATIONMESSAGE	将报警输出为操作员输入报警。

lpszMessage

要传送的报警文本。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已更改变量

FALSE

错误

注释

有关更改变量值的详细信息，请参见 DMSetValueExStr 函数。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_PARAM	无效参数
DM_E_OOM	内存不足

所需文件

dmclient_exstr.h

dmclient.lib

dmclient.dll

相关函数

DMSetValueExStr	更改变量值
DMSetValueWaitExStr	更改变量值并发出通知
DMSetValueWaitMessageExStr	更改变量值并发出通知和输出报警

3.2.6.26 DMSetValueWait

使用

该函数将由 lpdmVarKey 描述的变量的值更改为在 lpdmValue 中指定的值。

与 DMSetValue 不同，DMSetValueWait 允许在本地设备上成功完成更新后通知应用程序。

声明

```
BOOL DMSetValueWait (
    LPDWORD                 pdwTAID,
    LPDM_VARKEY             lpdmVarKey,
    DWORD                   dwItems,
    LPVARIANT               lpdmValue,
    DWORD                   dwTimeOut,
    DM_COMPLETITION_PROC    lpfnCompletition,
    LPVOID                  lpvUser,
    LPCMN_ERROR             lpdmError);
```

参数

pdwTAID

指向包含由数据管理器在成功调用函数之后所分配事务 ID 的变量的指针。

lpdmVarKey

指向识别要更改的变量值的第一个 DM_VARKEY (页 1869) 结构的指针。

dwItems

所传递结构的数量（与将要更改的变量值的数量相对应）。

lpdmValue

指向要更改的变量的第一个新值的指针。

dwTimeout

应用程序的最长等待时间（毫秒）。如果在等待周期结束后未写入所有变量，则会使用适当的错误代码调用回调函数。

lpfnCompletition

指向回调函数的指针，在所有请求的变量更新后或在等待时间过后调用该回调函数。

当程序报告通知例程时，必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

在极少数情况下，在函数调用返回之前已经传送了“通知”。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向第一个具有 CMN_ERROR 类型的 dwItems 错误结构的指针。在写入变量过程中出现错误时，系统会将错误信息写入适当的结构中。因此，请记得要为这些结构预留空间。

返回值

TRUE

变量已更改。

FALSE

可通过错误结构识别出现的错误。

注释

可以在 DMSetValue 函数中找到有关更改变量值的更多信息。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_PARAM	无效参数

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DMSetValue (页 2024)	更改变量值
DMSetValueMessage (页 2030)	更改变量值并发出报警
DMSetValueWaitMessage (页 2041)	更改变量值并发出通知和报警
DM_COMPLETITION_PROC (页 2046)	更改变量值（回调）

示例

OnTestVariablenSetvaluewait (页 2165) "TESTCDoc.cpp"

参见

DM_VARKEY (页 1869)

DMGetValueWait (页 1991)

DMGetValueWaitEx (页 1994)

DMSetValue (页 2024)

DMSetValueMessage (页 2030)

DMSetValueWaitMessage (页 2041)

DM_COMPLETITION_PROC (页 2046)

OnTestVariablenSetvaluewait (页 2165)

3.2.6.27 DMSetValueWaitExStr

声明

```
BOOL DMSetValueWaitExStr (
    LPDWORD                 pdwTAID,
    LPVARIANT               lpvdmVarKey,
    DWORD                   dwItems,
    LPVARIANT               lpvdmValue,
    LPDWORD                 lpdwState,
    DWORD                   dwTimeOut,
    DM_COMPLETITION_PROC    lpfnCompletition,
    LPVOID                  lpvUser,
    LPCMN_ERROR             lpdmError);
```

说明

该函数将由 lpvdmVarKey 描述的变量的值更改为在 lpvdmValue 中指定的值。

与 DMSetValueExStr 函数不同，若已执行更新，DMSetValueWaitExStr 将在本地计算机上启用应用程序以显示通知消息。

与 DMSetValueWait 不同，变量列表将作为指向类型为 VT_ARRAY | VT_VARIANT 的 VARIANT 的指针传递。因此，变量名没有长度限制。

参数

pdwTAID

指向包含由数据管理器在成功调用函数之后所分配事务 ID 的变量的指针。

lpvdmVarKey

指向类型为 VT_ARRAY | VT_VARIANT 的 VARIANT 的指针，用于传递变量列表或在仅指定了一个变量的情况下传递单个 VARIANT。

在 VT_I4 类型的相关列表元素和 VT_BSTR 类型的变量名称中输入 TagID。此外，还可输入 VT_LPSTR (PROPVARIANT) 作为已分配的 ASCII 字符串进行传递，例如，用以传递来自脚本的常量名称。VT_LPSTR 随后会在内部转换为所需的 VT_BSTR 类型。

对于 WinCC 版本 5.0 或更高版本中的多客户端项目，此时可能需要添加服务器前缀（参见“项目类型和版本”）。

dwItems

传入 lpvdmValue 的值的数量与 lpdmError 中的错误结构。该数量必须至少与 lpvdmVarKey 中输入数组的数量一样多。

如果该数量大于输入数组的数量，将忽略超过的值并将 DM_E_PARAM 分配给受影响的错误结构，因为无法进行分配。

如果该数量小于输入数组的数量，将在第一个错误结构中返回错误 DM_E_PARAM，并且不写入值。

lpvdmValue

指向要更改的变量的第一个新值的指针。

lpdwState

指向存储单元的指针，该单元中存储的信息指示是否可成功更改变量值或是否出现了错误。

值 0 (OK) 意味着已成功完成传输/修改，且至少存在一个指定变量。仅在不存在任何指定变量时，才会返回 DM_VARSTATE_INVALID_KEY 错误状态。

DM_VARSTATE_NOT_ESTABLISHED	(0x0001)	未与伙伴建立连接
DM_VARSTATE_HANDSHAKE_ERROR	(0x0002)	协议错误
DM_VARSTATE_HARDWARE_ERROR	(0x0004)	网络模块故障
DM_VARSTATE_MAX_LIMIT	(0x0008)	超出组态的上限
DM_VARSTATE_MIN_LIMIT	(0x0010)	超出组态的下限
DM_VARSTATE_MAX_RANGE	(0x0020)	超出格式上限
DM_VARSTATE_MIN_RANGE	(0x0040)	超出格式下限
DM_VARSTATE_CONVERSION_ERROR	(0x0080)	转换错误显示（与 DM_VARSTATE_..._RANGE 一起显示）
DM_VARSTATE_STARTUP_VALUE	(0x0100)	变量的初始化值
DM_VARSTATE_DEFAULT_VALUE	(0x0200)	变量的替换值
DM_VARSTATE_ADDRESS_ERROR	(0x0400)	通道寻址错误
DM_VARSTATE_INVALID_KEY	(0x0800)	变量未找到/不可用
DM_VARSTATE_ACCESS_FAULT	(0x1000)	不允许访问变量
DM_VARSTATE_TIMEOUT	(0x2000)	超时/通道无反馈
DM_VARSTATE_SERVERDOWN	(0x4000)	服务器停机

dwTimeout

应用程序的最长等待时间（毫秒）。如果等待时间过期后未更新完所有变量，则调用回调函数并返回相应的错误代码。

lpfnCompletition

指向回调函数的指针，该回调函数在所有请求的变量更新后或在等待时间超时后调用。

如果程序请求通知例程，则必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

在极少数情况下，通知在函数调用返回之前就已返回。

lpvUser

指向应用程序特定数据的指针。此函数不会评估该指针，但会使其在回调函数中重新可用。

lpdmError

指向第一个 CMN_ERROR 类型的 dwItems 错误结构的指针。写入变量出错时，系统会将错误信息写入相应的结构。因此，切记要为这些结构预留空间。

返回值

TRUE

已更改变量

FALSE

可通过错误结构识别出现的错误。

注释

有关更改变量值的详细信息，请参见 DMSetValueExStr 函数。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_PARAM	无效参数

所需文件

dmclient_exstr.h

dmclient.lib

dmclient.dll

相关函数

DMSetValueExStr	更改变量值
DMSetValueMessageExStr	更改变量值并输出报警
DMSetValueWaitMessageExStr	更改变量值并发出通知和输出报警
DM_COMPLETITION_PROC	更改变量值（回调）

示例

可在 DMGetValueExStr 页面找到示例。

3.2.6.28 DMSetValueWaitMessage

使用

该函数将 DMSetValueMessage 函数和 DMSetValueWait 函数组合。与 DMSetValueWait 函数不同，该函数只能更改一个变量。

将由 lpdmVarKey 描述的变量的值更改为在 lpdmValue 中指定的值。在成功更改变量后，输出可自由定义的报警文本并且通知应用程序已在本地设备上成功执行更新。

声明

```
BOOL DMSetValueWaitMessage (
    LPDWORD                 pdwTAID,
    LPDM_VARKEY             lpdmVarKey,
    LPVARIANT               lpdmValue,
    DWORD                   dwTimeOut,
    DM_COMPLETITION_PROC    lpfnCompletition,
    DWORD                   fFlags,
    LPSTR                   lpszMessage,
    LPVOID                  lpvUser,
    LPCMN_ERROR             lpdmError);
```

参数

pdwTAID

指向包含由数据管理器在成功调用函数之后所分配事务 ID 的变量的指针。

lpdmVarKey

指向可识别要更改的变量的 DM_VARKEY (页 1869) 结构的指针。

lpdmValue

指向将要更改变量的新值的指针。

dwTimeOut

应用程序的最长等待时间（毫秒）。如果在等待周期结束后未写入所有变量，则会使用适当的错误代码调用回调函数。

lpfnCompletition

指向回调函数的指针，在所有请求的变量更新后或在等待时间过后调用该回调函数。

当程序报告通知例程时，必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

在极少数情况下，在函数调用返回之前已经传送了“通知”。

fFlags

fFlags 确定如何处理报警文本：

DMSVM_OPERATIONREPORT	将报警存储在操作员输入日志中。
DMSVM_OPERATIONMESSAGE	将报警输出为操作员输入报警。

lpszMessage

输出报警文本。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向第一个具有 CMN_ERROR 类型的 dwItems 错误结构的指针。在写入变量过程中出现错误时，系统会将错误信息写入适当的结构中。因此，请记得要为这些结构预留空间。

返回值

TRUE

变量已更改。

FALSE

错误。

注释

可以在 DMSetValue 函数中找到有关更改变量值的更多信息。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_PARAM	无效参数
DM_E_OOM	内存不足

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DM_COMPLETITION_PROC (页 2046)	更改变量值（回调）
DMSetValue (页 2024)	更改变量值
DMSetValueMessage (页 2030)	更改变量值并发出报警
DMSetValueWait (页 2035)	更改变量值并发出通知

参见

DM_VARKEY (页 1869)

DMSetValue (页 2024)

DMSetValueMessage (页 2030)

DMSetValueWait (页 2035)

OnTestVariablenSetvaluewait (页 2165)

DM_COMPLETITION_PROC (页 2046)

3.2.6.29 DMSetValueWaitMessageExStr

声明

```
BOOL DMSetValueWaitMessageExStr (
    LPDWORD                 pdwTAID,
    LPVARIANT               lpvdmVarKey,
    LPVARIANT               lpvdmValue,
    DWORD                   dwTimeOut,
    DM_COMPLETITION_PROC    lpfnCompletition,
    DWORD                   fFlags,
    LPTSTR                  lpszMessage,
    LPVOID                  lpvUser,
    LPCMN_ERROR             lpdmError);
```

说明

该函数兼具 DMSetValueMessageExStr 和 DMSetValueWaitExStr 的功能。

与 DMSetValueWaitExStr 函数不同，该函数一次只能更改一个变量。

更改由 lpvdmVarKey 描述的变量的值以在 lpvdmValue 中获得指定的值。如果已成功更改变量值，将传送所选的报警文本。如果已执行更新，应用程序还可选择在本地计算机上显示通知消息。

与 DMSetValueWaitMessage 不同，变量列表将作为指向 VARIANT 的指针传递。因此，变量名称的长度没有限制。

参数

pdwTAID

指向包含由数据管理器在成功调用函数之后所分配事务 ID 的变量的指针。

lpvdmVarKey

指向 VARIANT 的指针，用于传递变量名称、TagID 或单个 VARIANT（如果仅指定了一个用以识别要更改的变量的变量）。

在 VT_I4 类型的相关列表元素和 VT_BSTR 类型的变量名称中输入 TagID。此外，还可输入 VT_LPSTR (PROPVARIANT) 作为已分配的 ASCII 字符串进行传递，例如，用以传递来自脚本的常量名称。VT_LPSTR 随后会在内部转换为所需的 VT_BSTR 类型。

对于 WinCC 版本 5.0 或更高版本中的多客户端项目，此时可能需要添加服务器前缀（参见“项目类型和版本”）。

lpvdmValue

指向将要更改变量的新值的指针。

dwTimeOut

应用程序的最长等待时间（毫秒）。如果等待时间过期后未更新完所有变量，则调用回调函数并返回相应的错误代码。

lpfnCompletition

指向回调函数的指针，该回调函数在所有请求的变量更新后或在等待时间超时后调用。

如果程序请求通知例程，则必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

在极少数情况下，通知在函数调用返回之前就已返回。

fFlags

fFlags 指定如何处理报警文本：

DMSVM_OPERATIONREPORT	将报警存储在操作员输入日志中。
DMSVM_OPERATIONMESSAGE	将报警输出为操作员输入报警。

lpszMessage

要传送的报警文本。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向第一个 CMN_ERROR 类型的 dwItems 错误结构的指针。写入变量出错时，系统会将错误信息写入相应的结构。因此，切记要为这些结构预留空间。

返回值

TRUE

已更改变量

FALSE

错误

注释

有关更改变量值的详细信息，请参见 DMSetValueExStr 函数。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_PARAM	无效参数
DM_E_OOM	内存不足

所需文件

dmclient_exstr.h

dmclient.lib

dmclient.dll

相关函数

DM_COMPLETITION_PROC	更改变量值（回调）
DMSetValueExStr	更改变量值
DMSetValueMessageExStr	更改变量值并输出报警
DMSetValueWaitExStr	更改变量值并发出通知

3.2.6.30 DM_COMPLETITION_PROC

说明

为了通知应用程序已成功更改变量值，必须提供 DM_COMPLETITION_PROC 类型的回调函数。

声明

```
BOOL ( * DM_COMPLETITION_PROC) (
    DWORD      dwTAID,
    LPDWORD    lpdmVarState,
    DWORD      dwItems,
    LPVOID     lpvUser);
```

参数

dwTAID

数据管理器针对更改变量值功能所分配的事务 ID。

lpdmVarState

指向第一个存储单元的指针，该单元中存储的信息指示是否可成功更改变量值或是否出现了错误。

0 (OK) 意味着发送/更改成功并且至少存在一个指定变量。仅当不存在指定变量时，才出现具有 DM_VARSTATE_INVALID_KEY 状态的错误。

DM_VARSTATE_NOT_ESTABLISHED	(0x0001)	未与伙伴建立连接
DM_VARSTATE_HANDSHAKE_ERROR	(0x0002)	协议错误
DM_VARSTATE_HARDWARE_ERROR	(0x0004)	网络模块故障
DM_VARSTATE_MAX_LIMIT	(0x0008)	违反了组态的上限
DM_VARSTATE_MIN_LIMIT	(0x0010)	违反了组态的下限
DM_VARSTATE_MAX_RANGE	(0x0020)	超出格式上限
DM_VARSTATE_MIN_RANGE	(0x0040)	超出格式下限
DM_VARSTATE_CONVERSION_ERROR	(0x0080)	显示转换错误（与 DM_VARSTATE_..._RANGE 一起显示）
DM_VARSTATE_STARTUP_VALUE	(0x0100)	变量的初始化值
DM_VARSTATE_DEFAULT_VALUE	(0x0200)	变量的替换值
DM_VARSTATE_ADDRESS_ERROR	(0x0400)	通道寻址错误
DM_VARSTATE_INVALID_KEY	(0x0800)	变量未找到/不可用
DM_VARSTATE_ACCESS_FAULT	(0x1000)	不允许访问变量
DM_VARSTATE_TIMEOUT	(0x2000)	超时/通道无反馈
DM_VARSTATE_SERVERDOWN	(0x4000)	服务器停机

dwItems

变量更改的数量，其状态将传送到 lpdmVarState。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

返回值视执行情况而定

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

当程序报告通知例程时，必须按照指定的时间间隔清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

在极少数情况下，在函数调用返回之前已经传送了“通知”。

所需文件

dmclient.h

相关函数

DMSetValueWait (页 2035)	更改变量值并发出通知
DMSetValueWaitMessage (页 2041)	更改变量值并发出通知和报警

参见

DMSetValueWait (页 2035)

DMSetValueWaitMessage (页 2041)

3.2.6.31 DMShowVarPropertiesExStr

声明

```
BOOL DMShowVarPropertiesExStr (
    LPCTSTR         lpszProjectFile,
    HWND            hwndParent,
    LPCTSTR         lpszVariableName,
    DWORD           dwVariableID,
    LPCMN_ERROR     lpdmError);
```

说明

该函数可打开用于编辑变量属性的对话框。

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称可在运行系统中通过 DMEnumOpenedProjects 或 DMGetRuntimeProject 检索。

hwndParent

用作该对话框父窗口的窗口的句柄。

lpszVariableName

指向要显示属性的变量名称的指针。

如果 lpszVariableName 为“NULL”，则需指定有效的 dwVariableID。若未指定，将返回 DM_E_PARAM 错误。

dwVariableID

变量的 ID。如果已在 lpszVariableName 中指定名称，则不使用。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

单击“确定”(OK) 关闭对话框。

FALSE

出错或单击“取消”(Cancel) 关闭对话框

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_INVALID_KEY	未找到变量
DM_E_CANCEL	用户已在对话框中选择“取消”(Cancel)
DM_E_PARAM	至少已传递一个存在错误的参数

所需文件

dmclient_exstr.h

dmclient.lib

dmclient.dll

示例

脚本示例按钮 DMShowVarDatabaseExStr：

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
BOOL bRet = FALSE;
CHAR szProjectName[256];
CHAR szVariableName[256];
DWORD dwVariableID;
CMN_ERROR err;
HWND hwndParent;

szProjectName[0] = 0;
szVariableName[0] = 0;
dwVariableID = 0L;
memset(&err, 0, sizeof(err));
hwndParent = NULL;
```

```
printf("\r\n\r\n########## enter Test with DMShowVarDatabaseExStr
##########");

bRet = DMGetRuntimeProject(szProjectName, 256, &err);
if (!bRet)
{
    printf("\r\n  error DMGetRuntimeProject =>[%s],
err=%ld,%ld,%ld,%ld,%ld,[%s]", szProjectName,
          err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}

memset(&err, 0, sizeof(err));

dwVariableID = GetTagDWord("dwVarKeyID_1");
printf("\r\n  call DMShowVarPropertiesExStr with ID");
bRet = DMShowVarPropertiesExStr(szProjectName, hwndParent, NULL,
dwVariableID, &err);
if (!bRet)
{
    printf("\r\n  error
DMShowVarPropertiesExStr:err=%ld,%ld,%ld,%ld,%ld,[%s]",
          err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}
else
{
    printf("\r\n  DMShowVarPropertiesExStr OK!");
}

strncpy(szVariableName, GetTagChar("szVarKeyName_1"), 255);
szVariableName[255] = 0;

printf("\r\n  call DMShowVarPropertiesExStr with VarName");
bRet = DMShowVarPropertiesExStr(szProjectName, hwndParent,
szVariableName, 0L, &err);
if (!bRet)
{
    printf("\r\n  error
DMShowVarPropertiesExStr:err=%ld,%ld,%ld,%ld,%ld,[%s]",
          err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}
else
{
    printf("\r\n  DMShowVarPropertiesExStr OK!");
}

printf("\r\n########## exit Test with DMShowVarProperitesExStr
##########\r\n");
}
```

3.2.6.32 DMShowVarDatabase

使用

该函数为指定的项目打开变量选择对话框。与 ShowVarDatabaseMulti 函数不同，该函数仅选择一个变量。

声明

```
BOOL DMShowVarDatabase (
    LPCSTR              lpszProjectFile,
    HWND                hwndParent,
    LPDM_DLGOPTIONS     lpdmOptions,
    LPDM_VARFILTER      lpdmFilter,
    LPDM_VARKEY         lpdmVarKey,
    LPCMN_ERROR         lpdmError);
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

hwndParent

用作该对话框父窗口的窗口的句柄。

lpdmOptions

指向 DM_DLGOPTIONS (页 1844) 结构的指针，该结构指定了对话框应的响应方式。当指针是 NULL 时，选择标准对话框。

lpdmFilter

指向 DM_VARFILTER (页 1859) 过滤器结构的指针。当指针是 NULL 时，显示所有变量。

仅支持按变量名和变量类型过滤。不执行按变量组和变量连接过滤。

lpdmVarKey

指向 DM_VARKEY (页 1869) 结构的指针。如果要在对话框关闭后显示变量的属性，则 lpdmVarKey 包含该变量的关键字。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

单击“确定”(OK) 关闭对话框。

FALSE

出错或单击“取消”(Cancel) 关闭对话框。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_INV_PRJ	未找到/加载指定项目

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DMShowVarDatabaseMulti (页 2057)	打开变量选择对话框

参见

DM_VARKEY (页 1869)

DM_VARFILTER (页 1859)

DM_DLGOPTIONS (页 1844)

DMShowVarDatabaseMulti (页 2057)

3.2.6.33 DMShowVarDatabaseExStr

声明

```
BOOL DMShowVarDatabaseExStr (
    LPCTSTR            lpszProjectFile,
    HWND               hwndParent,
    LPDM_DLGOPTIONS    lpdmOptions,
    LPDM_VARFILTER     lpdmFilter,
    LPTSTR*            lppszVariableName,
    LPDWORD            lpdwVarNameCharCount,
    LPDWORD            lpdwVariableID,
    LPCMN_ERROR        lpdmError);
```

说明

该函数为指定的项目打开变量选择对话框。与 ShowVarDatabaseMultiExStr 不同，该函数只能选择选择一个变量。

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称可在运行系统中通过 DMEnumOpenedProjects 或 DMGetRuntimeProject 检索。

hwndParent

用作该对话框父窗口的窗口的句柄。

lpdmOptions

指向包含对话框说明的 DM_DLGOPTIONS 结构的指针；如果为 NULL，则为标准对话框。

lpdmFilter

指向 DM_VARFILTER 过滤器结构的指针。若为 NULL，则显示所有变量。

仅支持按变量名和变量类型过滤。不执行按变量组和变量连接过滤。

lpszVariableName

指向变量名称返回缓冲区指针的指针。

若为 NULL，必须存在指向 lpdwVariableID 的有效指针。随后仅返回 ID 而不返回名称。不考虑 dwVarNameCharCount。若两个指针均为 NULL，则返回 DM_E_PARAM。

若内部指针为 NULL，则分配和返回缓冲区。缓冲区的大小将返回到 lpdwVarNameCharCount 中。

lpdwVarNameCharCount

指向存储返回缓冲区大小的 DWORD 的指针。必须选择足够大的大小，以存储带零终止的变量名称。

如果缓冲区过小，将存储截断的名称并返回 DM_E_OOM 错误。

lpdwVariableID

指向存储变量 ID 的 DWORD 的指针。

若为 NULL，必须存在指向 lpdwVariableName 的有效指针以及有效的 dwVarNameCharCount 大小。随后将仅返回变量名称。

无法保证始终返回变量 ID，例如 PackageTag、特定 S7 变量等。在这些情况下，将返回 0。

在某些情况下，可通过后续调用 DMGetVarInfoExStr 来补充信息。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

单击“确定”(OK) 关闭对话框。

FALSE

出错或单击“取消”(Cancel) 关闭对话框。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_INV_PRJ	未找到/加载指定项目
DM_E_PARAM	参数不一致
DM_E_OOM	内存错误：dwVarNameCharCount 中指定的缓冲区 lpszVariableName 过小并且名称被截断

所需文件

dmclient_exstr.h

dmclient.lib

dmclient.dll

相关函数

DMShowVarDatabaseMultiExStr	打开变量选择对话框

示例

脚本示例按钮 DMShowVarDatabaseExStr：

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
BOOL bRet = FALSE;
CMN_ERROR err;
CHAR szProjectName[256];
HWND hwndParent;
CHAR szVariableName[256];
LPSTR pszVariableName;
DWORD dwVarNamCharCount = 256;
DWORD dwVarID = 0L;

szProjectName[0] = 0;
szVariableName[0] = 0;
hwndParent = NULL;
pszVariableName= szVariableName;

memset(&err, 0, sizeof(err));

printf("\r\n\r\n########## enter Test with DMShowVarDatabaseExStr
##########");

bRet = DMGetRuntimeProject(szProjectName, 256, &err);
if (!bRet)
{
    printf("\r\n  error DMGetRuntimeProject =>[%s],
err=%ld,%ld,%ld,%ld,%ld,[%s]", szProjectName,
          err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}

bRet = DMShowVarDatabaseExStr(szProjectName, hwndParent, NULL,
NULL,
```

```
                                    &pszVariableName, &dwVarNamCharCount,
&dwVarID, &err);
if (!bRet)
{
    printf("\r\n  error
DMShowVarDatabaseExStr:err=%ld,%ld,%ld,%ld,%ld,[%s]",
          err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}
else
{
    printf("\r\n  DMShowVarDatabaseEsStr OK!  (set to first
line)");
    printf("\r\n  szVarName=[%s], dwVarID=[%ld]", szVariableName,
dwVarID);
    SetTagChar("szVarKeyName_1",szVariableName);
    SetTagDWord("dwVarKeyID_1",dwVarID);
}

}
```

3.2.6.34 DMShowVarDatabaseMulti

使用

该函数为指定的项目打开变量选择对话框。可以选择多个变量，这与 ShowVarDatabase 不同。

声明

```
BOOL DMShowVarDatabaseMulti (
    LPCSTR                      lpszProjectFile,
    HWND                        hwndParent,
    LPDM_DLGOPTIONS             lpdmOptions,
    LPDM_VARFILTER              lpdmFilter,
    LPDWORD                     lpdwItems,
    DM_NOTIFY_SELECT_VAR_PROC   lpfnVariables,
    LPVOID                      lpvUser,
    LPCMN_ERROR                 lpdmError);
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

hwndParent

用作该对话框父窗口的窗口的句柄。

lpdmOptions

指向 DM_DLGOPTIONS (页 1844) 结构的指针，该结构指定了对话框应的响应方式。若是 NULL，则使用标准对话框。

lpdmFilter

指向 DM_VARFILTER (页 1859) 过滤器结构的指针。若是 NULL,，则显示所有变量。

仅支持按变量名和变量类型过滤。不执行按变量组和变量连接过滤。

lpdwItems

指向 DWORD 缓冲区的指针，该缓冲区接收相应数量的所选变量。

lpfnVariables

指向为每个所选变量调用的回调函数的指针。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

单击“确定”(OK) 关闭对话框。

FALSE

出错或单击“取消”(Cancel) 关闭对话框。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_INV_PRJ	未找到/加载指定项目

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DMShowVarDatabase (页 2052)	打开变量选择对话框
DM_NOTIFY_SELECT_VAR_PROC (页 2063)	打开变量选择对话框（回调）

参见

DM_VARFILTER (页 1859)

DMShowVarDatabase (页 2052)

DM_DLGOPTIONS (页 1844)

DM_NOTIFY_SELECT_VAR_PROC (页 2063)

3.2.6.35 DMShowVarDatabaseMultiExStr

声明

```
BOOL DMShowVarDatabaseMultiExStr (
    LPCTSTR                             lpszProjectFile,
    HWND                                hwndParent,
    LPDM_DLGOPTIONS                     lpdmOptions,
    LPDM_VARFILTER                      lpdmFilter,
    LPDWORD                             lpdwItems,
    DM_NOTIFY_SELECT_VAR_PROC_EXSTR     lpfnVariables,
    LPVOID                              lpvUser,
    LPCMN_ERROR                         lpdmError);
```

说明

该函数为指定的项目打开变量选择对话框。与 ShowVarDatabaseExStr 不同，该函数可选择多个变量。

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称可在运行系统中通过 DMEnumOpenedProjects 或 DMGetRuntimeProject 检索。

hwndParent

用作该对话框父窗口的窗口的句柄。

lpdmOptions

指向包含对话框说明的 DM_DLGOPTIONS 结构的指针；如果为 NULL，则为标准对话框。

lpdmFilter

指向 DM_VARFILTER 过滤器结构的指针。若为 NULL，则显示所有变量。

仅支持按变量名和变量类型过滤。不执行按变量组和变量连接过滤。

lpdwItems

指向存储所选变量总数的 DWORD 缓冲区的指针。

lpfnVariables

指向为每个所选变量调用的回调函数的指针。

lpvUser

指向应用程序特定数据的指针。此函数不会评估该指针，但会使其在回调函数中重新可用。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

单击“确定”(OK) 关闭对话框。

FALSE

出错或单击“取消”(Cancel) 关闭对话框。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_INV_PRJ	未找到/加载指定项目

所需文件

dmclient_exstr.h

dmclient.lib

dmclient.dll

相关函数

DMShowVarDatabaseExStr	打开变量选择对话框
DM_NOTIFY_SELECT_VAR_PROC_EXSTR	打开变量选择对话框（回调）

示例

脚本示例按钮 DMShowVarDatabaseMultiExStr：

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
extern BOOL  DM_NotifySelectVarProcA(LPCSTR lpszVariableName,
DWORD dwVariableID, LPVOID lpvUser);

BOOL bRet = FALSE;
CMN_ERROR err;
CHAR szProjectName[256];
HWND hwndParent;
DWORD dwItems;
DWORD dwInxDecr;

szProjectName[0] = 0;
hwndParent = NULL;
memset(&err, 0, sizeof(err));
dwItems = 0;
dwInxDecr = 4; /*for decrement index to save in DM tags from
callback set to lpvUser*/

printf("\r\n\r\n########## enter Test with
DMShowVarDatabaseMultiExStr ##########");
```

```
bRet = DMGetRuntimeProject(szProjectName, 256, &err);
if (!bRet)
{
    printf("\r\n  error DMGetRuntimeProject =>[%s],
err=%ld,%ld,%ld,%ld,%ld,[%s]", szProjectName,
          err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}

bRet = DMShowVarDatabaseMultiExStr(szProjectName, hwndParent,
NULL, NULL,
                                   &dwItems, DM_NotifySelectVarProcA,
&dwInxDecr, &err);
if (!bRet)
{
    printf("\r\n  error
DMShowVarDatabaseMultiExStr:err=%ld,%ld,%ld,%ld,%ld,[%s]",
          err.dwError1, err.dwError2, err.dwError3, err.dwError4,
err.dwError5, err.szErrorText);
}
else
{
    printf("\r\n  DMShowVarDatabaseExStrMulti OK!");
    printf("\r\n  [%ld] Var's selected", dwItems);
}

printf("\r\n########## exit Test with DMShowVarDatabaseMultiExStr
##########\r\n");
}
```

脚本示例项目函数 DM_NotifySelectVarProcA：

```
BOOL  DM_NotifySelectVarProcA(LPCSTR lpszVariableName, DWORD
dwVariableID, LPVOID lpvUser)
{
BOOL bRet = TRUE;
DWORD* pdwInxDecr = NULL;

pdwInxDecr = (DWORD*)lpvUser;

printf("\r\n\r\n#### DM_NotifySelectVarProc entry: ####");
printf("\r\n    VarName=[%s], VarID=[%ld], lpvUser=[0x%08lx]",
lpszVariableName, dwVariableID, lpvUser);
if (pdwInxDecr)
{
    // handle max 4 outputs, ignore all others
    if(4 == *pdwInxDecr )
    {
        SetTagChar("szVarKeyName_4",lpszVariableName);
        SetTagDWord("dwVarKeyID_4",dwVariableID);
```

```
        printf("\r\n        save in szVarKeyName_4 and
dwVarKeyID_4");
    }
    if(3 == *pdwInxDecr )
    {
        SetTagChar("szVarKeyName_3",lpszVariableName);
        SetTagDWord("dwVarKeyID_3",dwVariableID);
        printf("\r\n        save in szVarKeyName_3 and
dwVarKeyID_3");
    }
    if(2 == *pdwInxDecr )
    {
        SetTagChar("szVarKeyName_2",lpszVariableName);
        SetTagDWord("dwVarKeyID_2",dwVariableID);
        printf("\r\n        save in szVarKeyName_2 and
dwVarKeyID_2");
    }
    if(1 == *pdwInxDecr )
    {
        SetTagChar("szVarKeyName_1",lpszVariableName);
        SetTagDWord("dwVarKeyID_1",dwVariableID);
        printf("\r\n        save in szVarKeyName_1 and
dwVarKeyID_1");
    }
}
if (pdwInxDecr && (0L < *pdwInxDecr))
{
    *pdwInxDecr = *pdwInxDecr - 1L;
    printf("\r\n        Inx=%ld", *pdwInxDecr);
}

printf("\r\n#### DM_NotifySelectVarProc exit ####\r\n");

return bRet;
}
```

3.2.6.36 DM_NOTIFY_SELECT_VAR_PROC

说明

为了评估 DMShowVarDatabaseMulti 函数选择的变量，必须提供 DM_NOTIFY_SELECT_VAR_PROC 类型的回调函数。

声明

```
BOOL ( * DM_NOTIFY_SELECT_VAR_PROC) (
    LPDM_VARKEY    lpdmVarKey,
    DWORD          dwItem,
    LPVOID         lpvUser );
```

参数

lpdmVarKey

指向具有变量名称和 ID 的第一个 DM_VARKEY (页 1869) 类型结构的指针。

dwItem

传入 lpdmVarKey 的结构的数量。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

dmclient.h

相关函数

DMShowVarDatabaseMulti (页 2057)	打开变量选择对话框

参见

DM_VARKEY (页 1869)

DMShowVarDatabaseMulti (页 2057)

3.2.6.37 GAPICreateNewVariable

使用

创建新变量或检查变量是否存在。 该函数只能用于临时 $ 变量。

声明

```
BOOL GAPICreateNewVariable (
    LPMCP_NEWVARIABLE_DATA    pData,
    LPCMN_ERROR               lpdmError);
```

参数

pData

指向具有变量数据的 MCP_NEWVARIABLE_DATA (页 1873) 结构的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。 出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

变量已创建。

当检查时： 变量尚不存在。

FALSE

错误。

当使用 Errorcode1 = DM_E_ALREADY_EXIST 检查时： 变量存在

注释

有更多高级函数。

GAPICreateNewVariable4
GAPICreateNewVariableEx4
GAPICreateNewVariable5

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_ALREADY_EXIST	已存在要创建的对象

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

GAPICreateNewVariable4 (页 2067)	创建变量
GAPICreateNewVariableEx4 (页 2070)	创建带有创建者 ID 的变量
GAPICreateNewVariable5 (页 2069)	创建带有创建者 ID 的变量

参见

MCP_NEWVARIABLE_DATA (页 1873)

GAPICreateNewVariable4 (页 2067)

GAPICreateNewVariable5 (页 2069)

GAPICreateNewVariableEx4 (页 2070)

3.2.6.38 GAPICreateNewVariable4

使用

创建新变量或检查变量是否存在。该函数只能用于临时 $ 变量。

该函数额外指定了标定信息，因此与 GAPICreateNewVariable 不同。

声明

```
BOOL GAPICreateNewVariable4 (
    LPMCP_NEWVARIABLE_DATA_4    pData,
    LPCMN_ERROR                 lpdmError);
```

参数

pData

指向具有变量数据的 MCP_NEWVARIABLE_DATA_4 (页 1875) 结构的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

变量已创建。

当检查时：变量尚不存在。

FALSE

错误。

当使用 Errorcode1 = DM_E_ALREADY_EXIST 检查时：变量存在。

注释

有更多高级函数。

GAPICreateNewVariableEx4	创建带有创建者 ID 的变量
GAPICreateNewVariable5	创建带有创建者 ID 的变量

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_ALREADY_EXIST	已存在要创建的对象

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

GAPICreateNewVariable (页 2065)	创建变量
GAPICreateNewVariableEx4 (页 2070)	创建带有创建者 ID 的变量
GAPICreateNewVariable5 (页 2069)	创建带有创建者 ID 的变量

参见

GAPICreateNewVariable (页 2065)

MCP_NEWVARIABLE_DATA_4 (页 1875)

GAPICreateNewVariable5 (页 2069)

GAPICreateNewVariableEx4 (页 2070)

3.2.6.39 GAPICreateNewVariable5

使用

创建新变量或检查变量是否存在。该函数只能用于临时 $ 变量。

该函数与 GAPICreateNewVariable4 不同，它指定了创建者 ID 以及起始值和替换值，也可为文本变量指定这些值。

声明

```
BOOL GAPICreateNewVariable5 (
    DWORD                       dwCreatorID,
    LPMCP_NEWVARIABLE_DATA_5    pData,
    LPCMN_ERROR                 lpdmError);
```

参数

dwCreatorID

通过创建者标识，可确定对象的创建者。

保留值 0 – 10100 以及 11000 – 11100，供内部或特定系统使用。

pData

指向具有变量数据的 MCP_NEWVARIABLE_DATA_5 (页 1878) 结构的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

变量已创建。

当检查时：变量尚不存在。

FALSE

错误。

当使用 Errorcode1 = DM_E_ALREADY_EXIST 检查时：变量存在。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_ALREADY_EXIST	已存在要创建的对象

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

GAPICreateNewVariable (页 2065)	创建变量
GAPICreateNewVariable4 (页 2067)	创建变量
GAPICreateNewVariableEx4 (页 2070)	创建带有创建者 ID 的变量

参见

GAPICreateNewVariable (页 2065)

GAPICreateNewVariable4 (页 2067)

MCP_NEWVARIABLE_DATA_5 (页 1878)

GAPICreateNewVariableEx4 (页 2070)

MCP_VARIABLE_LIMITS5 (页 1892)

3.2.6.40 GAPICreateNewVariableEx4

使用

创建新变量或检查变量是否存在。 该函数只能用于临时 $ 变量。

该函数指定了创建者 ID，因此与 GAPICreateNewVariable4 不同。

声明

```
BOOL GAPICreateNewVariableEx4 (
    DWORD                       dwCreatorID,
    LPMCP_NEWVARIABLE_DATA_4    pData,
    LPCMN_ERROR                 lpdmError);
```

参数

dwCreatorID

通过创建者标识，可确定对象的创建者。

保留值 0 – 10100 以及 11000 – 11100，供内部或特定系统使用。

pData

指向具有变量数据的 MCP_NEWVARIABLE_DATA_4 (页 1875) 结构的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。 出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

变量已创建。

当检查时： 变量尚不存在。

FALSE

错误。

当使用 Errorcode1 = DM_E_ALREADY_EXIST 检查时： 变量存在。

注释

有一个高级函数。

GAPICreateNewVariable5

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_ALREADY_EXIST	已存在要创建的对象

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

GAPICreateNewVariable (页 2065)	创建变量
GAPICreateNewVariable4 (页 2067)	创建变量
GAPICreateNewVariable5 (页 2069)	创建带有创建者 ID 的变量

参见

GAPICreateNewVariable (页 2065)

GAPICreateNewVariable4 (页 2067)

MCP_NEWVARIABLE_DATA_4 (页 1875)

GAPICreateNewVariable5 (页 2069)

3.2.7 处理结构化变量的函数

3.2.7.1 GAPIEnumTypeMembers

使用

该函数返回属于结构化变量的全部变量的名称。

声明

```
BOOL GAPIEnumTypeMembers (
    LPCSTR                      lpszProjectFile,
    LPCSTR                      lpszTypeName,
    DM_ENUM_TYPEMEMBERS_PROC    lpfnCallback,
    LPVOID                      lpvUser,
    LPCMN_ERROR                 lpdmError);
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

如果输入了空字符串，则会对当前打开的项目执行内部 DMEnumOpenedProjects。

在运行系统中只能指定当前打开的项目。系统会拒绝所有其它指定，并显示错误 (DM_E_NOT_CONNECTED)。

lpszTypeName

要列出其变量的结构化变量的类型名称。

lpfnCallback

指向为每个变量调用的回调函数的指针。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

列出结构化变量中的变量。

FALSE

错误。

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DM_ENUM_TYPEMEMBERS_PROC (页 2074)	列出结构化变量中的变量（回调）
GAPIEnumTypeMembersEx (页 2076)	列出结构化变量中的变量
GAPIEnumTypeMembersEx4 (页 2081)	列出结构化变量中的变量

参见

DM_ENUM_TYPEMEMBERS_PROC (页 2074)

GAPIEnumTypeMembersEx (页 2076)

GAPIEnumTypeMembersEx4 (页 2081)

3.2.7.2 DM_ENUM_TYPEMEMBERS_PROC

说明

为了评估系统列出的变量的名称，必须提供 DM_ENUM_TYPEMEMBERS_PROC 类型的回调函数。

声明

```
BOOL ( * DM_ENUM_TYPEMEMBERS_PROC) (
    LPCSTR    lpszMemberName,
    LPVOID    lpvUser );
```

参数

lpszStructTypeName

指向变量名称的指针，这些变量与一个结构化变量相关。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

说明

如有可能，仅在此处复制数据。以下回调中的函数调用类型可导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 的 ODK 函数
- 调用其它枚举的枚举

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

GAPIEnumTypeMembers (页 2072)	列出结构化变量中的变量

参见

GAPIEnumTypeMembers (页 2072)

3.2.7.3 GAPIEnumTypeMembersEx

使用

该函数返回一段描述，其中包含结构化变量中所有变量的预定义值，但是不包含标定数据。

若要读取标定数据，需使用 GAPIEnumTypeMembersEx4 函数。

声明

```
BOOL GAPIEnumTypeMembersEx (
    LPCSTR                          lpszProjectFile,
    LPCSTR                          lpszTypeName,
    DM_ENUM_TYPEMEMBERS_PROC_EX     lpfnCallback,
    LPVOID                          lpvUser,
    LPCMN_ERROR                     lpdmError);
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

如果输入了空字符串，则会对当前打开的项目执行内部 DMEnumOpenedProjects。

在运行系统中只能指定当前打开的项目。系统会拒绝所有其它指定，并显示错误 (DM_E_NOT_CONNECTED)。

lpszTypeName

要列出其变量的结构化变量的类型名称。

客户端上只能列举局部变量。

lpfnCallback

指向为每个变量调用的回调函数的指针。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

结构化变量中的变量已列出

FALSE

错误。

注释

有一个高级函数 GAPIEnumTypeMembersEx4.。

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DM_ENUM_TYPEMEMBERS_PROC_EX (页 2079)	列出结构化变量中的变量（回调）
GAPIEnumTypeMembers (页 2072)	列出结构化变量中的变量
GAPIEnumTypeMembersEx4 (页 2081)	列出结构化变量中的变量

参见

GAPIEnumTypeMembers (页 2072)

DM_ENUM_TYPEMEMBERS_PROC_EX (页 2079)

GAPIEnumTypeMembersEx4 (页 2081)

3.2.7.4 GAPIEnumTypeMembersExStr

声明

```
BOOL GAPIEnumTypeMembersExStr (
    LPCTSTR                             lpszProjectFile,
    LPCTSTR                             lpszTypeName,
    DM_ENUM_TYPEMEMBERS_PROC_EXSTR      lpfnCallback,
    LPVOID                              lpvUser,
    LPCMN_ERROR                         lpdmError);
```

说明

该函数返回一段描述，其中包含结构化变量中所有变量的默认值；但是不返回标定数据。

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称可在运行系统中通过 DMEnumOpenedProjects 或 DMGetRuntimeProject 检索。

在 WinCC 版本 V5.0 SP2 或更高版本中，输入空白字符串将导致对当前打开的项目执行内部 DMEnumOpenedProjects。

在运行系统中，只能对当前打开的项目进行输入。WinCC V5.0 SP2 或更高版本会拒绝其它输入并出现错误 DM_E_NOT_CONNECTED。

lpszTypeName

结构化变量中要枚举的变量类型的名称。

多客户端 (V5) 或客户端 (V6) 上只能枚举局部变量类型。

lpfnCallback

指向为每个变量调用的回调函数的指针。

lpvUser

指向应用程序特定数据的指针。此函数不会评估该指针，但会使其在回调函数中重新可用。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已枚举结构化变量中的变量

FALSE

错误

所需文件

dmclient_exstr.h

dmclient.lib

dmclient.dll

相关函数

DM_ENUM_TYPEMEMBERS_PROC_EXSTR	列出结构化变量中的变量（回调）
GAPIEnumTypeMembersEx4	列出结构化变量中的变量

3.2.7.5 DM_ENUM_TYPEMEMBERS_PROC_EX

说明

为了评估系统列出的变量的描述，必须提供 DM_ENUM_TYPEMEMBERS_PROC_EX 类型的回调函数。

声明

```
BOOL ( * DM_ENUM_TYPEMEMBERS_PROC_EX) (
    LPDM_VARKEY                 lpdmVarKey,
    LPMCP_NEWVARIABLE_DATA_EX   lpdmVarDataEx,
    LPVOID                      lpvUser );
```

参数

lpdmVarKey

指向具有变量密钥（ID 和名称）的第一个 AUTOHOTSPOT 类型结构的指针。

lpdmVarDataEx

指向具有变量说明的 MCP_NEWVARIABLE_DATA_EX (页 1880) 类型结构的指针。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

dmclient.h

相关函数

GAPIEnumTypeMembersEx (页 2076)	列出结构化变量中的变量

参见

GAPIEnumTypeMembersEx (页 2076)

MCP_NEWVARIABLE_DATA_EX (页 1880)

3.2.7.6 GAPIEnumTypeMembersEx4

使用

该函数返回一段完整描述，其中包含结构化变量中所有变量的预定义值。

声明

```
BOOL GAPIEnumTypeMembersEx4 (
    LPCSTR                          lpszProjectFile,
    LPCSTR                          lpszTypeName,
    DM_ENUM_TYPEMEMBERS_PROC_EX4    lpfnCallback,
    LPVOID                          lpvUser,
    LPCMN_ERROR                     lpdmError);
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

如果输入了空字符串，则会对当前打开的项目执行内部 DMEnumOpenedProjects。

在运行系统中只能指定当前打开的项目。拒绝任何其它指定 (DM_E_NOT_CONNECTED)。

lpszTypeName

要列出其变量的结构化变量的类型名称。

客户端上只能列举局部变量。

lpfnCallback

指向为每个变量调用的回调函数的指针。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

结构化变量中的变量已列出

FALSE

错误。

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DM_ENUM_TYPEMEMBERS_PROC_EX4 (页 2082)	列出结构化变量中的变量（回调）
GAPIEnumTypeMembers (页 2072)	列出结构化变量中的变量

参见

GAPIEnumTypeMembers (页 2072)

GAPIEnumTypeMembersEx (页 2076)

DM_ENUM_TYPEMEMBERS_PROC_EX4 (页 2082)

3.2.7.7 DM_ENUM_TYPEMEMBERS_PROC_EX4

说明

为了评估系统列出的变量的描述，必须提供 DM_ENUM_TYPEMEMBERS_PROC_EX4 类型的回调函数。该函数额外指定了标定信息，因此与 DM_ENUM_TYPEMEMBERS_PROC_EX 不同。

声明

```
BOOL ( * DM_ENUM_TYPEMEMBERS_PROC_EX4) (
    LPDM_VARKEY                    lpdmVarKey,
    LPMCP_NEWVARIABLE_DATA_EX4     lpdmVarDataEx,
    LPVOID                         lpvUser );
```

参数

lpdmVarKey

指向具有变量密钥（ID 和名称）的第一个 DM_VARKEY (页 1869) 类型结构的指针。

lpdmVarDataEx

指向具有变量说明的 MCP_NEWVARIABLE_DATA_EX4 (页 1883) 类型结构的指针。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

dmclient.h

相关函数

GAPIEnumTypeMembersEx4 (页 2081)	列出结构化变量中的变量

参见

GAPIEnumTypeMembersEx4 (页 2081)

DM_VARKEY (页 1869)

MCP_NEWVARIABLE_DATA_EX4 (页 1883)

3.2.7.8 GAPIEnumTypes

使用

该函数列出了已组态结构化变量类型的名称和 ID 号。

声明

```
BOOL GAPIEnumTypes (
    LPCSTR                lpszProjectFile,
    DM_ENUM_TYPES_PROC    lpfnCallback,
    LPVOID                lpvUser,
    LPCMN_ERROR           lpdmError);
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

如果输入了空字符串，则会对当前打开的项目执行内部 DMEnumOpenedProjects。

在运行系统中只能指定当前打开的项目。系统会拒绝所有其它指定，并显示错误 (DM_E_NOT_CONNECTED)。

lpfnCallback

指向接收变量类型数据的回调函数的指针。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

结构化变量的类型已列出。

FALSE

错误。

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DM_ENUM_TYPES_PROC (页 2086)	列出结构化变量的类型（回调）

示例

枚举所有结构类型 (页 2134) "DM01.cpp"

参见

DM_ENUM_TYPES_PROC (页 2086)

枚举所有结构类型 (页 2134)

3.2.7.9 DM_ENUM_TYPES_PROC

说明

为了评估系统列出的变量类型，必须提供 DM_ENUM_TYPES_PROC 类型的回调函数。

声明

```
BOOL ( * DM_ENUM_TYPES_PROC) (
    LPCSTR    lpszTypeName,
    DWORD     dwTypeID,
    DWORD     dwCreatorID,
    LPVOID    lpvUser );
```

参数

lpszTypeName

指向变量类型名称的指针。

dwTypeID

dwTypeID 与 GAPICreateType 分配的变量类型的 ID 一致。

dwCreatorID

通过创建者标识，可确定对象的创建者。

保留值 0 – 10100 以及 11000 – 11100，供内部或特定系统使用。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

dmclient.h

相关函数

GAPIEnumTypes (页 2084)	列出结构化变量的类型

示例

枚举所有结构类型 (页 2134) "DM01.cpp"

参见

GAPIEnumTypes (页 2084)

枚举所有结构类型 (页 2134)

3.2.8 处理连接的函数

3.2.8.1 DMEnumConnectionData

使用

该函数确定有关所组态逻辑连接的信息。

声明

```
BOOL DMEnumConnectionData (
    LPCSTR                      lpszProjectFile,
    LPDM_CONNKEY                lpdmConnKey,
    DWORD                       dwItems,
    DM_ENUM_CONNECTION_PROC     lpfnCallback,
    LPVOID                      lpvUser,
    LPCMN_ERROR                 lpdmError);
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称通过 DMEnumOpenedProjects 或运行系统中的 DMGetRuntimeProject 来确定。

lpdmConnKey

指向第一个 DM_CONNKEY (页 1840) 类型结构的指针。这些结构用于指定逻辑连接，将列举这些连接数据。

dwItems

逻辑连接的数量，将列举这些连接数据。

值 0 会触发列出所有连接。

lpfnCallback

指向接收逻辑连接数据的回调函数的指针。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

数据已列出。

FALSE

错误。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DM_ENUM_CONNECTION_PROC (页 2091)	列出连接数据（回调）

示例

枚举所有连接 (页 2136) "DM01.cpp"

OnTestEnumConnectionDataAll (页 2142) "TESTCDoc.cpp"

参见

DM_CONNKEY (页 1840)

DM_ENUM_CONNECTION_PROC (页 2091)

枚举所有连接 (页 2136)

OnTestEnumConnectionDataAll (页 2142)

3.2.8.2 DMEnumConnectionDataExStr

声明

```
BOOL DMEnumConnectionDataExStr (
    LPCTSTR                          lpszProjectFile,
    DM_ENUM_CONNECTION_PROC_EXSTR    lpfnCallback,
    LPVOID                           lpvUser,
    LPDWORD                          lpdwConnectionCount,
    LPCMN_ERROR                      lpdmError);
```

说明

该函数用于确定所有组态逻辑连接。

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

项目文件名称可在运行系统中通过 DMEnumOpenedProjects 或 DMGetRuntimeProject 检索。

lpfnCallback

指向接收逻辑连接数据的回调函数的指针。

lpvUser

指向应用程序特定数据的指针。此函数不会评估该指针，但会使其在回调函数中重新可用。

lpdwConnectionCount

指向返回现有连接数量的“DWORD”的指针。

这样，“lpfnCallback = NULL”首先确定连接需要的内存是多少。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已枚举数据

FALSE

错误

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器

所需文件

dmclient_exstr.h

dmclient.lib

dmclient.dll

相关函数

DM_ENUM_CONNECTION_PROC_EXSTR	列表连接数据（回调）

3.2.8.3 DM_ENUM_CONNECTION_PROC

说明

为了评估系统列出的关于逻辑连接的信息，必须提供 DM_ENUM_CONNECTION_PROC 类型的回调函数。

声明

```
BOOL ( * DM_ENUM_CONNECTION_PROC) (
    LPDM_CONNECTION_DATA    lpdmConData,
    LPVOID                  lpvUser);
```

参数

lpdmConData

指向其中存储逻辑连接数据的 DM_CONNECTION_DATA (页 1839) 类型结构的指针。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

dmclient.h

相关函数

DMEnumConnectionData (页 2087)	列出连接数据

示例

枚举所有连接 (页 2136) "DM01.cpp"

参见

DM_CONNECTION_DATA (页 1839)

DMEnumConnectionData (页 2087)

枚举所有连接 (页 2136)

3.2.9 用于工作平台的函数

3.2.9.1 DMGetOSVersion

使用

确定 PC 上使用的操作系统。

声明

```
DWORD DMGetOSVersion (
    VOID);
```

参数

无。

返回值

返回值指示操作系统：

DM_OS_UNKNOWN	0	未知操作系统
DM_OS_NT	1	Windows NT
DM_OS_32S	2	带 Win 32s 的 Windows 3.x
DM_OS_CHICAGO	3	Windows 95
DM_OS_2000	4	Windows 2000
DM_OS_XP	5	Windows XP
DM_OS_2003	6	Windows 2003

DM_OS_VISTA_32	0x06001	Windows Vista（32 位）
DM_OS_VISTA_SP1_32	0x06011	Windows Vista Service Pack 1（32 位）
DM_OS_VISTA_SP2_32	0x06021	Windows Vista Service Pack 2（32 位）
DM_OS_VISTA_64	0x96001	Windows Vista（64 位）
DM_OS_VISTA_SP1_64	0x96011	Windows Vista Service Pack 1（64 位）
DM_OS_VISTA_SP2_64	0x96021	Windows Vista Service Pack 2（64 位）
DM_OS_S2008_32	0x06003	Windows Server 2008（32 位）
DM_OS_S2008_SP1_32	0x06013	Windows Server 2008 Service Pack 1（32 位）
DM_OS_S2008_SP2_32	0x06023	Windows Server 2008 Service Pack 2（32 位）
DM_OS_S2008_64	0x96003	Windows Server 2008（64 位）
DM_OS_S2008_SP1_64	0x96013	Windows Server 2008 Service Pack 1（64 位）
DM_OS_S2008_SP2_64	0x96023	Windows Server 2008 Service Pack 2（64 位）
DM_OS_W7_32	0x06101	Windows 7（32 位）
DM_OS_W7_SP1_32	0x06111	Windows 7 Service Pack 1（32 位）
DM_OS_W7_64	0x96101	Windows 7（64 位）
DM_OS_W7_SP1_64	0x96111	Windows 7 Service Pack 1（64 位）
DM_OS_S2008R2_32	0x06103	Windows Server 2008 R2（32 位）

DM_OS_S2008R2_SP1_32	0x06113	Windows Server 2008 R2 Service Pack 1（32 位）
DM_OS_S2008R2_64	0x96103	Windows Server 2008 R2（64 位）
DM_OS_S2008R2_SP1_64	0x96113	Windows Server 2008 R2 Service Pack 1（64 位）
DM_OS_W8_32	0x06201	Windows 8（32 位）
DM_OS_W8_64	0x96201	Windows 8（64 位）
DM_OS_S2012_32	0x06203	Windows Server 2012（32 位）
DM_OS_S2012_64	0x96203	Windows Server 2012（64 位）
DM_OS_W81_32	0x06301	Windows 8.1（32 位）
DM_OS_W81_64	0x96301	Windows 8.1（64 位）
DM_OS_S2012R2_32	0x06303	Windows Server 2012 R2（32 位）
DM_OS_S2012R2_64	0x96303	Windows Server 2012 R2（64 位）
DM_OS_W10_64	0x96401	Windows 10（64 位）
DM_OS_S2016_64	0x96403	Windows Server 2016（64 位）

所需文件

dmclient.h

dmclient.lib

dmclient.dll

3.2.9.2 DMGetSystemLocale

使用

获得当前所用组态语言的代码。

声明

```
BOOL DMGetSystemLocale (
    LPDWORD        lpdwLocaleID,
    LPCMN_ERROR    lpdmError);
```

参数

lpdwLocaleID

指向当前组态的语言代码的指针。

可能的返回值是如下语言的代码：

德语	0x0407
英语	0x0409
法语	0x040C

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

当前所用组态语言已确定。

FALSE

错误

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器

所需文件

dmclient.h

dmclient.lib

dmclient.dll

示例

OnTestSystemLocale (页 2149) "TESTCDoc.cpp"

参见

OnTestSystemLocale (页 2149)

3.2.9.3 DMSetLanguage

使用

切换组态语言。

声明

```
BOOL DMSetLanguage (
    DWORD          dwLocaleID,
    LPCMN_ERROR    lpdmError );
```

参数

dwLocaleID

指向要设置语言的代码的指针。可能的值是如下语言的代码：

德语	0x0407
英语	0x0409
法语	0x040C

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

语言已切换。

FALSE

错误。

错误消息

DM_E_NOT_SUPPORTED	请求的服务不可用

所需文件

dmclient.h

dmclient.lib

dmclient.dll

3.2.9.4 DMShowLanguageDialog

使用

该函数用于打开可选择语言的对话框。

声明

```
BOOL DMShowLanguageDialog (
    HWND           hwndParent,
    DWORD          dwFlags,
    DWORD          dwSetLocaleIDs[],
    UINT           uSetIDArraySize,
    LPDWORD        lpdwGetLocaleID,
    LPCMN_ERROR    lpdmError);
```

参数

hwndParent

用作该对话框父窗口的窗口的句柄。

dwFlags

dwFlags 指示在对话框中显示并可进行选择的语言：

DM_LANGDLG_REMOVE	设置	除了 EnumSystemLocales (LCID_SUPPORTED) 数组中的语言之外，所有可用 dwSetLocaleIDs 注册的语言都显示。
	未设置	dwSetLocaleIDs 数组中的所有语言都显示。
DM_LANGDLG_ONLY_PRIMARY	设置	只显示主要语言 (PRIMARY_LANGUAGE)。
	未设置	也显示次要语言。
DM_LANGDLG_NO_NOTIFY		单击“确定”(OK) 关闭对话框时，不通知应用程序

dwSetLocaleIDs

包含语言 ID 的数组，这些语言将在对话框中显示。

uSetIDArraySize

dwSetLocaleIDs 数组的大小。

lpdwGetLocaleID

指向存储所选语言 ID 的 DWORD 的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

单击“确定”(OK) 关闭对话框。

FALSE

出错或单击“取消”(Cancel) 关闭对话框。

错误消息

DM_E_CANCEL	用户已在对话框中选择“取消”(Cancel)

所需文件

dmclient.h

dmclient.lib

dmclient.dll

3.2.10 更新变量的函数

3.2.10.1 DMBeginStartVarUpdate

使用

触发更新请求。使用该函数分配用于标识事务的事务 ID。

声明

```
BOOL DMBeginStartVarUpdate (
    LPDWORD            pdwTAID
    LPCMN_ERROR        lpdmError )
```

参数

pdwTAID

指向包含由数据管理器在成功调用函数之后所分配事务 ID 的变量的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

事务 ID 已分配。

FALSE

错误。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器

所需文件

dmclient.h

dmclient.lib

dmclient.dll

示例

OnTestVariablenBeginstartvarupdate (页 2151) "TESTCDoc.cpp"

参见

OnTestVariablenBeginstartvarupdate (页 2151)

DMEndStartVarUpdate (页 2101)

DMResumeVarUpdate (页 2103)

DMStartVarUpdate (页 2105)

DMStartVarUpdateEx (页 2109)

DMStopVarUpdate (页 2122)

DMSuspendVarUpdate (页 2123)

3.2.10.2 DMEndStartVarUpdate

使用

成功调用函数后，数据管理器会更新变量并将新值转发到所需应用程序。

声明

```
BOOL DMEndStartVarUpdate (
    DWORD          dwTAID,
    LPCMN_ERROR    lpdmError );
```

参数

dwTAID

dwTAID 包含因调用 DMBeginStartVarUpdate 函数分配的事务 ID。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

更新已启动。

FALSE

错误。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_INVALID_TAID	无效事务 ID

所需文件

dmclient.h

相关函数

DMBeginStartVarUpdate (页 2100)	触发更新请求
DMStartVarUpdate (页 2105)	定义更新变量

示例

OnTestVariablenEndstartvarupdate (页 2152) "TESTCDoc.cpp"

参见

OnTestVariablenEndstartvarupdate (页 2152)

DMBeginStartVarUpdate (页 2100)

DMStartVarUpdate (页 2105)

3.2.10.3 DMResumeVarUpdate

使用

在调用参考计数器之后事务达到值 0 时，开始更新该事务定义的所有变量。

该函数与 DMSuspendVarUpdate 相对，应总是成对调用两个函数。

声明

```
BOOL DMResumeVarUpdate (
    DWORD          dwTAID,
    LPCMN_ERROR    lpdmError);
```

参数

dwTAID

dwTAID 包含因调用 DMBeginStartVarUpdate 函数分配的事务 ID。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

继续更新。

FALSE

错误。

注释

事务有一个参考计数器，每次调用 DMSuspendVarUpdate 函数时递增。如果应该继续更新变量，则必须多次调用 DMResumeVarUpdate 直到参考计数器返回到 0 位置。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_INVALID_TAID	无效事务 ID

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DMBeginStartVarUpdate (页 2100)	触发更新请求
DMSuspendVarUpdate (页 2123)	停止更新变量

示例

OnTestVariablenResumevarupdate (页 2163) "TESTCDoc.cpp"

参见

DMBeginStartVarUpdate (页 2100)

DMSuspendVarUpdate (页 2123)

OnTestVariablenResumevarupdate (页 2163)

3.2.10.4 DMStartVarUpdate

使用

该函数用于指定将要由数据管理器更新的变量。执行更新时，数据管理器会将新的变量值传递给回调函数中的应用程序。

DMStartVarUpdateEx 是更高级的函数，可以在 DM_NOTIFY_VARIABLEEX_PROC 回调中另外返回 DM_VAR_UPDATE_STRUCTEX 结构中的质量代码。

声明

```
BOOL DMStartVarUpdate (
    DWORD                       dwTAID,
    LPDM_VARKEY                 lpdmVarKey,
    DWORD                       dwItems,
    DWORD                       dwCycle,
    DM_NOTIFY_VARIABLE_PROC     lpfnVariable,
    LPVOID                      lpvUser,
    LPCMN_ERROR                 lpdmError);
```

参数

dwTAID

dwTAID 包含因调用 DMBeginStartVarUpdate 函数分配的事务 ID。

lpdmVarKey

指向第一个 DM_VARKEY (页 1869) 类型结构的指针，可通过该类型结构指定要更新的变量。

dwItems

传递的 DM_VARKEY 结构中的要更新的变量数。

dwCycle

dwCycle 中定义的更新周期将应用于由 lpdmVarKey 指定的所有变量。升级周期由更新周期列表中的条目索引定义。

“改变时”	非周期性	索引：0
“250 ms”	250	索引：1
“500 ms”	500	索引：2
“1 s”	1000	索引：3

“2 s”	2000	索引：4
“5 s”	5000	索引：5
“10 s”	10000	索引：6
“1 min”	60000	索引：7
“5 min”	300000	索引：8
“10 min”	600000	索引：9
“1 h”	3600000	索引：10
“用户周期 1”	（例如 2000）	索引：11
“用户周期 2”	（例如 3000）	索引：12
“用户周期 3”	（例如 4000）	索引：13
“用户周期 4”	（例如 5000）	索引：14
“用户周期 5”	（例如 10000）	索引：15

lpfnVariable

指向 DM_NOTIFY_VARIABLE_PROC 回调函数以通过数据管理器传送数据的指针。

如果 lpfnVariable == NULL，则至少在要求的周期内对数据管理器的过程映像执行更新，所达到的程度为：基于当前系统负载、计算机性能和耦合介质等物理条件可行。当前值的确定需要应用程序采用调用 DMGetValue 函数的方式独立执行。

当程序报告通知例程时，必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

在极少数情况下，在函数调用返回之前已经传送了“通知”。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

更新变量已指定。

FALSE

错误。

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DMBeginStartVarUpdate (页 2100)	触发更新请求
DM_NOTIFY_VARIABLE_PROC (页 2107)	指定更新变量（回调）

参见

DMEndStartVarUpdate (页 2101)

DM_VARKEY (页 1869)

DMBeginStartVarUpdate (页 2100)

DM_NOTIFY_VARIABLE_PROC (页 2107)

3.2.10.5 DM_NOTIFY_VARIABLE_PROC

说明

为了评估系统确定的数据，必须提供 DM_NOTIFY_VARIABLE_PROC 类型的回调函数。

声明

```
BOOL ( * DM_NOTIFY_VARIABLE_PROC) (
    DWORD                    dwTAID,
    LPDM_VAR_UPDATE_STRUCT   lpdmvus,
    DWORD                    dwItems,
    LPVOID                   lpvUser);
```

参数

dwTAID

数据管理器针对调用函数所分配的事务 ID。

lpdmvus

指向包含所需变量值的第一个 DM_VAR_UPDATE_STRUCT (页 1854) 类型结构的指针。

dwItems

lpdmvus 中传送的结构数（与返回的变量值的数目相对应）。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

返回值视执行情况而定

注释

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

如果程序声明了“通知”例程，必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

在极少数情况下，在函数调用返回之前已经传送了“通知”。

所需文件

dmclient.h

相关函数

DMGetValueWait (页 1991)	获取更新的变量值
DMStartVarUpdate (页 2105)	定义更新变量

参见

DMStartVarUpdate (页 2105)

DM_VAR_UPDATE_STRUCT (页 1854)

DMGetValueWait (页 1991)

3.2.10.6 DMStartVarUpdateEx

使用

该函数用于指定将要由数据管理器更新的变量。执行更新时，数据管理器在该回调函数中将新的变量值传递给应用程序。

相较 DMStartVarUpdate 而言，该函数还会在 DM_NOTIFY_VARIABLEEX_PROC 回调中返回 DM_VAR_UPDATE_STRUCTEX 结构中的质量代码。

声明

```
BOOL DMStartVarUpdateEx (
    DWORD                       dwTAID,
    LPDM_VARKEY                 lpdmVarKey,
    DWORD                       dwItems,
    DWORD                       dwCycle,
    DM_NOTIFY_VARIABLEEX_PROC   lpfnVariable,
    LPVOID                      lpvUser,
    LPCMN_ERROR                 lpdmError);
```

参数

dwTAID

dwTAID 包含因调用 DMBeginStartVarUpdate 函数分配的事务 ID。

lpdmVarKey

指向第一个 DM_VARKEY (页 1869) 类型结构的指针，可通过该类型结构指定要更新的变量。

dwItems

传递的 DM_VARKEY 结构中的要更新的变量数。

dwCycle

此处定义的更新周期将应用于由 lpdmVarKey 指定的所有变量。升级周期由更新周期列表中的条目索引定义。

“改变时”	非周期性	索引：0
“250 ms”	250	索引：1
“500 ms”	500	索引：2
“1 s”	1000	索引：3
“2 s”	2000	索引：4
“5 s”	5000	索引：5
“10 s”	10000	索引：6
“1 分钟”	60000	索引：7
“5 min”	300000	索引：8
“10 min”	600000	索引：9
“1 h”	3600000	索引：10
“用户周期 1”	（例如 2000）	索引：11
“用户周期 2”	（例如 3000）	索引：12
“用户周期 3”	（例如 4000）	索引：13
“用户周期 4”	（例如 5000）	索引：14
“用户周期 5”	（例如 10000）	索引：15

lpfnVariable

指向 DM_NOTIFY_VARIABLEEX_PROC 回调函数以通过数据管理器传送数据的指针。

如果 lpfnVariable == NULL，则至少在要求的周期内对数据管理器的过程映像执行更新，所达到的程度为：基于当前系统负载、计算机性能和耦合介质等物理条件可行。当前值的确定需要应用程序采用调用 DMGetValue 函数的方式独立执行。

当程序报告通知例程时，必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

更新变量已指定。

FALSE

错误。

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DMBeginStartVarUpdate (页 2100)	触发更新请求
DM_NOTIFY_VARIABLEEX_PROC (页 2119)	指定更新变量（回调）

参见

DM_VARKEY (页 1869)

DMBeginStartVarUpdate (页 2100)

DM_NOTIFY_VARIABLEEX_PROC (页 2119)

3.2.10.7 DMStartVarUpdateExStr

声明

```
BOOL DMStartVarUpdateExStr (
    DWORD                              dwTAID,
    DWORD                              dwFlags,
    LPVARIANT                          lpvdmVarKey,
    LPVARIANT                          lpvCookie,
    DWORD                              dwCycle,
    DM_NOTIFY_VARIABLE_PROC_EXSTR   lpfnVariable,
    LPVOID                             lpvUser,
    LPCMN_ERROR                        lpdmError);
```

说明

该函数用于指定将要由数据管理器更新的变量。执行更新时，数据管理器在该回调函数中将新的变量值传递给应用程序。

参数

dwTAID

dwTAID 包含调用 DMBeginStartVarUpdate 函数时分配的事物 ID。

dwFlags

如果需要以 VT_LPSTR 格式在 DM_NOTIFY_VARIABLE_PROC_EXSTR 的 DM_VAR_UPDATE_STRUCT_EXSTR 结构的 VARIANT 中返回变量名称，则可在此指定 DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR 标志。

lpvdmVarKey

指向类型为 VT_ARRAY | VT_VARIANT 的 VARIANT 的指针，用于传递可识别要更改的变量的变量列表。

在 VT_I4 类型的相关列表元素和 VT_BSTR 类型的变量名称中输入 TagID。此外，还可输入 VT_LPSTR (PROPVARIANT) 作为已分配的 ASCII 字符串进行传递，例如，用以传递来自脚本的常量名称。VT_LPSTR 随后会在内部转换为所需的 VT_BSTR 类型。

对于 WinCC 版本 5.0 或更高版本中的多客户端项目，此时可能需要添加服务器前缀（参见“项目类型和版本”）。

lpvCookie

指向各变量用户相关数据附加列表的 VARIANT 的指针。

该指针可用来替代之前 DM_VARKEY 的 lpvUserData，并且也会返回到每个变量的 DM_VAR_UPDATE_STRUCT_EXSTR 结构中。

dwCycle

此处定义的更新周期将应用于通过 lpvdmVarKey 指定的所有变量。该升级周期通过更新周期列表中包含的条目索引定义。

“改变时”：	非周期性	索引：0
“250 毫秒”	250	索引：1
“500 毫秒”	500	索引：2
“1 秒”	1000	索引：3
“2 秒”	2000	索引：4
“5 秒”	5000	索引：5
“10 秒”	10000	索引：6
“1 分钟”	60000	索引：7
“5 分钟”	300000	索引：8
“10 分钟”	600000	索引：9
“1 小时”	3600000	索引：10
“用户周期 1”	（例如 2000）	索引：11
“用户周期 2”	（例如 3000）	索引：12
“用户周期 3”	（例如 4000）	索引：13
“用户周期 4”	（例如 5000）	索引：14
“用户周期 5”	（例如 10000）	索引：15

lpfnVariable

指向 DM_NOTIFY_VARIABLEEX_PROC 回调函数以通过数据管理器传送数据的指针。

如果“lpfnVariable”为“NULL”，则至少在要求的周期内对数据管理器的过程映像执行更新，所达到的程度为：基于当前系统负载、计算机性能和耦合介质等物理条件可行；但是，每个值都需要通过在应用程序中调用 DMGetValue 函数来单独确定。

如果程序请求通知例程，则必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

lpvUser

指向应用程序特定数据的指针。此函数不会评估该指针，但会使其在回调函数中重新可用。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已指定更新变量

FALSE

错误

所需文件

dmclient_exstr.h

dmclient.lib

dmclient.dll

相关函数

DMBeginStartVarUpdate	触发更新请求

示例

脚本示例“按钮 DMStartVarUpdateExStr”：

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
#pragma code ("OleAut32.dll")
//#include "OleAuto.h"
SAFEARRAY * SafeArrayCreateVector(VARTYPE vt, long lLbound,
unsigned int cElements );
HRESULT SafeArrayPtrOfIndex(SAFEARRAY FAR* psa, long FAR*
rgIndices, void HUGEP* FAR* ppvData );
HRESULT SafeArrayLock(SAFEARRAY FAR* psa);
HRESULT SafeArrayLock(SAFEARRAY FAR* psa);
#pragma code()
```

```
extern BOOL DM_NOTIFY_VARIABLE_PROC_EXSTR
DM_NotifyVariableProcExStr_VarUpdate(
            DWORD                                dwTAID,
    LPDM_VAR_UPDATE_STRUCT_EXSTRA lpdmvus,
    DWORD                         dwItems,
    LPVOID                        lpvUser);

VARIANT vVarKey;
VARIANT* pvElem;
SAFEARRAY* parrayKeys;
HRESULT hr;
CMN_ERROR err;
BOOL bRet;
long lInx;
DWORD dwTAID;
DWORD dwFlags;
DWORD dwCycle;


memset(&err, 0, sizeof(err));
bRet = FALSE;
parrayKeys = NULL;
lInx = 0L;
dwFlags = DM_FLAG_RETURN_PROPVARIANT_VT_LPSTR;
//dwCycle = 3; /* 1sec*/
dwCycle = 0; /* on change*/


printf("\r\n\r\n########## enter Test with DMStartVarUpdateExStr
##########");

dwTAID = GetTagDWord("dwUpdTAID");

if (dwTAID)
{
    printf("\r\nStartVarUpdate always running.Stop it before
start again!");
    printf("\r\n########## exit Test with DMStartVarUpdateExStr
(nothing done) ##########\r\n");
    return;
}

VariantInit(&vVarKey);
parrayKeys = SafeArrayCreateVector(VT_VARIANT, 0L, 4);
vVarKey.vt = VT_ARRAY | VT_VARIANT;
vVarKey.u.parray = parrayKeys;

SafeArrayLock(parrayKeys);
lInx = 0L;
hr = SafeArrayPtrOfIndex(parrayKeys, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = "dwVal_1";
```

```
lInx = 1L;
hr = SafeArrayPtrOfIndex(parrayKeys, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = "dwVal_2";
lInx = 2L;
hr = SafeArrayPtrOfIndex(parrayKeys, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = "dwVal_3";
lInx = 3L;
hr = SafeArrayPtrOfIndex(parrayKeys, &lInx, &pvElem);
pvElem->vt = VT_LPSTR;
pvElem->u.pbVal = "dwVal_4";
SafeArrayUnlock(parrayKeys);

bRet = DMBeginStartVarUpdate(&dwTAID, &err);
if (FALSE == bRet)
{
    printf("\r\n  error DMBeginStartVarUpdate,
err=%ld,%ld,%ld,%ld,%ld,[%s]",
           err.dwError1, err.dwError2, err.dwError3,
           err.dwError4, err.dwError5, err.szErrorText);
}
else
{
    printf("\r\n  DMBeginStartVarUpdate  (dwTAID=%ld) OK.",
dwTAID);
}
if (dwTAID)
{
    SetTagDWord("dwUpdTAID",dwTAID);

    memset(&err, 0, sizeof(err));
    bRet = DMStartVarUpdateExStr(dwTAID, dwFlags, &vVarKey,
dwCycle,

DM_NotifyVariableProcExStr_VarUpdate, NULL, &err);
    if (FALSE == bRet)
    {
        printf("\r\n  error DMStartVarUpdateExStr
(dwTAID=%ld):err=%ld,%ld,%ld,%ld,%ld,[%s]",
               dwTAID, err.dwError1, err.dwError2, err.dwError3,
                    err.dwError4, err.dwError5, err.szErrorText);
    }
    else
    {
        printf("\r\n  DMStartVarUpdateExStr  (dwTAID=%ld) OK.",
dwTAID);
    }

    memset(&err, 0, sizeof(err));
    bRet = DMEndStartVarUpdate(dwTAID, &err);
    if (FALSE == bRet)
```

```
    {
        printf("\r\n  error DMEndStartVarUpdate
(dwTAID=%ld):err=%ld,%ld,%ld,%ld,%ld,[%s]",
               dwTAID, err.dwError1, err.dwError2, err.dwError3,
                    err.dwError4, err.dwError5, err.szErrorText);
    }
    else
    {
        printf("\r\n  DMEndStartVarUpdate  (dwTAID=%ld) OK.",
dwTAID);
    }
}
else
{
    printf("\r\n  dwTAID == 0L ???");
}


printf("\r\n########## exit Test with DMStartVarUpdateExStr
(dwTAID=%ld) ##########\r\n", dwTAID);

}
```

脚本示例“项目函数 DM_NotifySelectVarProcA”：

```
#pragma code ("OleAut32.dll")
//#include "OleAuto.h"
HRESULT VariantChangeType( VARIANTARG FAR* pvargDest, VARIANTARG
FAR* pvarSrc, unsigned short wFlags, VARTYPE vt);
#pragma code()

BOOL DM_NotifyVariableProcExStr_VarUpdate(
            DWORD                               dwTAID,
    LPDM_VAR_UPDATE_STRUCT_EXSTRA lpdmvus,
    DWORD                         dwItems,
    LPVOID lpvUser)
{
BOOL bRet = FALSE;
int i = 0;
HRESULT hr = 0L;

printf("\r\n*** DM_NotifyVariableProcExStr_VarUpdate entry
(dwTAID=%ld) ***", dwTAID);

for (i = 0; i < dwItems; i++)
{
    hr = VariantChangeType((VARIANTARG*)&(lpdmvus[i].vdmValue),
                           (VARIANTARG*)&(lpdmvus[i].vdmValue), 0,
VT_R8);
    if (VT_LPSTR == lpdmvus[i].vdmVarKey.vt)
    {
```

```
        printf("\r\n    [%d]:{VarKey=[%s]", i,
(LPCSTR)lpdmvus[i].vdmVarKey.u.pbVal);
    }
    else if (VT_I4 ==  lpdmvus[i].vdmVarKey.vt)
    {
        printf("\r\n    [%d]:{VarKey=[%ld]", i,
lpdmvus[i].vdmVarKey.u.lVal);
    }
    else
    {
        printf("\r\n    [%d]:{unexpected VarKey.vt=[%d]",
lpdmvus[i].vdmVarKey.vt);
    }
    printf(", dmValue=[%g]", lpdmvus[i].vdmValue.u.dblVal);
    printf(", dwState=[%ld]", lpdmvus[i].dwState);
    printf(", dwType=[%ld]", lpdmvus[i].dmTypeRef.dwType);
    printf(", dwSize=[%ld]", lpdmvus[i].dmTypeRef.dwSize);
    if (lpdmvus[i].dmTypeRef.lpszTypeName)
    {
        printf(", lpszTypeName=[%s]",
lpdmvus[i].dmTypeRef.lpszTypeName);
        printf(", dwNameCharCount=[%ld]",
lpdmvus[i].dmTypeRef.dwNameCharCount);
    }
    else
    {
        printf(", lpszTypeName=[NULL]");
    }
    printf("}");
}

printf("\r\n*** DM_NotifyVariableProcExStr_VarUpdate exit
(dwTAID=%ld)***", dwTAID);
return bRet;
}
```

脚本示例“按钮 DMStopVarUpdate”:

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
lpszPropertyName)
{
DWORD dwTAID;
CMN_ERROR err;
BOOL bRet;

dwTAID = GetTagDWord("dwUpdTAID");
memset(&err, 0, sizeof(err));
bRet = FALSE;

if (dwTAID)
```

```
{
    bRet = DMStopVarUpdate(dwTAID, &err);
    if (FALSE == bRet)
    {
        printf("\r\n!!!   error DMStopVarUpdate
(dwTAID=%ld):err=%ld,%ld,%ld,%ld,%ld,[%s]   !!!\r\n",
                dwTAID, err.dwError1, err.dwError2, err.dwError3,
                     err.dwError4, err.dwError5, err.szErrorText);
        memset(&err, 0, sizeof(err));
        bRet = DMStopAllUpdates(&err);
        if (FALSE == bRet)
        {
            printf("\r\n!!!   error DMStopAllUpdates
(dwTAID=%ld):err=%ld,%ld,%ld,%ld,%ld,[%s]   !!!\r\n",
                  dwTAID, err.dwError1, err.dwError2, err.dwError3,
                               err.dwError4, err.dwError5,
err.szErrorText);
        }
        else
        {
            printf("\r\n  DMStopAllUpdates OK.");
        }
    }
    else
    {
       printf("\r\n  DMStopVarUpdate  (dwTAID=%ld) OK.", dwTAID);
    }
}

dwTAID = 0L;

SetTagDWord("dwUpdTAID",dwTAID);
}
```

3.2.10.8 DM_NOTIFY_VARIABLEEX_PROC

说明

为了评估系统确定的数据，必须提供 DM_NOTIFY_VARIABLEEX_PROC 类型的回调函数。

声明

```
BOOL ( * DM_NOTIFY_VARIABLEEX_PROC) (
    DWORD                       dwTAID,
    LPDM_VAR_UPDATE_STRUCTEX  lpdmvus,
    DWORD                       dwItems,
    LPVOID                      lpvUser);
```

参数

dwTAID

数据管理器针对调用函数所分配的事务 ID。

lpdmvus

指向包含所需变量值的第一个 DM_VAR_UPDATE_STRUCTEX (页 1856) 类型结构的指针。

dwItems

lpdmvus 中传送的结构数（与返回的变量值的数目相对应）。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

返回值视执行情况而定

注释

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

如果程序声明了“通知”例程，必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

在极少数情况下，在函数调用返回之前已经传送了“通知”。

所需文件

dmclient.h

相关函数

DMGetValueWaitEx (页 1994)	获取更新的变量值
DMStartVarUpdateEx (页 2109)	定义更新变量

参见

DMStartVarUpdateEx (页 2109)

DM_VAR_UPDATE_STRUCTEX (页 1856)

DMGetValueWaitEx (页 1994)

3.2.10.9 DMStopAllUpdates

使用

停止应用系统要求的所有变量更新。关闭相应模块后，最好调用该函数。

声明

```
BOOL DMStopAllUpdates (
    LPCMN_ERROR    lpdmError);
```

参数

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

变量更新已停止。

FALSE

错误。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器

所需文件

dmclient.h

dmclient.lib

dmclient.dll

示例

OnTestVariablenStopallupdates (页 2166) "TESTCDoc.cpp"

参见

OnTestVariablenStopallupdates (页 2166)

3.2.10.10 DMStopVarUpdate

使用

停止特定事务的变量更新。

声明

```
BOOL DMStopVarUpdate (
    DWORD          dwTAID,
    LPCMN_ERROR    lpdmError);
```

参数

dwTAID

dwTAID 包含因调用 DMBeginStartVarUpdate 函数分配的事务 ID。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

变量更新已停止。

FALSE

错误。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_INVALID_TAID	无效事务 ID

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DMBeginStartVarUpdate (页 2100)	触发更新请求

示例

OnTestVariablenStopvarupdate (页 2167) "TESTCDoc.cpp"

参见

OnTestVariablenStopvarupdate (页 2167)

DMBeginStartVarUpdate (页 2100)

3.2.10.11 DMSuspendVarUpdate

使用

中断事务定义的所有变量的更新。

将在数据管理器的过程映像内持续更新的变量。但是，变量值不再传送到应用程序。

声明

```
BOOL DMSuspendVarUpdate (
    DWORD          dwTAID,
    LPCMN_ERROR    lpdmError);
```

参数

dwTAID

dwTAID 包含因调用 DMBeginStartVarUpdate 函数分配的事务 ID。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

更新已暂停。

FALSE

错误。

注释

事务具有参考计数器，因此可以针对一个事务多次调用 DMSuspendVarUpdate 函数。如果事务已处于“暂停”(SUSPENDED) 状态，则仅参考计数器的值相应递增。

要继续更新，必须反复调用 DMResumeVarUpdate 直到参考计数器返回到 0 位置。

错误消息

DM_E_NOT_CONNECTED	未连接到数据管理器
DM_E_INVALID_TAID	无效事务 ID

所需文件

dmclient.h

dmclient.lib

dmclient.dll

相关函数

DMBeginStartVarUpdate (页 2100)	触发更新请求
DMResumeVarUpdate (页 2103)	继续更新变量

示例

OnTestVariablenSuspendvarupdate (页 2168) "TESTCDoc.cpp"

参见

DMResumeVarUpdate (页 2103)

DMBeginStartVarUpdate (页 2100)

OnTestVariablenSuspendvarupdate (页 2168)

3.2.11 示例

3.2.11.1 连接到 DM

示例

```
// ========================================================================
// ========================================================================
// Desc. : Modul with examples for Data-Manager
// ************************************************************************
#include "stdafx.h" // if MFC classes
// #include "odkapi.h" // if console application
#include <time.h>
TCHAR g_szProjectFile[255] = {0};
TCHAR g_szDSNName[255] = {0};
#include "DM01.h"

//{{ODK_EXAMPLE}Connection to DM (MCP)}
//{{FUNCTION}DMGetConnectionState (MCP)}
//{{FUNCTION}DMConnect (MCP)}
//{{FUNCTION}DM_NOTIFY_PROC (MCP)}
//{{FUNCTION}DMDisConnect (MCP)}
//{{FUNCTION}(END)}

// ========================================================================
// Function: MyDMConnect(void) ODK DM CS
// ========================================================================
BOOL MyDMConnect(void)
{
    CMN_ERROR Error;
    BOOL ret = FALSE;
    TCHAR szText[255];
    TCHAR szAppName[255];
    VOID* pvUser = AfxGetApp();
    _tcsncpy_s(szAppName, _countof(szAppName), _T("MyODKApp_23"), _TRUNCATE);
    memset(&Error, 0,sizeof(CMN_ERROR));
    ret = DMGetConnectionState(&Error);
    if(FALSE == ret) // not connected
    {
        memset(&Error, 0,sizeof(CMN_ERROR));
        ret = DMConnect(szAppName, MyDMNotifyCallback, pvUser, &Error);
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in DMConnect: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
```

```
        {
            _sntprintf_s(szText,_countof(szText), _TRUNCATE, _T("DMConnect"));
        }
        ODKTrace(szText);
}
    else // already connected
    {
        _sntprintf_s(szText,_countof(szText), _TRUNCATE, _T("DMGetConnectionState: OK"));
    }
    else // already connected
    {
        _sntprintf_s(szText,_countof(szText), _TRUNCATE, _T("DMGetConnectionState: OK"));
        ODKTrace(szText);
    }
    return(ret);
}

// IMPLEMENTATION
// ========================================================================
// Function: MyDMGetConnectionState(void) ODK DM CS
// ========================================================================
BOOL MyDMGetConnectionState(void)
{
    CMN_ERROR Error;
    BOOL ret = FALSE;
    TCHAR szText[255];
    TCHAR szAppName[255];
    VOID* pvUser = AfxGetApp();
    _tcsncpy_s(szAppName, _countof(szAppName), _T("MyODKApp_23"), _TRUNCATE);
    memset(&Error, 0,sizeof(CMN_ERROR));
    ret = DMGetConnectionState(&Error);
    if(FALSE == ret) // not connected
    {
        memset(&Error, 0,sizeof(CMN_ERROR));
        ret = DMConnect(szAppName, MyDMNotifyCallback, pvUser, &Error);
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in DMConnect: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText,_countof(szText), _TRUNCATE, _T("DMConnect"));
        }
        ODKTrace(szText);
        //printf("%s\r\n",szText);
    }
    else // already connected
    {
        _sntprintf_s(szText,_countof(szText), _TRUNCATE, _T("DMGetConnectionState: OK"));
```

```
        ODKTrace(szText);
    }
    return(ret);
}

// ========================================================================
// Function: MyDMDisConnect(void) ODK DM CS
// ========================================================================
BOOL MyDMDisConnect(void)
{
    CMN_ERROR Error;
    BOOL ret = FALSE;
    TCHAR szText[255];
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = DMDisConnect(&Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in DMConnect: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText,_countof(szText), _TRUNCATE, _T("DMDisconnect"));
    }
    ODKTrace(szText);
    return(ret);
}

// ========================================================================
// Function: MyDMNotifyCallback
// ========================================================================
BOOL MyDMNotifyCallback(DWORD dwNotifyClass, DWORD dwNotifyCode, LPBYTE lpbyData,
                DWORD dwItems, LPVOID lpvUser)
{
    lpvUser;
    lpbyData;
    TCHAR szText[255];
    _sntprintf_s(szText ,_countof(szText), _TRUNCATE, _T("**DMNotifyCallback**"));
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  NotifyClass =
0x%08X"),dwNotifyClass);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  NotifyCode = 0x%08X"),
dwNotifyCode);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  Items = %d"), dwItems);
    ODKTrace(szText);
    //printf("%s\r\n",szText);
_sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("********************"));
    ODKTrace(szText);
```

```
    return(TRUE);
}

//{{ODK_EXAMPLE}(END)}
```

参见

DMConnect (页 1909)

DMDisconnect (页 1916)

DMGetConnectionState (页 1929)

3.2.11.2 变量的枚举数据

示例

```
//{{ODK_EXAMPLE}Enum Data of Tags (MCP)}
//{{FUNCTION}DMEnumVarData4 (MCP)}
//{{FUNCTION}DM_ENUM_VARIABLE_PROC4 (MCP)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: MyDMEnumVarData4(void) ODK DM CS
// ========================================================================
// Desc. :
//-------------------------------------------------------------------------------
BOOL MyDMEnumVariable4Callback(LPDM_VARKEY lpdmVarKey, LPDM_VARIABLE_DATA4 lpdmVarData,
LPVOID lpvUser)
{
    lpvUser;
    TCHAR szText[255];
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("DMEnumVariableCallback"));
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...szName=%s
dwVarProperty=0x%04X"),lpdmVarKey->szName, lpdmVarData->dwVarProperty);
    ODKTrace(szText);
    return TRUE;
}
    void MyDMEnumVarData4()
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    DM_VARKEY dmVarKey[2];
    TCHAR szText[255];
    CMN_ERROR Error;
    VOID* pUser = NULL;
    memset(&Error,0,sizeof(CMN_ERROR));
    memset(&dmVarKey, 0,2 * sizeof(DM_VARKEY));
    dmVarKey[0].dwKeyType = DM_VARKEY_NAME;
_tcsncpy_s(dmVarKey[0].szName, _countof(dmVarKey[0].szName), _T("INT_TEST_VAR"),
_TRUNCATE);
    dmVarKey[1].dwKeyType = DM_VARKEY_NAME;
_tcsncpy_s(dmVarKey[1].szName, _countof(dmVarKey[1].szName), _T("EXT_TEST_VAR"),
_TRUNCATE);
    MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
    ret = DMEnumVarData4 (/*PROJ_PATH*/g_szProjectFile,
                &dmVarKey[0],
                2,
                MyDMEnumVariable4Callback,
                pUser,
                &Error);
    if(FALSE == ret)
```

```
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in DMEnumVarData4: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMEnumVarData4 (页 1965)

DM_ENUM_VARIABLE_PROC4 (页 1967)

3.2.11.3 枚举打开项目

示例

```
//{{ODK_EXAMPLE}Enum open projects (MCP)}
//{{FUNCTION}DMEnumOpenedProjects (MCP)}
//{{FUNCTION}DM_ENUM_OPENED_PROJECTS_PROC (MCP)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: MyDMEnumOpenedProjects(void) ODK DM CS
// ========================================================================
// Desc. : Inquire project informations
// ========================================================================
BOOL MyDMEnumOpenProjectsCallback( LPDM_PROJECT_INFO lpInfo, LPVOID lpvUser )
{
    lpvUser;
    // this callback is only called once by every call of DMEnumOpenedProjects
    // because, there can only opened one project at the same time
    TCHAR szText[255];
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("**DMEnumOpenProjects**"));
    ODKTrace(szText);
    // copy the project file name to g_szProjectFile for global use
    // many API-functions need this name to select the project database
    _tcsncpy_s(g_szProjectFile, _countof(g_szProjectFile), lpInfo->szProjectFile,
_TRUNCATE);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  ProjectFile = %s"),lpInfo-
>szProjectFile);
    ODKTrace(szText);

    _tcsncpy_s(g_szDSNName, _countof(g_szDSNName), lpInfo->szDSNName, _TRUNCATE);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  DSNName = %s"),lpInfo-
>szDSNName);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("  DataLocale = 0x%08X"),lpInfo-
>dwDataLocale);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("**********************"));
    ODKTrace(szText);
    return( TRUE ); // only 1. element
}

BOOL MyDMEnumOpenedProjects(void)
{
    CMN_ERROR Error;
    BOOL ret= FALSE;
    DWORD dwItems;
    TCHAR szText[255];
    VOID* pvUser = AfxGetApp();
    memset(&Error, 0, sizeof(CMN_ERROR));
```

```
    ret = MyDMConnect();
    if(TRUE == ret)
    {
        ret = DMEnumOpenedProjects(&dwItems, MyDMEnumOpenProjectsCallback, pvUser, &Error);
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
DMEnumOpenedProjects: E1= 0x%08lx ; E2= 0x%08lx ; %s"), %s",
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("DMEnumOpenedProjects"));
        }
        ODKTrace(szText);
    }
    return(ret);
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMEnumOpenedProjects (页 1938)

DM_ENUM_OPENED_PROJECTS_PROC (页 1940)

3.2.11.4 枚举所有结构类型

示例

```
//{{ODK_EXAMPLE}Enumerate all structured types (MCP)}
//{{FUNCTION}GAPIEnumTypes (MCP)}
//{{FUNCTION}DM_ENUM_TYPES_PROC (MCP)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: MyGAPIEnumTypes(void) ODK DM CS
// ========================================================================
// Desc. :
//-------------------------------------------------------------------------------
BOOL MyDMEnumTypeCallback(LPCSTR lpszTypeName,DWORD dwTypeID,
                DWORD dwCreatorID, LPVOID lpvUser )
{
    lpvUser;
    TCHAR szText[255];
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("**DMEnumTypeCallback**"));
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TypeName = %s TypeID = 0x%08x
CreatorID = %d"),
        lpszTypeName,dwTypeID,dwCreatorID);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("**********************"));
    ODKTrace(szText);
    return TRUE;
}

void MyGAPIEnumTypes()
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    CMN_ERROR Error;
    BOOL ret = FALSE;
    TCHAR szText[255];
    TCHAR szProjectFile[255];
    VOID* pvUser = NULL;
    ret = FALSE;
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = MyDMGetConnectionState(); //check the connection state to DM
    if(FALSE != ret)
    {
        MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
        _tcsncpy_s(szProjectFile, _countof(szProjectFile), /*PROJ_PATH*/g_szProjectFile,
_TRUNCATE);
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = GAPIEnumTypes(szProjectFile,
                MyDMEnumTypeCallback,
                pvUser,
```

```
                &Error);
        if(FALSE == ret)
        {
           _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in DMEnumTypes: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText,_countof(szText), _TRUNCATE, _T("DMEnumTypes"));
        }
        ODKTrace(szText);
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

GAPIEnumTypes (页 2084)

DM_ENUM_TYPES_PROC (页 2086)

3.2.11.5 枚举所有连接

示例

```
//{{ODK_EXAMPLE}Enumerate all connections (MCP)}
//{{FUNCTION}DMEnumConnectionData (MCP)}
//{{FUNCTION}DM_ENUM_CONNECTION_PROC (MCP)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: MyDMEnumConnectionData(void) ODK DM CS
// ========================================================================
// Desc. :
//-------------------------------------------------------------------------------
BOOL MyDMEnumConnectionCallback(LPDM_CONNECTION_DATA lpdmConData, LPVOID lpvUser)
{
    lpvUser;
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("DMEnumConnectionCallback"));
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...Connection = %s"),lpdmConData-
>szConnection);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...UnitName = %s"),lpdmConData-
>szUnitName);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...Common=%s"),lpdmConData-
>szCommon);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...Specific=%s"),lpdmConData-
>szSpecific);
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...VarNum=%d"),lpdmConData-
>dwVarNum);
    ODKTrace(szText);
    return TRUE;
}

void MyDMEnumConnectionData()
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    CMN_ERROR Error;
    TCHAR szText[255];
    DWORD dwItems = 0;
    TCHAR szProjectFile[255];
    DM_CONNKEY ConnKey;
    memset(&ConnKey,0,sizeof(ConnKey));
    memset(&Error,0,sizeof(Error));
    ret = MyDMGetConnectionState();
    if(FALSE != ret)
```

```
    {
        MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
        _tcsncpy_s(szProjectFile, _countof(szProjectFile), /*PROJ_PATH*/g_szProjectFile,
_TRUNCATE);
        ret = DMEnumConnectionData(szProjectFile,
                &ConnKey,
                dwItems,
                MyDMEnumConnectionCallback,
                NULL,
                &Error);
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
DMEnumConnectionData: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("DMEnumConnectionData"));
        }
        ODKTrace(szText);
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMEnumConnectionData (页 2087)

DM_ENUM_CONNECTION_PROC (页 2091)

3.2.11.6 查询项目信息

示例

```
//{{ODK_EXAMPLE}Inquire project informations (MCP)}
//{{FUNCTION}DMGetProjectInformation (MCP)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: DMGetProjectInformation(void) ODK DM CS
// ========================================================================
// Desc. : Inquire project informations
// ========================================================================
void MyDMGetProjectInformation(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    CMN_ERROR Error;
    BOOL ret = FALSE;
    TCHAR szText[255];
    TCHAR szProjectFile[_MAX_PATH +1];
    VOID* pvUser = AfxGetApp();
    DM_PROJECT_INFO Info;
    memset(&Error, 0, sizeof(CMN_ERROR));
    memset(&Info, 0, sizeof(DM_PROJECT_INFO));
    ret = MyDMGetConnectionState(); //check the connection state
    if(FALSE != ret)
    {
        MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
        _tcsncpy_s(szProjectFile, _countof(szProjectFile), /*PROJ_PATH*/g_szProjectFile,
_TRUNCATE);
        ret = DMGetProjectInformation(szProjectFile, &Info, &Error);
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
DMGetProjectInformation: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
            ODKTrace(szText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE,
_T("**DMGetProjectInformation:%d**"), ret);
            ODKTrace(szText);
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("
ProjectFile=%s"),Info.szProjectFile);
            ODKTrace(szText);
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("
DSNName=%s"),Info.szDSNName);
            ODKTrace(szText);
```

```
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("
DataLocale=0x%08X"),Info.dwDataLocale);
            ODKTrace(szText);
            _sntprintf_s(szText, _countof(szText), _TRUNCATE,
_T("******************************"), ret);
            ODKTrace(szText);
        }
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMGetProjectInformation (页 1943)

3.2.11.7 OnTestDeactivateRuntimeProject

示例

```
//{{ODK_EXAMPLE}OnTestDeactivateRuntimeProject (MCP)}
//{{FUNCTION}DMDeactivateRTProject (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestDeactivateRuntimeProject()
{
    CCmnError cmnError;
    if(!::DMDeactivateRTProject(cmnError))
    {
        cmnError.Show(__T("DMDeactivateRTProject failed\n"));
    }
}
//{{ODK_EXAMPLE}(END)}
```

3.2.11.8 OnTestEnumGroupsAll

示例

```
//{{ODK_EXAMPLE}OnTestEnumGroupsAll (MCP)}
//{{FUNCTION}DMEnumVarGrpData (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestEnumGroupsAll()
{
    CCmnError cmnError;
    if(!::DMEnumVarGrpData((LPSTR)(LPCTSTR) m_strProject, NULL, 0, EnumVarGrpProc, this,
cmnError))
    {
      cmnError.Show(__T("DMEnumVarGrpData failed\n"));
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMEnumVarGrpData (页 1969)

3.2.11.9 OnTestEnumVariables

示例

```
//{{ODK_EXAMPLE}OnTestEnumVariables (MCP)}
//{{FUNCTION}DMEnumVariables (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestEnumVariables()
{
    DM_VARFILTER dmVarFilter;
    memset( &dmVarFilter, 0, sizeof(DM_VARFILTER) );
    //-----------------------------------------------
    // Initialisierung des Filters //
    //-----------------------------------------------
    // //
        dmVarFilter.dwFlags = DM_VARFILTER_TYPE;
        dmVarFilter.dwNumTypes = 3;
        DWORD dwFilterTypes[3];
        dwFilterTypes[0] = DM_VARTYPE_BIT;
        dwFilterTypes[1] = DM_VARTYPE_DWORD;
        dwFilterTypes[2] = DM_VARTYPE_DOUBLE;
        dmVarFilter.pdwTypes = dwFilterTypes;
        dmVarFilter.lpszName = __T("VAR_1_BIT");
        dmVarFilter.lpszGroup = __T("VARGROUP_1");
        dmVarFilter.lpszConn = __T("TF_CONN_1");
    // //
    //-----------------------------------------------
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if(!DMEnumVariables((LPSTR)(LPCTSTR) m_strProject, NULL/*&dmVarFilter*/,
EnumVariablesProc, this, &Error))
    {
        Error.Show(__T("DMEnumVariables failed\n"));
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMEnumVariables (页 1974)

3.2.11.10 OnTestEnumConnectionDataAll

示例

```
//{{ODK_EXAMPLE}OnTestEnumConnectionDataAll (MCP)}
//{{FUNCTION}DMEnumConnectionData (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestEnumConnectionDataAll()
{
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if( !::DMEnumConnectionData( m_strProject, NULL, 0, EnumConnectionDataProc, this,
&Error))
    {
    Error.Show(__T("DMEnumConnectionData failed\n"));
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMEnumConnectionData (页 2087)

3.2.11.11 OnTestMachines

示例

```
//{{ODK_EXAMPLE}OnTestMachines (MCP)}
//{{FUNCTION}DMGetMachineTable (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestMachines()
{
    CMN_ERROR Error;
    memset(&Error,0,sizeof(CMN_ERROR));
    DM_MACHINE_TABLE dmMachineTable;
    memset(&dmMachineTable, 0, sizeof(dmMachineTable));
    if(!DMGetMachineTable(m_strProject, &dmMachineTable, &Error))
    {
    // Error.Show(__TEXT("DMGetMachineTable failed\n"));
    // AfxMessageBox( strError );
    }
    else
    {
        for( int i = 0; i < dmmachinetable.nNumMachines; i++ )
        {
            CString strData;
            strData.Format( _T( "Computer : %s, Type : %s, Site : %s" ),
                dmMachineTable.tm[i].szMachineName,
                dmMachineTable.tm[i].fServer ? _T( "Server" ) : _T( "Client or ES" ),
                dmMachineTable.tm[i].fLocal ? _T( "local" ) : _T( "remote" ));
            PutStr( strData);
        }
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMGetMachineTable (页 1934)

3.2.11.12 OnTestProjectInfo

示例

```
//{{ODK_EXAMPLE}OnTestProjectInfo (MCP)}
//{{FUNCTION}DMGetProjectInformation (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestProjectInfo()
{
    CCmnError cmnError;
    DM_PROJECT_INFO ProjectInfo;
    memset(&ProjectInfo, 0, sizeof(DM_PROJECT_INFO));
    if(!DMGetProjectInformation(m_strProject, &ProjectInfo, cmnError))
    {
        cmnError.Show(__TEXT("DMGetProjectInformation failed\n"));
    }
    else
    {
        CString strData;
        strData.Format(_T("Projekt : %s, Data Source : %s, DataLocale : %08X"),
                ProjectInfo.szProjectFile,
                ProjectInfo.szDSNName,
                ProjectInfo.dwDataLocale );
        PutStr(strData);
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMGetProjectInformation (页 1943)

3.2.11.13 OnTestProjectPaths

示例

```
//{{ODK_EXAMPLE}OnTestProjectPaths (MCP)}
//{{FUNCTION}DMGetProjectDirectory (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestProjectPaths()
{
    DM_DIRECTORY_INFO dmDirInfo;
    CCmnError Error;
    memset(&dmDirInfo, 0, sizeof(DM_DIRECTORY_INFO));
    memset(&Error,0,sizeof(CCmnError));
    if(!DMGetProjectDirectory(m_strAppName, m_strProject,
                &dmDirInfo, &Error))
    {
        Error.Show(__TEXT("DMGetProjectDirectory failed\n"));
    }
    else
    {
        CString strData;
        strData.Format(_T("szProjectDir = %s"), dmDirInfo.szProjectDir);
        PutStr(strData);
        strData.Format(_T("szProjectAppDir = %s"), dmDirInfo.szProjectAppDir);
        PutStr(strData);
        strData.Format(_T("szProjectGlobalLibDir = %s"), dmDirInfo.szGlobalLibDir);
        PutStr(strData);
        strData.Format(_T("szProjectLibDir = %s"), dmDirInfo.szProjectLibDir);
        PutStr(strData);
        strData.Format(_T("szLokalProjectAppDir = %s"), dmDirInfo.szLokalProjectAppDir);
        PutStr(strData);
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMGetProjectDirectory (页 1941)

3.2.11.14 OnTestOpenProject

示例

```
//{{ODK_EXAMPLE}OnTestOpenProject (MCP)}
//{{FUNCTION}DMOpenProject (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestOpenProject()
{
    #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    TCHAR szProject[_MAX_PATH + 1];
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    CTestCliView* pView = GetFirstView();
    memset(szProject,0, sizeof(szProject)); // Delete projectname to call dialog
            // Or set fixed projectname
    strcpy( szProject, _T(PROJ_PATH) );
    if(!DMOpenProject(pView->GetSafeHwnd(),
            szProject, NELEM(szProject), &Error))
    {
        Error.Show(__TEXT("DMOpenProject failed.\n"));
    }
    else
    {
        AfxMessageBox(szProject);
        m_strProject = szProject;
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMOpenProjectPlus (页 1947)

3.2.11.15 OnTestOpenProjects

示例

```
//{{ODK_EXAMPLE}OnTestOpenProjects (MCP)}
//{{FUNCTION}DMEnumOpenedProjects (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestOpenProjects()
{
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if(!DMEnumOpenedProjects(NULL, OpenProjectsProc, this, &Error))
    {
        Error.Show(__TEXT("DMEnumOpenedProjects failed\n"));
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMEnumOpenedProjects (页 1938)

3.2.11.16 OnTestRuntimeProject

示例

```
//{{ODK_EXAMPLE}OnTestRuntimeProject (MCP)}
//{{FUNCTION}DMGetRuntimeProject (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestRuntimeProject()
{
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    TCHAR szBuffer[_MAX_PATH + 1];
    if(!DMGetRuntimeProject(szBuffer, NELEM(szBuffer), &Error))
    {
        Error.Show(__TEXT("DMGetRuntimeProject failed\n"));
    }
    else
    {
        PutStr(szBuffer);
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMGetRuntimeProject (页 1945)

3.2.11.17 OnTestSystemLocale

示例

```
//{{ODK_EXAMPLE}OnTestSystemLocale (MCP)}
//{{FUNCTION}DMGetSystemLocale (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestSystemLocale()
{
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    DWORD dwLocaleID = 0;
    if(!DMGetSystemLocale(&dwLocaleID, &Error))
    {
        Error.Show(__TEXT("DMGetSystemLocale failed\n"));
    }
    else
    {
        CString strData;
        strData.Format(_T("Systemlocale : %08X"), dwLocaleID);
        PutStr(strData);
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMGetSystemLocale (页 2096)

3.2.11.18 OnTestUpdateCycles

示例

```
//{{ODK_EXAMPLE}OnTestUpdateCycles (MCP)}
//{{FUNCTION}DMEnumUpdateCycles (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestUpdateCycles()
{
    DWORD dwNumCalls = 0;
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if( !DMEnumUpdateCycles(m_strProject, &dwNumCalls,
                EnumCyclesProc, this, &Error))
    {
            Error.Show(__TEXT("DMEnumUpdateCycles failed\n"));
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMEnumUpdateCycles (页 1921)

3.2.11.19 OnTestVariablenBeginstartvarupdate

示例

```
//{{ODK_EXAMPLE}OnTestVariablenBeginstartvarupdate (MCP)}
//{{FUNCTION}DMBeginStartVarUpdate (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenBeginstartvarupdate()
{
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if(!DMBeginStartVarUpdate(&m_dwTAID, &Error))
    {
    Error.Show(_T("DMBeginStartVarUpdate failed\n"));
    }
    else
    {
    CString strData;
    strData.Format(_T("DMBeginStartVarUpdate: TAID:%lu."), m_dwTAID);
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMBeginStartVarUpdate (页 2100)

3.2.11.20 OnTestVariablenEndstartvarupdate

示例

```
//{{ODK_EXAMPLE}OnTestVariablenEndstartvarupdate (MCP)}
//{{FUNCTION}DMEndStartVarUpdate (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenEndstartvarupdate()
{
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if( !DMEndStartVarUpdate(m_dwTAID, &Error))
    {
        Error.Show(_T("DMEndStartVarUpdate failed\n"));
    }
    else
    {
        CString strData;
        strData.Format(_T("DMEndStartVarUpdate: TAID:%lu."), m_dwTAID);
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMEndStartVarUpdate (页 2101)

3.2.11.21 OnTestVariablenGetvalue

示例

```
//{{ODK_EXAMPLE}OnTestVariablenGetvalue (MCP)}
//{{FUNCTION}DMGetValue (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenGetvalue()
{
    int nNum = GetVarCount();
    LPDM_VARKEY lpdmVarKey = GetVarKeys();

    LPDM_VAR_UPDATE_STRUCT lpdmvus = new DM_VAR_UPDATE_STRUCT[nNum];
    memset(lpdmvus, 0, sizeof(DM_VAR_UPDATE_STRUCT) * nNum);
    // the included VARIANT's in the DM_VAR_UPDATE_STRUCT's then initialized to VT_EMPTY
with the memset,
    // don't do this later again, because VT_BSTR's can be present after DMGetValue(...)!

    CCmnError Error;
    memset(&Error, 0, sizeof(CCmnError));

    if( !DMGetValue(lpdmVarKey, nNum, lpdmvus, &Error))
    {
        Error.Show(_T("DMGetValue failed\n"));
    }
    else
    {
        CString strData;
        for( int i=0; i < nmum; i++)
        {
            LPDM_VAR_UPDATE_STRUCT lpdmvus2 = &lpdmvus[i];
            switch( lpdmvus2->dmTypeRef.dwType )
            {
            case DM_VARTYPE_BIT:
                if (lpdmvus2->dmValue.boolVal)
                {
                strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = TRUE"),
                lpdmvus2->dmVarKey.szName,
                lpdmvus2->dmVarKey.dwID);
                }
                else
                {
                strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = FALSE"),
                lpdmvus2->dmVarKey.szName,
                lpdmvus2->dmVarKey.dwID);
                }
                break;
            case DM_VARTYPE_BYTE:
                strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %u"),
```

```
        lpdmvus2->dmVarKey.szName,
        lpdmvus2->dmVarKey.dwID,
        lpdmvus2->dmValue.bVal);
        break;
    case DM_VARTYPE_SBYTE:
        strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %d"),
        lpdmvus2->dmVarKey.szName,
        lpdmvus2->dmVarKey.dwID,
        lpdmvus2->dmValue.bVal);
        break;
    case DM_VARTYPE_WORD:
        strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %u"),
        lpdmvus2->dmVarKey.szName,
        lpdmvus2->dmVarKey.dwID,
        lpdmvus2->dmValue.iVal);
        break;
    case DM_VARTYPE_SWORD:
        strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %d"),
        lpdmvus2->dmVarKey.szName,
        lpdmvus2->dmVarKey.dwID,
        lpdmvus2->dmValue.iVal);
        break;
    case DM_VARTYPE_DWORD:
        strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %u"),
        lpdmvus2->dmVarKey.szName,
        lpdmvus2->dmVarKey.dwID,
        lpdmvus2->dmValue.lVal);
        break;
    case DM_VARTYPE_SDWORD:
        strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %d"),
        lpdmvus2->dmVarKey.szName,
        lpdmvus2->dmVarKey.dwID,
        lpdmvus2->dmValue.lVal);
        break;
    case DM_VARTYPE_FLOAT:
        strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %f"),
        lpdmvus2->dmVarKey.szName,
        lpdmvus2->dmVarKey.dwID,
        lpdmvus2->dmValue.fltVal);
        break;
    case DM_VARTYPE_DOUBLE:
        strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %f"),
        lpdmvus2->dmVarKey.szName,
        lpdmvus2->dmVarKey.dwID,
        lpdmvus2->dmValue.dblVal);
        break;
    case DM_VARTYPE_TEXT_8:
        strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %s"),
        lpdmvus2->dmVarKey.szName,
        lpdmvus2->dmVarKey.dwID,
        lpdmvus2->dmValue.bstrVal);
```

```
                break;
    /*
            case DM_VARTYPE_TEXT_16:
                strData.Format(_T("GetValue: Variable: %s\t( ID = %d ):\tWert = %s"),
                lpdmvus2->dmVarKey.szName,
                lpdmvus2->dmVarKey.dwID,
                lpdmvus2->dmValue.bstrVal);
                break;
    */
            default:
                strData.Format(_T("Unbekannter Variablentyp !"));
            }
            PutStr(strData);

           // clear the given VARIANT's in every array element, because a VT_BSTR's can be
present
          // do not only delete the array later without clearing the Variants here because
of memory leak's
            VariantClear(&(lpdmvus2->dmValue));
        }
    }
    delete []lpdmVarKey;
    delete []lpdmvus;
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMGetValue (页 1977)

3.2.11.22 OnTestVariablenGetvaluewait

示例

```
//{{ODK_EXAMPLE}OnTestVariablenGetvaluewait (MCP)}
//{{FUNCTION}DMGetValueWait (MCP)}
//{{FUNCTION}(END)}
//-----< Notification-Callback for GetValueWait >----------------------------
BOOL GetValueWaitNotify(DWORD dwTAID,
                LPDM_VAR_UPDATE_STRUCT lpdmvus,
                DWORD dwItems,
                LPVOID lpvUser)
{
    dwTAID;
    CTestCliDoc* pDoc = (CTestCliDoc*)lpvUser;
    for(DWORD i=0; i <dwitems; i++)
    {
        LPDM_VAR_UPDATE_STRUCT lpdmvus2 = &lpdmvus[i];
        CString strData;
        switch(lpdmvus2->dmTypeRef.dwType)
        {
        case DM_VARTYPE_BIT:
            if (lpdmvus2->dmValue.boolVal)
            {
                strData.Format(_T("GetValueWait: Tag: %s\t( ID = %d ):\tValue = TRUE,
State=%04x"),
                lpdmvus2->dmVarKey.szName,
                lpdmvus2->dmVarKey.dwID,
                lpdmvus2->dwState);
            }
            else
            {
                strData.Format(_T("GetValueWait: Tag: %s\t( ID = %d ):\tValue = FALSE,
State=%04x"),
                lpdmvus2->dmVarKey.szName,
                lpdmvus2->dmVarKey.dwID,
                lpdmvus2->dwState);
            }
            break;
        case DM_VARTYPE_BYTE:
           strData.Format(_T("GetValueWait: Tag: %s\t( ID = %d ):\tValue = %u, State=%04x"),
                lpdmvus2->dmVarKey.szName,
                lpdmvus2->dmVarKey.dwID,
                lpdmvus2->dmValue.bVal,
                lpdmvus2->dwState);
            break;
        case DM_VARTYPE_SBYTE:
           strData.Format(_T("GetValueWait: Tag: %s\t( ID = %d ):\tValue = %d, State=%04x"),
                lpdmvus2->dmVarKey.szName,
```

```
            lpdmvus2->dmVarKey.dwID,
            lpdmvus2->dmValue.bVal,
            lpdmvus2->dwState);
        break;
    case DM_VARTYPE_WORD:
      strData.Format(_T("GetValueWait: Tag: %s\t( ID = %d ):\tValue = %u, Status=%04x"),
            lpdmvus2->dmVarKey.szName,
            lpdmvus2->dmVarKey.dwID,
            lpdmvus2->dmValue.iVal,
            lpdmvus2->dwState);
        break;
    case DM_VARTYPE_SWORD:
       strData.Format(_T("GetValueWait: Tag: %s\t( ID = %d ):\tValue = %d, State=%04x"),
            lpdmvus2->dmVarKey.szName,
            lpdmvus2->dmVarKey.dwID,
            lpdmvus2->dmValue.iVal,
            lpdmvus2->dwState);
        break;
    case DM_VARTYPE_DWORD:
       strData.Format(_T("GetValueWait: Tag: %s\t( ID = %d ):\tValue = %u, State=%04x"),
            lpdmvus2->dmVarKey.szName,
            lpdmvus2->dmVarKey.dwID,
            lpdmvus2->dmValue.lVal,
            lpdmvus2->dwState);
        break;
    case DM_VARTYPE_SDWORD:
       strData.Format(_T("GetValueWait: Tag: %s\t( ID = %d ):\tValue = %d, State=%04x"),
            lpdmvus2->dmVarKey.szName,
            lpdmvus2->dmVarKey.dwID,
            lpdmvus2->dmValue.lVal,
            lpdmvus2->dwState);
        break;
    case DM_VARTYPE_FLOAT:
       strData.Format(_T("GetValueWait: Tag: %s\t( ID = %d ):\tValue = %f, State=%04x"),
            lpdmvus2->dmVarKey.szName,
            lpdmvus2->dmVarKey.dwID,
            lpdmvus2->dmValue.fltVal,
            lpdmvus2->dwState);
        break;
    case DM_VARTYPE_DOUBLE:
       strData.Format(_T("GetValueWait: Tag: %s\t( ID = %d ):\tValue = %f, State=%04x"),
            lpdmvus2->dmVarKey.szName,
            lpdmvus2->dmVarKey.dwID,
            lpdmvus2->dmValue.dblVal,
            lpdmvus2->dwState);
        break;
    case DM_VARTYPE_TEXT_8:
       strData.Format(_T("GetValueWait: Tag: %s\t( ID = %d ):\tValue = %s, State=%04x"),
            lpdmvus2->dmVarKey.szName,
            lpdmvus2->dmVarKey.dwID,
            lpdmvus2->dmValue.bstrVal,
```

```
                lpdmvus2->dwState);
            break;
        case DM_VARTYPE_RAW:
            {
                LPBYTE pArray;
                HRESULT hr;
                hr = SafeArrayAccessData(lpdmvus2->dmValue.parray, (VOID **) &pArray);
                if (! FAILED(hr))
                {
                strData.Format(_T("Var:%s: Type:raw data\nValues:%02x %02x %02x %02x\n"),
                lpdmvus2->dmVarKey.szName,
                pArray[0],
                pArray[1],
                pArray[2],
                pArray[3]);
                SafeArrayUnaccessData(lpdmvus2->dmValue.parray);
                }
                else
                {
                strData.Format(_T("SafeArrayAccessData failed\n"));
                }
                break;
            }
        default:
            strData.Format(_T("GetValueWait: Unknonw data type !"));
        }
        pDoc->PutStr(strData);
    }
    return(TRUE);
}

//-----< OnTestVariablenGetvaluewait >-------------------------------------------
void CTestCliDoc::OnTestVariablenGetvaluewait()
{
    int nNum = GetVarCount();
    LPDM_VARKEY lpdmVarKey = GetVarKeys();
    LPDM_VAR_UPDATE_STRUCT lpdmvus = new DM_VAR_UPDATE_STRUCT[nNum];
    memset(lpdmvus, 0, sizeof(DM_VAR_UPDATE_STRUCT) * nNum);
    DWORD dwTAID = 5;
    DWORD dwTimeOut = 5000;
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if(!DMGetValueWait(&dwTAID,
                lpdmVarKey,
                nNum,
                FALSE,
                dwTimeOut,
                GetValueWaitNotify,
                this,
                &Error))
    {
```

```
        Error.Show(_T("DMGetValueWait (FALSE) failed\n"));
    }
    delete []lpdmVarKey;
    delete []lpdmvus;
}

//-----< OnTestVariablenGetvaluewait >-------------------------------------------
void CTestCliDoc::OnTestVariablenGetvaluewaitTrue()
{
    int nNum = GetVarCount();
    LPDM_VARKEY lpdmVarKey = GetVarKeys();
    LPDM_VAR_UPDATE_STRUCT lpdmvus = new DM_VAR_UPDATE_STRUCT[nNum];
    memset(lpdmvus, 0, sizeof(DM_VAR_UPDATE_STRUCT) * nNum);
    DWORD dwTAID = 5;
    DWORD dwTimeOut = 5000;
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if( !DMGetValueWait(&dwTAID,
                lpdmVarKey,
                nNum,
                TRUE,
                dwTimeOut,
                GetValueWaitNotify,
                this,
                &Error))
    {
                Error.Show(_T("DMGetValueWait (TRUE) failed\n"));
    }
    delete []lpdmVarKey;
    delete []lpdmvus;
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMGetValueWait (页 1991)

3.2.11.23 OnTestVariablenGetVarInfo

示例

```
//{{ODK_EXAMPLE}OnTestVariablenGetVarInfo (MCP)}
//{{FUNCTION}DMGetVarInfo (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenGetVarInfo()
{
    int nNum = GetVarCount();
    LPDM_VARKEY lpdmVarKey = GetVarKeys();
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if(!::DMGetVarInfo(m_strProject, lpdmVarKey, nNum, &Error))
    {
        Error.Show(_T("DMGetVarInfo failed\n"));
    }
    else
    {
        CString strData;
        for(int i=0; i<nnum; i++)
        {
            strData.Format(_T("DMGetVarInfo: Variablenname: %s, VarID:%lu."),
                lpdmVarKey[i].szName,
                lpdmVarKey[i].dwID);
            PutStr(strData);
        }
    }
    delete []lpdmVarKey;
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMGetVarInfo (页 1998)

3.2.11.24 OnTestVariablenGetvarlimits

示例

```
//{{ODK_EXAMPLE}OnTestVariablenGetvarlimits (MCP)}
//{{FUNCTION}DMGetVarLimits (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenGetvarlimits()
{
    int nNum = GetVarCount();
    LPDM_VARLIMIT lpdmVarLimit = new DM_VARLIMIT[nNum];
    memset(lpdmVarLimit, 0, sizeof(DM_VARLIMIT) * nNum);
    LPDM_VARKEY lpdmVarKey = GetVarKeys();
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if(!::DMGetVarLimits(m_strProject, lpdmVarKey, nNum,
                lpdmVarLimit, &Error))
    {
        Error.Show(__T("DMGetVarLimits failed\n"));
    }
    else
    {
        CString strData;
        for(int i=0; i<nnum; i++)
        {
            strData.Format(_T("DMGetVarLimits: Var:%s, MaxRange:%lf, MinRange:%lf,
MinLimit:%lf, MaxLimit:%lf."),
            lpdmVarKey[i].szName,
            lpdmVarLimit[i].dmMaxRange.dblVal,
            lpdmVarLimit[i].dmMinRange.dblVal,
            lpdmVarLimit[i].dmMaxLimit.dblVal,
            lpdmVarLimit[i].dmMinLimit.dblVal);
            PutStr(strData);
        }
    }
    delete []lpdmVarLimit;
    delete []lpdmVarKey;
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMGetVarLimits (页 2010)

3.2.11.25 OnTestVariablenGetvartype

示例

```
//{{ODK_EXAMPLE}OnTestVariablenGetvartype (MCP)}
//{{FUNCTION}DMGetVarType (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenGetvartype()
{
    int nNum = GetVarCount();
    LPDM_TYPEREF lpdmTypeRef = new DM_TYPEREF[nNum];
    memset(lpdmTypeRef, 0, sizeof(DM_TYPEREF) * nNum);
    LPDM_VARKEY lpdmVarKey = GetVarKeys();
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if(!::DMGetVarType(m_strProject, lpdmVarKey, nNum,
                lpdmTypeRef, &Error))
    {
        Error.Show(_T("DMGetVarType failed\n"));
    }
    else
    {
        CString strData;
        for(int i=0; i<nnum; i++)
        {
            strData.Format(_T("DMGetVarType: Var:%s, Size:%lu, Name:%s."),
                lpdmVarKey[i].szName,
                lpdmVarKey[i].dwID);
            PutStr(strData);
        }
    }
    delete []lpdmVarKey;
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMGetVarType (页 2017)

3.2.11.26 OnTestVariablenResumevarupdate

示例

```
//{{ODK_EXAMPLE}OnTestVariblenResumevarupdate (MCP)}
//{{FUNCTION}DMResumeVarUpdate (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenResumevarupdate()
{
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if(!DMResumeVarUpdate(m_dwTAID, &Error))
    {
        Error.Show(_T("DMResumeVarUpdate failed.\n"));
    }
    else
    {
        CString strData;
        strData.Format(_T("DMResumeVarUpdate ok."));
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMResumeVarUpdate (页 2103)

3.2.11.27 OnTestVariablenSetvalue

示例

```
//{{ODK_EXAMPLE}OnTestVariablenSetvalue (MCP)}
//{{FUNCTION}DMSetValue (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenSetvalue()
{
    int nNum = GetVarCount();
    LPDM_VARKEY lpdmVarKey = GetVarKeys();
    LPDWORD lpdmVarState = new DWORD[nNum];
    INT i;
    CString strData;
    CCmnError cmnError;
    for (i = 0; i < 1; i++)</p>
    {
        DWORD dwStart = GetTickCount();
        DWORD dwEnd;
        if (! DMSetValue(lpdmVarKey, nNum, m_varValues, lpdmVarState, cmnError))
        {
            break;
        }
        dwEnd = GetTickCount();
        strData.Format("DMSetValue OK, Dauer %d ms", dwEnd - dwStart);
        PutStr(strData);
    }
    delete []lpdmVarKey;
    delete []lpdmVarState;
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMSetValue (页 2024)

3.2.11.28 OnTestVariablenSetvaluewait

示例

```
//{{ODK_EXAMPLE}OnTestVariablenSetvaluewait (MCP)}
//{{FUNCTION}DMSetValueWait (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenSetvaluewait()
{
    int nNum = GetVarCount();
    LPDM_VARKEY lpdmVarKey = GetVarKeys();
    LPDWORD lpdmVarState = new DWORD[nNum];
    DWORD dwTimeOut = 1000L;
    CCmnError cmnError;
    if(!::DMSetValueWait(&m_dwTAID, lpdmVarKey,nNum, m_varValues,dwTimeOut,
                CompletionProc, this, cmnError))
    {
        cmnError.Show(_T("DMSetValueWait failed\n"));
    }
    else
    {
        CString strData;
        strData.Format(_T("DMSetValueWait ok."));
    }
    delete []lpdmVarKey;
    delete []lpdmVarState;
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMSetValueWaitMessage (页 2041)

DMSetValueWait (页 2035)

3.2.11.29 OnTestVariablenStopallupdates

示例

```
//{{ODK_EXAMPLE}OnTestVariablenStopallupdates (MCP)}
//{{FUNCTION}DMStopAllUpdates (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenStopallupdates()
{
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if(!DMStopAllUpdates(&Error))
    {
        Error.Show(_T("DMStopAllUpdates failed\n"));
    }
    else
    {
        CString strData;
        strData.Format(_T("DMStopAllUpdates ok."));
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMStopAllUpdates (页 2121)

3.2.11.30 OnTestVariablenStopvarupdate

示例

```
//{{ODK_EXAMPLE}OnTestVariablenStopvarupdate (MCP)}
//{{FUNCTION}DMStopVarUpdate (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenStopvarupdate()
{
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if( !DMStopVarUpdate(m_dwTAID, &Error))
    {
        Error.Show(_T("DMStopVarUpdate failed\n"));
    }
    else
    {
        CString strData;
        strData.Format(_T("DMStopVarUpdate: TAID:%lu."), m_dwTAID);
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMStopVarUpdate (页 2122)

3.2.11.31 OnTestVariablenSuspendvarupdate

示例

```
//{{ODK_EXAMPLE}OnTestVariablenSuspendvarupdate (MCP)}
//{{FUNCTION}DMSuspendVarUpdate (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestVariablenSuspendvarupdate()
{
    CCmnError Error;
    memset(&Error,0,sizeof(CCmnError));
    if(!DMSuspendVarUpdate(m_dwTAID, &Error))
    {
        Error.Show(_T("DMSuspendVarUpdate failed\n"));
    }
    else
    {
        CString strData;
        strData.Format(_T("DMSuspendVarUpdate ok."));
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMSuspendVarUpdate (页 2123)

3.2.11.32 OnTestWinCCShutdown

示例

```
//{{ODK_EXAMPLE}OnTestWinCCShutdown (MCP)}
//{{FUNCTION}DMExitWinCC (MCP)}
//{{FUNCTION}(END)}
void CTestCliDoc::OnTestWinCCShutdown()
{
    DMExitWinCC();
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMExitWinCC (页 1924)

3.2.11.33 以对话框方式打开项目

示例

```
//{{ODK_EXAMPLE}Open projectby means of dialog (MCP)}
//{{FUNCTION}DMOpenProject (MCP)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: MyDMOpenProject(void) ODK DM CS
// ========================================================================
// Desc. : open project, give name through WinCC-Dialog-Box
// ========================================================================
void MyDMOpenProject(void)
{
    CMN_ERROR Error;
    BOOL ret = FALSE;
    DWORD dwSize = 255;
    HWND handle = NULL;
    TCHAR szProjFile[255];
    TCHAR szText[255];
    _tcsncpy_s(szProjFile, _countof(szProjFile), _T("C:\\Siemens\\ODK\\Samples\\Projects\
\Demo\\odk.mcp"), _TRUNCATE);
    memset(&Error, 0,sizeof( CMN_ERROR ));
    ret = MyDMConnect(); // check connection to DataManager
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in MyDMConnect"));
        ODKTrace(szText);
    }
    else
    {
        memset(&Error, 0,sizeof( CMN_ERROR ));
        ret = DMOpenProject(handle, szProjFile, dwSize, &Error);
        if(ret == FALSE)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in DMOpenProject:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText,_countof(szText), _TRUNCATE, _T("DMOpenProject (%s)"),
szProjFile);
        }
        ODKTrace(szText);
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMOpenProjectPlus (页 1947)

3.2.11.34 读取变量

示例

```
// ========================================================================
// ========================================================================
// short : Modul with examples to DataManager-API
// RUNTIME
// SC: PO = Projekt opened PG= Projekt closed
// ************************************************************************
#include "stdafx.h" // if MFC classes
//#include "odkapi.h" // if console application
#include <time.h>
#include "dm02.h"

// ========================================================================
// 2.1 Interface IMPORT
// ========================================================================
//extern void ODKTrace(LPCTSTR);
extern BOOL MyDMConnect (void); // Connect
extern BOOL MyDMGetConnectionState(void);

// ========================================================================
// 2.2 Interface EXPORT
// ========================================================================
// ========================================================================
// 2.3 Interface LOCAL
// ========================================================================
// ========================================================================
// 3. Definitions
// ========================================================================

//{{ODK_EXAMPLE}Read tag (MCP)}
//{{FUNCTION}DMGetRuntimeProject (MCP)}
//{{FUNCTION}DMGetValue (MCP)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: MyDMGetValue(void) ODK DM CS
// ========================================================================
// short : Read tag
// ========================================================================
void MyDMGetValue(void)
{
    CMN_ERROR Error;
    BOOL ret = FALSE;
    DWORD dwSize = _MAX_PATH;
    TCHAR szText[255];
    TCHAR szProjFile[_MAX_PATH +1];
    TCHAR BstrValue[255]; //
```

```
    const short int nNum = 11; // tagcount
    const short int nTextMax = 20; // textlenght
    TCHAR TagNames[nNum][nTextMax]=
    {
        "VAR_1_BOOL", // boolVal; // VT_BOOL.
        "VAR_2_BYTE", // bVal; // VT_UI1
        "VAR_3_SBYTE", // iVal; // VT_I2
        "VAR_4_WORD", // lVal // VT_I4
        "VAR_5_SWORD", // iVal // VT_I2
        "VAR_6_DWORD", // dblVal // VT_R8
        "VAR_7_SDWORD", // lVal // VT_I4
        "VAR_8_FLOAT", // fltVal // VT_R4
        "VAR_9_DOUBLE", // dblVal // VT_R8
        "VAR_A_TEXT8", // bstrVal // VT_BSTR
        "VAR_B_TEXT16" // bstrVal // VT_BSTR
    };
    DM_VARKEY VarKey[nNum];

    DM_VAR_UPDATE_STRUCT VarUp[nNum];
    memset(&VarUp, 0, sizeof(DM_VAR_UPDATE_STRUCT) * nNum);
    // the included VARIANT's in the DM_VAR_UPDATE_STRUCT's then initialized to VT_EMPTY
with the memset,
    // don't do this later again, because VT_BSTR's can be present after DMGetValue(...)!

    memset(&VarKey,0, sizeof(DM_VARKEY) * nNum);
    memset(&Error,0,sizeof(Error));
    ret = MyDMGetConnectionState();
    if(FALSE != ret)
    {
        memset(&Error,0,sizeof(Error));
        // RunTime project
        ret = DMGetRuntimeProject(szProjFile, dwSize, &Error);
        if(FALSE != ret)
        {
            // fill out DM_VARKEY
            for(int iRead = 0; iRead <nNum; iRead++)
            {
                VarKey[iRead].dwKeyType = DM_VARKEY_NAME;
                VarKey[iRead].dwID = 0;
                strcpy( VarKey[iRead].szName, &TagNames[iRead][0]);
                VarKey[iRead].lpvUserData = (VOID *) iRead;
            }
            memset(&Error,0,sizeof(Error));
            ret = DMGetValue(VarKey, nNum, VarUp, &Error);
            if(FALSE != ret)
            {
                for(int iOut=0; iOut <nNum; iOut++)
                {
                    switch(VarUp[iOut].dmTypeRef.dwType)
                    {
                        case DM_VARTYPE_BIT: //vt = 3 VT_I4 = 3
```

```
                        {
                            sprintf(szText, "Index=%d Name=%s ID=%d Value=%d",
                                    iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,
                                    VarUp[iOut].dmValue.boolVal);
                            break;
                        }
                        case DM_VARTYPE_BYTE: //vt = 17 VT_UI1 = 17
                        {
                            sprintf(szText, "Index=%d Name=%s ID=%d Value=%d",
                                    iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,
                                    VarUp[iOut].dmValue.bVal);
                            break;
                        }
                        case DM_VARTYPE_SBYTE: //vt = 2 VT_I2 = 2
                        {
                            sprintf(szText, "Index=%d Name=%s ID=%d Value=%d",
                                    iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,
                                    VarUp[iOut].dmValue.iVal);
                            break;
                        }
                        case DM_VARTYPE_WORD: //vt = 3 VT_I4 = 3
                        {
                            sprintf(szText, "Index=%d Name=%s ID=%d Value=%d",
                                    iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,
                                    VarUp[iOut].dmValue.lVal);
                            break;
                        }
                        case DM_VARTYPE_SWORD: //vt = 2 VT_I2 = 2
                        {
                            sprintf(szText, "Index=%d Name=%s ID=%d Value=%d",
                                    iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,
                                    VarUp[iOut].dmValue.iVal);
                            break;
                        }
                        case DM_VARTYPE_DWORD: //vt = 5 VT_R8 = 5
                        {
                            sprintf(szText, "Index=%d Name=%s ID=%d Value=%d",
                                    iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,
                                    VarUp[iOut].dmValue.dblVal);
                            break;
                        }
                        case DM_VARTYPE_SDWORD: //vt = 3 VT_I4 = 3
                        {
                            sprintf(szText, "Index=%d Name=%s ID=%d Value=%d",
```

```
                                    iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,
                                    VarUp[iOut].dmValue.lVal);
                            break;
                        }
                        case DM_VARTYPE_FLOAT: //vt = 4 VT_R4 = 4
                        {
                            sprintf(szText, "Index=%d Name=%s ID=%d Value=%f",
                                    iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,
                                    VarUp[iOut].dmValue.fltVal);
                            break;
                        }
                        case DM_VARTYPE_DOUBLE: //vt = 5 VT_R8 = 5
                        {
                            sprintf(szText, "Index=%d Name=%s ID=%d Value=%f",
                                    iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,
                                    VarUp[iOut].dmValue.dblVal);
                            break;
                        }
                        case DM_VARTYPE_TEXT_8: //vt = 8 VT_BSTR = 8
                        {
                            ret = WideCharToMultiByte( CP_ACP,
(DWORD)0,VarUp[iOut].dmValue.bstrVal,
                                                       -1,(LPSTR)&BstrValue[0],128,NULL,NULL);
                            sprintf(szText, "Index=%d Name=%s ID=%d StrValue=%s",
                                    iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,BstrValue);
                            break;
                        }
                        case DM_VARTYPE_TEXT_16: // vt = 8 VT_BSTR = 8
                        {
                            ret = WideCharToMultiByte( CP_ACP,
(DWORD)0,VarUp[iOut].dmValue.bstrVal,
                                                       -1,(LPSTR)&BstrValue[0],128,NULL,NULL);
                            sprintf(szText, "Index=%d Name=%s ID=%d StrValue=%s",
                                    iOut, VarUp[iOut].dmVarKey.szName,
VarUp[iOut].dmVarKey.dwID,BstrValue);
                            break;
                        }
                        default:
                            break;
                    }// end switch case
                    ODKTrace(szText);
                    //printf("%s\r\n",szText);

                    VariantClear( &VarUp[iOut].dmValue );
                }//end for
            }
            else
```

```
            {
                sprintf(szText, "Error in DMGetValue: E1= 0x%08lx ; E2= 0x%08lx ; %s",
                Error.dwError1, Error.dwError2, Error.szErrorText);
                ODKTrace(szText);
                //printf("%s\r\n",szText);
            }
        }
        else
        {
            sprintf(szText, "Error in DMGetRuntimeProject: E1= 0x%08lx ; E2= 0x%08lx ; %s",
                Error.dwError1, Error.dwError2, Error.szErrorText);
            ODKTrace(szText);
            //printf("%s\r\n",szText);
        }
    }
    else
    {
        sprintf(szText, "Error in MyDMGetConnectionState: E1= 0x%08lx ; E2= 0x%08lx ; %s",
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        //printf("%s\r\n",szText);
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMGetRuntimeProject (页 1945)

DMGetValue (页 1977)

3.2.11.35 写入变量

示例

```
//{{ODK_EXAMPLE}Write tag (MCP)}
//{{FUNCTION}DMGetRuntimeProject (MCP)}
//{{FUNCTION}DMSetValue (MCP)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: MyDMSetValue(void) ODK DM CS
// ========================================================================
// short : Write tag
// ========================================================================
void MyDMSetValue(void)
{
    CMN_ERROR Error;
    BOOL ret = FALSE;
    DWORD dwSize = _MAX_PATH;
    TCHAR szText[255];
    TCHAR szProjFile[_MAX_PATH +1];
    TCHAR szTextValue[255]; // for TEXT8 Values
    const short int nNum = 11; // tagcount
    const short int nTextMax = 20; // textlenght
    TCHAR TagNames[nNum][nTextMax]=
    {
        "VAR_1_BOOL", // boolVal; // VT_BOOL.
        "VAR_2_BYTE", // bVal; // VT_UI1
        "VAR_3_SBYTE", // iVal; // VT_I2
        "VAR_4_WORD", // lVal // VT_I4
        "VAR_5_SWORD", // iVal // VT_I2
        "VAR_6_DWORD", // dblVal // VT_R8
        "VAR_7_SDWORD", // lVal // VT_I4
        "VAR_8_FLOAT", // fltVal // VT_R4
        "VAR_9_DOUBLE", // dblVal // VT_R8
        "VAR_A_TEXT8", // bstrVal // VT_BSTR
        "VAR_B_TEXT16" // bstrVal // VT_BSTR
        };
    DM_VARKEY VarKey[nNum];
    DM_VAR_UPDATE_STRUCT VarUp[nNum];
    VARIANT VarVal[nNum];
    DWORD VarSta[nNum];
    static short int Tmp[20];
    BSTR bstrText1, bstrText2;
    memset(&Error, 0, sizeof(CMN_ERROR));
    memset(&VarUp, 0, sizeof(DM_VAR_UPDATE_STRUCT) * nNum);
    memset(&VarKey,0, sizeof(DM_VARKEY) * nNum);
    ret = MyDMGetConnectionState();
    if(FALSE != ret)
    {
```

```
        // RunTime project
        ret = DMGetRuntimeProject(szProjFile, dwSize, &Error);
        if(FALSE != ret)
        {
            // read tag
            for(int i = 0; i < nNum; i++)
            {
                VarKey[i].dwKeyType = DM_VARKEY_NAME;
                VarKey[i].dwID = 0;
                _tcsncpy_s(VarKey[i].szName, _countof(VarKey[i].szName), &TagNames[i][0],
_TRUNCATE);
                VarKey[i].lpvUserData = (VOID *) i;
            }
            // init tag
            //"VAR_1_BOOL",
            VarVal[0].vt = VT_BOOL;
            VarVal[0].boolVal = TRUE;
            //"VAR_2_BYTE"
            VarVal[1].vt = VT_UI1;
            VarVal[1].bVal = 23;
            //"VAR_3_SBYTE",
            VarVal[2].vt = VT_I2;
            VarVal[2].iVal = -23;
            //"VAR_4_WORD",
            VarVal[3].vt = VT_I4;
            VarVal[3].lVal = 89;
            //"VAR_5_SWORD",
            VarVal[4].vt = VT_I2;
            VarVal[4].iVal = -89;
            //"VAR_6_DWORD",
            VarVal[5].vt = VT_R8;
            VarVal[5].dblVal = 1200;
            //"VAR_7_SDWORD",
            VarVal[6].vt = VT_I4;
            VarVal[6].lVal = -1200;
            //"VAR_8_FLOAT",
            VarVal[7].vt = VT_R4;
            VarVal[7].fltVal = (float)3.789;
            //"VAR_9_DOUBLE",
            VarVal[8].vt = VT_R8;
            VarVal[8].dblVal = (double)34.789;
            //"VAR_A_TEXT8",
            _tcsncpy_s(szTextValue, _countof(szTextValue), _T("VAR_A_TEXT8 Value"),
_TRUNCATE);
            ret = MultiByteToWideChar(CP_ACP,(DWORD)0, (LPCSTR) &szTextValue,-1,
(LPWSTR)&Tmp[0],30);
            // SysFreeString(pBSTR);
            bstrText1 = SysAllocString((OLECHAR FAR*)&Tmp[0]);
            VarVal[9].vt = VT_BSTR;
            VarVal[9].bstrVal = bstrText1;
            //"VAR_B_TEXT16"
```

```
            _tcsncpy_s(szTextValue, _countof(szTextValue), _T("VAR_B_TEXT16 Value"),
_TRUNCATE);
            ret = MultiByteToWideChar(CP_ACP,(DWORD)0, (LPCSTR)&szTextValue,-1,
(LPWSTR)&Tmp[0],30);
            // SysFreeString(pB2);
            bstrText2 = SysAllocString((OLECHAR FAR*)&Tmp[0]);
            VarVal[10].vt = VT_BSTR;
            VarVal[10].bstrVal = bstrText2;
            // set tag
            memset(&Error,0,sizeof(CMN_ERROR));
            ret = DMSetValue(&VarKey[0], nNum, &VarVal[0], &VarSta[0], &Error);
            if (FALSE == ret)
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in DMSetValue:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
                ODKTrace(szText);
            }
            else
            {
              _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("write Value to WinCC"));
                ODKTrace(szText);
            }
            SysFreeString(bstrText1);
            SysFreeString(bstrText2);
        }
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

DMGetRuntimeProject (页 1945)

DMSetValue (页 2024)

3.3 图形系统的函数

3.3.1 基本知识

3.3.1.1 函数概览

概述

PDLRTShowApp (页 2230)	显示图形系统
PDLRT_CALLBACK (页 2231)	运行时函数的回调函数
PDLRTClosePicture (页 2217)	关闭画面
PDLRTDisableClosePicture (页 2219)	禁用关闭画面函数
PDLRTEnableClosePicture (页 2220)	启用关闭画面函数
PDLRTGetCursorKeys (页 2233)	查询光标控制键
PDLRTGetDefPropEx (页 2242)	查询对象属性的默认值
PDLRTGetPropEx (页 2245)	查询对象属性的默认值
PDLRTGetFocus (页 2236)	查询输入焦点
PDLRTGetLink (页 2253)	查询对象属性和变量之间的链接
PDLRTGotoPicture (页 2222)	查询画面存储
PDLRTInquireFreeArea (页 2224)	确定窗口的空闲区域
PDLRTOpenPicture (页 2226)	运行系统中的根画面切换
PDLRTPictureNavigation (页 2228)	启用与禁用画面浏览
PDLRTSetCursorKeys (页 2238)	定义光标控制键
PDLRTSetFocus (页 2240)	定义输入焦点
PDLRTSetLink (页 2255)	定义对象属性和变量之间的链接

PDLRTSetMultiLink (页 2258)	定义对象属性和变量之间的链接（多个变量）
PDLRTSetPropEx (页 2248)	定义对象属性

3.3.1.2 结构概览

概述

FOCUSINFO (页 2216)	输入焦点数据
LINKINFO (页 2212)	对象属性和变量的链接
MULTILINK (页 2214)	对象属性和变量的链接
MULTILINKINFO (页 2215)	对象属性和变量的链接

3.3.1.3 错误消息

概述

CMN_ERROR 错误结构中的 API 函数可返回以下错误消息：

组态

PDLCS_ELIASES_NO_ERROR	0	未发生错误。
PDLCS_ERROR_NO_DISPATCH	5	无法运行 Ole 自动
PDLCS_ERROR_NO_PROJECT	10	项目不存在
PDLCS_ERROR_NO_PICTURE	15	画面不存在
PDLCS_ERROR_NO_OBJECT	20	对象不存在
PDLCS_ERROR_NO_PROPERTY	25	属性不存在
PDLCS_ERROR_NO_OBJTYPE	30	对象类型不存在
PDLCS_ERROR_NO_DYNAMIC	32	无可用动态
PDLCS_ERROR_ALREADY_OPEN	35	指定画面已打开
PDLCS_ERROR_WRITE_PROTECTED	40	指定画面为只读画面
PDLCS_ERROR_PICTURE_SAVE	45	无法保存指定画面

PDLCS_ERROR_PICTURE_SAVE_AS	50	无法保存指定画面
PDLCS_ERROR_PICTURE_CLOSE	55	无法关闭指定画面
PDLCS_ERROR_PICTURE_CLOSE_ALL	60	无法关闭画面
PDLCS_ERROR_PICTURE_IMPORT	65	无法导入画面
PDLCS_ERROR_PICTURE_CREATE	70	创建画面时发生一般错误
PDLCS_ERROR_PICTURE_EXISTS	75	画面已存在
PDLCS_ERROR_NEW_OBJECT_NAME	80	对象名称已存在
PDLCS_ERROR_NEW_OBJECT	85	无法创建对象
PDLCS_ERROR_DELETE_OBJECT	90	无法删除对象
PDLCS_ERROR_SET_PROPERTY	95	无法设置属性
PDLCS_ERROR_GET_PROPERTY	96	无法读取属性
PDLCS_ERROR_ENUM_OBJECTTYPE	100	枚举对象类型时出错
PDLCS_ERROR_ENUM_PROPERTY	105	枚举对象属性时出错
PDLCS_ERROR_CREATE_PROPERTY	110	无法定义属性
PDLCS_ERROR_SET_LINK	115	无法建立连接
PDLCS_ERROR_REMOVE_LINK	120	无法删除连接
PDLCS_ERROR_SET_ACTION	125	无法存储动作
PDLCS_ERROR_GET_ACTION	126	无法获取动作
PDLCS_ERROR_ENUM_DYNAMICS	127	无法枚举动态
PDLCS_ERROR_SET_DIRECTCONNECT	128	无法存储直接连接
PDLCS_ERROR_AUTOMATION	130	传送期间出错
PDLCS_ERROR_MEMORY	200	无内存

运行系统

PDLRT_OK	0	未发生错误。
PDLRT_APP_NOT_RUNNING	1	PDL 应用程序未启动
PDLRT_NO_TRANSFER	2	无法访问 PDL 应用程序
PDLRT_NO_PIC	3	未选择画面
PDLRT_NO_OBJ	4	找不到对象
PDLRT_NO_PROP	5	找不到属性
PDLRT_NO_MET	6	找不到对象的方法

PDLRT_LINK_NOT_SET	7	无法连接属性
PDLRT_WNAME_NOT_UNIQUE	8	新建窗口的实例名称不唯一
PDLRT_WCREATION_FAILED	9	无法创建窗口
PDLRT_PICTURE_NOT_LOADED	10	无法加载画面文件
PDLRT_ILLEGAL_USERLEVEL	11	操作员授权级别不足
PDLRT_NO_POSITION	12	无法放置矩形
PDLRT_POSITION_WITH_MODY	13	矩形已修改
PDLRT_E_ALREADY_CONNECTED	14	与 PDLRT 的另一个连接已存在
PDLRT_PICTURE_ALREADY_OPEN	15	画面已打开
PDLRT_BAD_OLE_CONVERSION	16	通过 OLE 自动转换期间出错
PDLRT_NO_LINK	17	属性不具有动态性
PDLRT_IND_LINK_READ_ERR	18	读取间接变量时出错
PDLRT_NO_IND_LINK	19	属性不具有间接动态性
PDLRT_FAILURE_PARAM	20	错误参数

3.3.1.4 常数

寻址模式

PDLRT_AM_DEFAULT	0	画面和对象的寻址是相对的
PDLRT_AM_PICTABS	1	画面的寻址是绝对的
PDLRT_AM_OBJABS	2	对象的寻址是绝对的

可以对 PDLRT_AM_PICTABS 和 PDLRT_AM_OBJABS 值进行或运算。

PDLRTInquireFreeArea

PDLRT_IQ_ONLY	0x01	仅具有以下结果的查询：“适合/不适合”
PDLRT_IQ_MODY_POSX	0x02	可修改矩形的 x 位置，以便可以放置该矩形
PDLRT_IQ_MODY_POSY	0x04	可修改矩形的 y 位置，以便可以放置该矩形

PDLRT_IQ_MODY_HEIGHT	0x08	可修改矩形的高度，以便可以放置该矩形
PDLRT_IQ_MODY_WIDTH	0x10	可修改矩形的宽度，以便可以放置该矩形
PDLRTOpenPicture		
PDLRT_WS_STYLE_FROM_PIC	0	显示属性从画面数据中获取
PDLRT_WS_BORDER	WS_BORDER	带边框的窗口
PDLRT_WS_CAPTION	WS_CAPTION	带标题和边框的窗口
PDLRT_WS_THICKFRAME	WS_THICKFRAME	带粗边框的窗口。大小可通过鼠标操作更改。
PDLRT_WS_HSCROLL	WS_HSCROLL	带水平滚动条的窗口
PDLRT_WS_VSCROLL	WS_VSCROLL	带垂直滚动条的窗口

更新周期

名称	索引
“改变时”	0
“250 毫秒”	1
“500 毫秒”	2
“1 秒”	3
“2 秒”	4
“5 秒”	5
“10 秒”	6
“1 分钟”	7
“5 分钟”	8
“10 分钟”	9
“1 小时”	10
“用户周期 1”	11
“用户周期 2”	12
“用户周期 3”	13
“用户周期 4”	14
“用户周期 5”	15

“窗口周期”	235
“画面周期”	255

3.3.1.5 对象属性列表 (A-K)（图形编辑器）

概述

说明

对于图形编辑器的许多 API 函数，您必须初始化 lpszPropName 参数。为此，必须指定对象属性的英语名称。

不得使用 Get/Set 属性函数处理类型 VT_USERDEFINED 的属性。

不得使用 VT_DISPATCH 和其它引用；只能使用标准类型以及最多使用一个 VT_VARIANT（对于简单类型的数组属性）。

OLE 自动名称	数据类型	属性名称	特性名称	对象关联
“ActualPointLeft”	VT_I4	当前值 X	当前值 X	多边形
“ActualPointTop”	VT_I4	当前值 Y	当前值 Y	多边形
“AdaptBorder”	VT_BOOL	调整边框	调整边框	按钮、静态文本、I/O 域、文本列表、复选框、单选框
“AdaptPicture”	VT_BOOL	调整画面	调整画面	画面窗口
“AdaptSize”	VT_BOOL	调整大小	调整大小	画面窗口
“AlarmHigh”	VT_R8	限制 AH	限制 AH	棒图
“AlarmLow”	VT_R8	限制 AL	限制 AL	棒图
“Alignment”	VT_BOOL	对齐	对齐	棒图
“AlignmentLeft”	VT_I4	X 对齐	X 对齐	按钮、静态文本、I/O 域、组显示、文本列表、复选框、单选框
“AlignmentTop”	VT_I4	Y 对齐	Y 对齐	按钮、静态文本、I/O 域、组显示、文本列表、复选框、单选框

“Analog”	VT_BOOL	模拟量	模拟量	DAClockCtrl
“AngleAlpha”	VT_I4	Alpha	Alpha	3D 棒图
“AngleBeta”	VT_I4	Beta	Beta	3D 棒图
“AngleMax”	VT_R8	AngleMax	AngleMax	WinCC 量表控件，
“AngleMin”	VT_R8	AngleMin	AngleMin	WinCC 量表控件，
“Application”	VT_BSTR	WindowContent	WindowContent	打印作业/脚本诊断
“Assignments”	VT_BSTR	分配	分配	文本列表
“AssumeOnExit”	VT_BOOL	退出时应用	退出时应用	I/O 域、文本列表
“AssumeOnFull”	VT_BOOL	完整时应用	完成输入时应用	I/O 域
“自动滚动”(AutoScroll)	VT_BOOL	自动滚动	自动滚动	WinCC 报警控件（WinCC V7 之前）
“AutoSize”	VT_I4	自动调整大小	自动调整大小	PButtonCtrl
“Average”	VT_BOOL	平均值	平均值	棒图
“Axe”	VT_I4	显示轴	显示轴	3D 棒图
“AxisSection”	VT_R8	轴线部分	轴线部分	棒图
“BackBorderWidth”	VT_I4	3D 边框宽度	3D 边框宽度	按钮、组显示、圆形按钮、滚动条对象
“BackColor”	VT_I4	背景色	背景色	按钮、椭圆、椭圆扇形、圆、扇形、多边形、矩形、圆角矩形、静态文本、3D 棒图、I/O 域、棒图、图形对象、组显示、文本列表、复选框、单选框、圆形按钮、滚动条对象
“BackColor”	VT_UI4	BackColor	BackColor	DAClockCtrl、WinCC 量表控件、PButtonCtrl、SliderCtrl，

“BackColor2”	VT_I4	棒图颜色	棒图颜色	棒图
“BackColor3”	VT_I4	棒图背景色	棒图背景色	棒图
“BackColorBottom”	VT_I4	下限背景色	下限背景色	滚动条对象
“BackColorTop”	VT_I4	上限背景色	上限背景色	滚动条对象
“BackFlashColorOff”	VT_I4	闪烁背景色关	闪烁背景色关	按钮、椭圆、椭圆扇形、圆、扇形、多边形、矩形、圆角矩形、静态文本、I/O 域、棒图、图形对象、文本列表、复选框、单选框、圆形按钮、滚动条对象
“BackFlashColorOn”	VT_I4	闪烁背景色关	闪烁背景色关	按钮、椭圆、椭圆扇形、圆、扇形、多边形、矩形、圆角矩形、静态文本、I/O 域、棒图、图形对象、文本列表、复选框、单选框、圆形按钮、滚动条对象
“Background”	VT_BOOL	背景色	背景色	3D 棒图
“BackgroundPicture”	VT_USERDEFINED	BackgroundPicture	BackgroundPicture	WinCC 量表控件，
“BarBackColor”	VT_UI4	BarBackColor	BarBackColor	SliderCtrl
“BarFillColor”	VT_UI4	BarFillColor	BarFillColor	SliderCtrl
“BarDepth”	VT_I4	棒图深度	棒图深度	3D 棒图
“BarHeight”	VT_I4	棒图高度	棒图高度	3D 棒图
“BarWidth”	VT_I4	棒图宽度	棒图宽度	3D 棒图
“BasePicReferenced”	VT_BOOL	基本画面参考	基本画面参考	状态显示
“BasePicTransColor”	VT_I4	基本画面透明颜色	基本画面透明颜色	状态显示
“BasePicture”	VT_BSTR	基本画面	基本画面	状态显示
“BasePicUseTransColor”	VT_BOOL	基本画面透明颜色开	基本画面透明颜色开	状态显示

“BaseX”	VT_I4	X 基准	X 基准	3D 棒图
“BaseY”	VT_I4	Y 基准	Y 基准	3D 棒图
“Beveler”	VT_I4	Beveler	Beveler	SliderCtrl
“BevelColorDown”	VT_UI4	BevelColorDown	BevelColorDown	SliderCtrl
“BevelColorUp”	VT_UI4	BevelColorUp	BevelColorUp	SliderCtrl
“BevelOuter”	VT_I4	BevelOuter	BevelOuter	WinCC 量表控件，
“BevelWidth”	VT_I4	BevelWidth	BevelWidth	WinCC 量表控件，
“BitNumber”	VT_I4		位号	状态显示
“BorderBackColor”	VT_I4	边框背景色、线背景色	边框背景色、线背景色	按钮、椭圆、椭圆弧、椭圆扇形、圆、圆弧、扇形、线、多边形、折线、矩形、圆角矩形、静态文本、I/O 域、棒图、图形对象、文本列表、复选框、单选框、圆形按钮、滚动条对象
“BorderColor”	VT_I4	边框颜色、线背景色	边框颜色、线背景色	按钮、椭圆、椭圆弧、椭圆扇形、圆、圆弧、扇形、线、多边形、折线、矩形、圆角矩形、静态文本、3D 棒图、I/O 域、棒图、图形对象、文本列表、状态显示、复选框、单选框、圆形按钮、滚动条对象
“BorderColorBottom”	VT_I4	3D 阴影颜色	3D 阴影颜色	按钮、圆形按钮
“BorderColorTop”	VT_I4	3D 边框颜色	3D 边框颜色	按钮、圆形按钮
“BorderEndStyle”	VT_UI4	线端样式	线端样式	线、折线

“BorderFlashColorOff”	VT_I4	闪烁边框颜色关、闪烁线颜色关	闪烁边框颜色关、闪烁线颜色关	按钮、椭圆、椭圆弧、椭圆扇形、圆、圆弧、扇形、线、多边形、折线、矩形、圆角矩形、静态文本、I/O 域、棒图、图形对象、文本列表、复选框、单选框、滚动条对象
“BorderFlashColorOn”	VT_I4	闪烁边框颜色开、闪烁线颜色开	闪烁边框颜色开、闪烁线颜色开	按钮、椭圆、椭圆弧、椭圆扇形、圆、圆弧、扇形、线、多边形、折线、矩形、圆角矩形、静态文本、I/O 域、棒图、图形对象、文本列表、复选框、单选框、圆形按钮、滚动条对象
“BorderStyle”	VT_UI4	线样式	线样式	按钮、椭圆、椭圆弧、椭圆扇形、圆、圆弧、扇形、线、多边形、折线、矩形、圆角矩形、静态文本、3D 棒图、I/O 域、图形对象、文本列表、状态显示、复选框、单选框、圆形按钮、滚动条对象
“BorderWidth”	VT_I4	线宽	线宽	按钮、椭圆、椭圆弧、椭圆扇形、圆、圆弧、扇形、线、多边形、折线、矩形、圆角矩形、静态文本、3D 棒图、I/O 域、棒图、图形对象、文本列表、状态显示、复选框、单选框、圆形按钮、滚动条对象

“BorderWidth”	VT_I4	BorderWidth	BorderWidth	WinCC 量表控件，
“BoxAlignment”	VT_BOOL	框对齐	框对齐	复选框、单选框
“BoxCount”	VT_I4	框数量	框数量	复选框、单选框
“BoxType”	VT_I4	域类型	域类型	I/O 域、文本列表
“Button1Width”	VT_I4	按钮 1 宽度	按钮 1 宽度	组显示
“Button2Width”	VT_I4	按钮 2 宽度（错误：按钮 1 宽度 2）	按钮 2 宽度	组显示
“Button3Width”	VT_I4	按钮 3 宽度（错误：按钮 1 宽度 3）	按钮 3 宽度	组显示
“Button4Width”	VT_I4	按钮 4 宽度	按钮 4 宽度	组显示
“ButtonColor”	VT_I4	按钮颜色	按钮颜色	滚动条对象
“ButtonCommand”	VT_I4	ButtonCommand	ButtonCommand	WinCC 报警控件（WinCC V7 之前）
“Caption”	VT_BOOL	标题	标题	打印作业/脚本诊断、画面窗口
“Caption”	VT_BSTR	说明	说明	WinCC 量表控件、PButtonCtrl、SliderCtrl，
“CaptionColor”	VT_UI4	CaptionColor	CaptionColor	WinCC 量表控件，
“CaptionFont”	VT_USERDEFINED	CaptionFont	CaptionFont	WinCC 量表控件，
“CaptionOffset”	VT_R8	CaptionOffset	CaptionOffset	WinCC 量表控件，
“CellCut”	VT_BOOL	CellCut	CellCut	WinCC 报警控件（WinCC V7 之前）
“CenterColor”	VT_UI4	CenterColor	CenterColor	WinCC 量表控件，
“CenterScale”	VT_R4	CenterScale	CenterScale	WinCC 量表控件，
“CheckAlarmHigh”	VT_BOOL	监视 AH	监视 AH	棒图
“CheckAlarmLow”	VT_BOOL	监视 AL	监视 AL	棒图
“CheckLimitHigh4”	VT_BOOL	监视 RH4	监视 RH4	棒图

“CheckLimitHigh5”	VT_BOOL	监视 RH5	监视 RH5	棒图
“CheckLimitLow4”	VT_BOOL	监视 RL4	监视 RL4	棒图
“CheckLimitLow5”	VT_BOOL	监视 RL5	监视 RL5	棒图
“CheckToleranceHigh”	VT_BOOL	监视 TH	监视 TH	棒图
“CheckToleranceLow”	VT_BOOL	监视 TL	监视 TL	棒图
“CheckWarningHigh”	VT_BOOL	监视 WH	监视 WH	棒图
“CheckWarningLow”	VT_BOOL	监视 WL	监视 WL	棒图
“ClearOnError”	VT_BOOL	输入无效时清除	输入无效时清除	I/O 域
“ClearOnNew”	VT_BOOL	输入时清除	输入时清除	I/O 域
“CloseButton”	VT_BOOL	可关闭	可关闭	打印作业/脚本诊断、画面窗口
“CollectValue”	VT_UI4	组值	组值	组显示
“ColMove”	VT_BOOL	ColMove	ColMove	WinCC 报警控件（WinCC V7 之前）
“ColorAlarmHigh”	VT_I4	棒图颜色 AH	棒图颜色 AH	棒图
“ColorAlarmLow”	VT_I4	棒图颜色 AL	棒图颜色 AL	棒图
“ColorBottom”	VT_I4	下限颜色	下限颜色	滚动条对象
“ColorChangeType”	VT_BOOL	更改颜色	更改颜色	棒图
“ColorLimitHigh4”	VT_I4	棒图颜色 RH4	棒图颜色 RH4	棒图
“ColorLimitHigh5”	VT_I4	棒图颜色 RH5	棒图颜色 RH5	棒图
“ColorLimitLow4”	VT_I4	棒图颜色 RL4	棒图颜色 RL4	棒图
“ColorLimitLow5”	VT_I4	棒图颜色 RL5	棒图颜色 RL5	棒图
“ColorToleranceHigh”	VT_I4	棒图颜色 TH	棒图颜色 TH	棒图

“ColorToleranceLow”	VT_I4	棒图颜色 TL	棒图颜色 TL	棒图
“ColorTop”	VT_I4	上限颜色	上限颜色	滚动条对象
“ColorWarningHigh”	VT_I4	棒图颜色 WH	棒图颜色 WH	棒图
“ColorWarningLow”	VT_I4	棒图颜色 WL	棒图颜色 WL	棒图
“ColTitle”	VT_BOOL	ColTitle	ColTitle	WinCC 报警控件（WinCC V7 之前）
“ColWidth”	VT_BOOL	ColWidth	ColWidth	WinCC 报警控件（WinCC V7 之前）
“CursorControl”	VT_BOOL	光标控制	光标控制	I/O 域、文本列表
“CursorMode”	VT_BOOL	CursorMode	CursorMode	WinCC 报警控件（WinCC V7 之前）、CCAxAlarmControl
“Danger”	VT_R4	危险	危险	WinCC 量表控件，
“DangerColor”	VT_UI4	DangerColor	DangerColor	WinCC 量表控件，
“DataFormat”	VT_I4	数据格式	数据格式	I/O 域
“DataLanguage”	VT_I4	组态语言	组态语言	按钮、静态文本、复选框、单选框
“Delta”	VT_R4	增量	增量	WinCC 量表控件，
“Direction”	VT_I4	棒图对齐	棒图对齐	3D 棒图、滚动条对象、棒图
“EditAtOnce”	VT_BOOL	立即输入	立即输入	I/O 域、文本列表
“EndAngle”	VT_I4	终止角度	终止角度	椭圆弧、椭圆扇形、圆弧、扇形
“Exponent”	VT_BOOL	指数显示	指数显示	棒图
“ExtendedOperation”	VT_BOOL	扩展操作	扩展操作	滚动条对象

“FillColor”	VT_I4	填充图案颜色	填充图案颜色	按钮、椭圆、椭圆扇形、圆、扇形、多边形、矩形、圆角矩形、静态文本、I/O域、棒图、图形对象、文本列表、复选框、单选框、圆形按钮、滚动条对象
“Filling”	VT_BOOL	动态填充	动态填充	按钮、椭圆、椭圆扇形、圆、扇形、多边形、矩形、圆角矩形、静态文本、图形对象、复选框、单选框、圆形按钮
“FillingIndex”	VT_I4	填充量	填充量	按钮、椭圆、椭圆扇形、圆、扇形、多边形、矩形、圆角矩形、静态文本、图形对象、复选框、单选框、圆形按钮、滚动条对象
“FillStyle”	VT_UI4	填充图案	填充图案	按钮、椭圆、椭圆扇形、圆、扇形、多边形、矩形、圆角矩形、静态文本、I/O域、棒图、图形对象、文本列表、复选框、单选框、圆形按钮、滚动条对象
“FillStyle2”	VT_UI4	棒图图案	棒图图案	棒图
“FlashBackColor”	VT_BOOL	闪烁背景激活	闪烁背景激活	按钮、椭圆、椭圆扇形、圆、扇形、多边形、矩形、圆角矩形、静态文本、I/O域、棒图、图形对象、文本列表、复选框、单选框、圆形按钮、滚动条对象

“FlashBorderColor”	VT_BOOL	闪烁线激活	闪烁线激活	按钮、椭圆、椭圆弧、椭圆扇形、圆、圆弧、扇形、线、多边形、折线、矩形、圆角矩形、静态文本、I/O 域、棒图、图形对象、文本列表、复选框、单选框、圆形按钮、滚动条对象
“FlashFlashPicture”	VT_BOOL	闪烁画面激活	闪烁画面激活	状态显示
“FlashForeColor”	VT_BOOL	闪烁文本激活	闪烁文本激活	按钮、静态文本、I/O 域、文本列表、复选框、单选框
“FlashPicReferenced”	VT_BOOL	闪烁画面参考	闪烁画面参考	状态显示
“FlashPicTransColor”	VT_I4	基本画面透明颜色	基本画面透明颜色	状态显示
“FlashPicture”	VT_BSTR	闪烁画面	闪烁画面	状态显示
“FlashPicUseTransColor”		闪烁画面透明颜色开	闪烁画面透明颜色开	状态显示
“FlashRate”	VT_I4	闪烁频率	闪烁频率	组显示
“FlashRateBackColor”	VT_I4	背景闪烁频率	背景闪烁频率	按钮、椭圆、椭圆扇形、圆、扇形、多边形、矩形、圆角矩形、静态文本、I/O 域、棒图、图形对象、文本列表、复选框、单选框、圆形按钮、滚动条对象

“FlashRateBorderColor”	VT_I4	闪烁线频率	背景闪烁频率	按钮、椭圆、椭圆弧、椭圆扇形、圆、圆弧、扇形、线、多边形、折线、矩形、圆角矩形、静态文本、I/O 域、棒图、图形对象、文本列表、复选框、单选框、圆形按钮、滚动条对象
“FlashRateFlashPic”	VT_I4	闪烁画面闪烁频率	闪烁画面闪烁频率	状态显示
“FlashRateForeColor”	VT_I4	文本闪烁频率	文本闪烁频率	按钮、静态文本、I/O 域、文本列表、复选框、单选框
“Font”	VT_USERDEFINED	字体	字体	DAClockCtrl、PButtonCtrl、SliderCtrl
“FontBold”	VT_BOOL	粗体	粗体	按钮、静态文本、I/O 域、棒图、组显示、文本列表、复选框、单选框
“FontBold”	VT_BOOL	FontBold	FontBold	PButtonCtrl
“FontItalic”	VT_BOOL	斜体	斜体	按钮、静态文本、I/O 域、组显示、文本列表、复选框、单选框
“FontItalic”	VT_BOOL	FontItalic	FontItalic	PButtonCtrl
“FontName”	VT_BSTR	字体	字体	按钮、静态文本、I/O 域、组显示、文本列表、复选框、单选框
“FontName”	VT_BSTR	FontName	FontName	PButtonCtrl
“FontPosition”	VT_USERDEFINED	FontPosition	FontPosition	SliderCtrl
“FontSize”	VT_I4	字号	字号	按钮、静态文本、I/O 域、组显示、文本列表、复选框、单选框

“FontSize”	VT_BSTR	FontSize	字号	PButtonCtrl
“FontStrikeThru”	VT_BOOL	FontStrikeThru	FontStrikeThru	PButtonCtrl
“FontUnderline”	VT_BOOL	下划线	下划线	按钮、静态文本、I/O 域、组显示、文本列表、复选框、单选框
“FontUnderline”	VT_BOOL	FontUnderline	FontUnderline	PButtonCtrl
“ForeColor”	VT_I4	字体颜色	字体颜色	按钮、静态文本、I/O 域、组显示、文本列表、复选框、单选框
“ForeColor”	VT_UI4	ForeColor	ForeColor	DAClockCtrl、PButtonCtrl、SliderCtrl
“ForeFlashColorOff”	VT_I4	闪烁文本颜色关	闪烁文本颜色关	按钮、静态文本、I/O 域、组显示、文本列表、复选框、单选框
“ForeFlashColorOn”	VT_I4	闪烁文本颜色开	闪烁文本颜色开	按钮、静态文本、I/O 域、组显示、文本列表、复选框、单选框
“FrameColor”	VT_UI4	边框颜色	边框颜色	WinCC 量表控件，
“FrameColorDown”	VT_UI4	FrameColorDown	FrameColorDown	PButtonCtrl
“FrameColorUp”	VT_UI4	FrameColorUp	FrameColorUp	PButtonCtrl
“FramePicture”	VT_USERDEFINED	FramePicture	FramePicture	WinCC 量表控件，
“FrameScale”	VT_R4	FrameScale	FrameScale	WinCC 量表控件，
“FrameSize”		FrameSize	FrameSize	PButtonCtrl
“FrameWidth”	VT_I4	FrameWidth	FrameWidth	PButtonCtrl
“Freeze”	VT_BOOL	冻结	冻结	WTVctrlCtrl
“GridLineHorz”	VT_BOOL	GridLineHorz	GridLineHorz	WinCC 报警控件（WinCC V7 之前）
“GridLineVert”	VT_BOOL	GridLineVert	GridLineVert	WinCC 报警控件（WinCC V7 之前）

“Height”	VT_I4	高度	高度	按钮、椭圆、椭圆弧、椭圆扇形、圆、圆弧、扇形、线、多边形、折线、矩形、圆角矩形、静态文本、3D 棒图、I/O 字段、棒图、图形对象、组显示、文本列表、状态显示、复选框、单选框、圆形按钮、滚动条对象、DAClockCtrl、WinCC 量表控件、WinCC 报警控件（WinCC V7 之前）、PButtonCtrl、SliderCtrl、WTVctrlCtrl，
“Height”	VT_I4	窗口高度	窗口高度	打印作业/脚本诊断、画面窗口
“HiddenInput”	VT_BOOL	隐藏输入	隐藏输入	I/O 域
“Hotkey”	VT_UI4	热键	热键	按钮
“HourNeedleHeight”	VT_I4	HourNeedleHeight	HourNeedleHeight	DAClockCtrl
“HourNeedleWidth”	VT_I4	HourNeedleWidth	HourNeedleWidth	DAClockCtrl
“Hysteresis”	VT_BOOL	滞后	滞后	棒图
“HysteresisRange”	VT_R8	滞后范围	滞后范围	棒图
“Index”	VT_I4	索引	索引	多边形、折线、状态显示、复选框、单选框
“InnerBevelOffset”	VT_I4	InnerBevelOffset	InnerBevelOffset	SliderCtrl
“InnerBevelStyle”	VT_I4	InnerBevelStyle	InnerBevelStyle	SliderCtrl

“InnerBevelWidth”	VT_I4	InnerBevelWidth	InnerBevelWidth	SliderCtrl
“ItemBorderBackColor”	VT_I4	分割背景色	分割背景色	文本列表
“ItemBorderColor”	VT_I4	分割线颜色	分割线颜色	文本列表
“ItemBorderStyle”	VT_UI4	分割线样式	分割线样式	文本列表

3.3.1.6 列出对象属性 (L-Z)（图形编辑器）

概述

说明

对于图形编辑器的许多 API 函数，您必须初始化 lpszPropName 参数。为此，必须指定对象属性的英语名称。

不得使用 Get/Set 属性函数处理类型 VT_USERDEFINED 的属性。

不得使用 VT_DISPATCH 和其它引用；只能使用标准类型以及最多使用一个 VT_VARIANT（对于简单类型的数组属性）。

OLE 自动名称	数据类型	属性名称	特性名称	对象关联
“LanguageSwitch”	VT_BOOL	语言切换	语言切换	文本列表
“Layer”	VT_I4	图层	图层	按钮、椭圆、椭圆弧、椭圆扇形、圆、圆弧、扇形、线、多边形、折线、矩形、圆角矩形、静态文本、3D 棒图、打印作业/脚本诊断、画面窗口、I/O 字段、棒图、图形对象、组显示、文本列表、状态显示、复选框、单选框、圆形按钮、滚动条对象、WinCC 量表控件、WinCC 报警控件（WinCC V7 之前）、PButtonCtrl、SliderCtrl、WTVctrlCtrl，

“Layer00Checked”	VT_BOOL	监视 0	监视 0	3D 棒图
“Layer01Checked”	VT_BOOL	监视 1	监视 1	3D 棒图
“Layer02Checked”	VT_BOOL	监视 2	监视 2	3D 棒图
“Layer03Checked”	VT_BOOL	监视 3	监视 3	3D 棒图
“Layer04Checked”	VT_BOOL	监视 4	监视 4	3D 棒图
“Layer05Checked”	VT_BOOL	监视 5	监视 5	3D 棒图
“Layer06Checked”	VT_BOOL	监视 6	监视 6	3D 棒图
“Layer07Checked”	VT_BOOL	监视 7	监视 7	3D 棒图
“Layer08Checked”	VT_BOOL	监视 8	监视 8	3D 棒图
“Layer09Checked”	VT_BOOL	监视 9	监视 9	3D 棒图
“Layer10Checked”	VT_BOOL	监视 10	监视 10	3D 棒图
“Layer00Color”	VT_I4	棒图颜色 0	棒图颜色 0	3D 棒图
“Layer01Color”	VT_I4	棒图颜色 1	棒图颜色 1	3D 棒图
“Layer02Color”	VT_I4	棒图颜色 2	棒图颜色 2	3D 棒图
“Layer03Color”	VT_I4	棒图颜色 3	棒图颜色 3	3D 棒图
“Layer04Color”	VT_I4	棒图颜色 4	棒图颜色 4	3D 棒图
“Layer05Color”	VT_I4	棒图颜色 5	棒图颜色 5	3D 棒图
“Layer06Color”	VT_I4	棒图颜色 6	棒图颜色 6	3D 棒图
“Layer07Color”	VT_I4	棒图颜色 7	棒图颜色 7	3D 棒图
“Layer08Color”	VT_I4	棒图颜色 8	棒图颜色 8	3D 棒图
“Layer09Color”	VT_I4	棒图颜色 9	棒图颜色 9	3D 棒图
“Layer10Color”	VT_I4	棒图颜色 10	棒图颜色 10	3D 棒图
“Layer00Value”	VT_R8	限值 0	限值 0	3D 棒图

“Layer01Value”	VT_R8	限值 1	限值 1	3D 棒图
“Layer02Value”	VT_R8	限值 2	限值 2	3D 棒图
“Layer03Value”	VT_R8	限值 3	限值 3	3D 棒图
“Layer04Value”	VT_R8	限值 4	限值 4	3D 棒图
“Layer05Value”	VT_R8	限值 5	限值 5	3D 棒图
“Layer06Value”	VT_R8	限值 6	限值 6	3D 棒图
“Layer07Value”	VT_R8	限值 7	限值 7	3D 棒图
“Layer08Value”	VT_R8	限值 8	限值 8	3D 棒图
“Layer09Value”	VT_R8	限值 9	限值 9	3D 棒图
“Layer10Value”	VT_R8	限值 10	限值 10	3D 棒图
“Left”	VT_I4	X 位置	X 位置	按钮、椭圆、椭圆弧、椭圆扇形、圆、圆弧、扇形、线、多边形、折线、矩形、圆角矩形、静态文本、3D 棒图、打印作业/脚本诊断、画面窗口、I/O 字段、棒图、图形对象、组显示、文本列表、状态显示、复选框、单选框、圆形按钮、滚动条对象、WinCC 量表控件、WinCC 报警控件（WinCC V7 之前）、PButtonCtrl、SliderCtrl、WTVctrlCtrl，
“LeftComma”	VT_I4	小数点左边的位数	小数点左边的位数	棒图
“LightEffect”	VT_BOOL	淡化效果	淡化效果	3D 棒图
“LimitHigh4”	VT_R8	限制 RH4	限制 RH4	棒图
“LimitHigh5”	VT_R8	限制 RH5	限制 RH5	棒图
“LimitLow4”	VT_R8	限制 RL4	限制 RL4	棒图
“LimitLow5”	VT_R8	限制 RL5	限制 RL5	棒图
“LimitMax”	VT_R8	上限值	上限	IO 域
“LimitMin”	VT_R8	下限值	下限	IO 域
“LineFont”	VT_BOOL	LineFont	LineFont	WinCC 报警控件（WinCC V7 之前）

“LineHeight”	VT_BOOL	LineHeight	LineHeight	WinCC 报警控件（WinCC V7 之前）
“LineTitle”	VT_BOOL	LineTitle	LineTitle	WinCC 报警控件（WinCC V7 之前）
“ListType”	VT_I4	列表类型	列表类型	文本列表
“LockBackColor”	VT_I4	锁定背景色	锁定背景色	组显示
“LockStatus”	VT_BOOL	锁定显示	锁定显示	组显示
“LockText”	VT_BOOL	锁定显示文本	锁定显示文本	组显示
“LockTextColor”	VT_I4	锁定字体颜色	锁定字体颜色	组显示
“LongStrokesBold”	VT_BOOL	长轴线部分	长轴线部分	棒图
“LongStrokesOnly”	VT_BOOL	仅长轴线部分	仅长轴线部分	棒图
“LongStrokesSize”	VT_UI4	轴线部分的长度	轴线部分的长度	棒图
“LongStrokesTextEach”	VT_UI4	标记每一个	标记每一个	棒图
“Marker”	VT_BOOL	限制标记	限制标记	棒图
“Max”	VT_R8	最大值	最大值	3D 棒图、滚动条对象、棒图
“MaximizeButton”	VT_BOOL	可以最大化	可以最大化	打印作业/脚本诊断、画面窗口
“MCGUBackColorOff”	VT_I4	离开未确认 - 背景色关	离开未确认 - 背景色关	组显示
“MCGUBackColorOn”	VT_I4	离开未确认 - 背景色开	离开未确认 - 背景色开	组显示
“MCGUBackFlash”	VT_BOOL	离开未确认 - 背景闪烁	离开未确认 - 背景闪烁	组显示
“MCGUTextColorOff”	VT_I4	离开未确认 - 文本颜色关	进入 - 背景色关	组显示
“MCGUTextColorOn”	VT_I4	离开未确认 - 文本颜色开	进入 - 背景色开	组显示
“MCGUTextFlash”	VT_BOOL	离开未确认 - 文本闪烁	离开未确认 - 背景闪烁	组显示

“MCKOBackColorOff”	VT_I4	进入 - 背景色关	进入 - 背景色关	组显示
“MCKOBackColorOn”	VT_I4	进入 - 背景色开	进入 - 背景色开	组显示
“MCKOBackFlash”	VT_BOOL	进入 - 背景闪烁	进入 - 背景闪烁	组显示
“MCKOTextColorOff”	VT_I4	进入 - 文本颜色关	进入 - 文本颜色关	组显示
“MCKOTextColorOn”	VT_I4	进入 - 文本颜色开	进入 - 文本颜色开	组显示
“MCKOTextFlash”	VT_BOOL	进入 - 文本闪烁	进入 - 文本闪烁	组显示
“MCKQBackColorOff”	VT_I4	进入确认 - 背景色关	进入确认 - 背景色关	组显示
“MCKQBackColorOn”	VT_I4	进入确认 - 背景色开	进入确认 - 背景色开	组显示
“MCKQBackFlash”	VT_BOOL	进入确认 - 背景闪烁	进入确认 - 背景闪烁	组显示
“MCKQTextColorOff”	VT_I4	进入确认 - 文本颜色关	进入确认 - 文本颜色关	组显示
“MCKQTextColorOn”	VT_I4	进入确认 - 文本颜色开	进入确认 - 文本颜色开	组显示
“MCKQTextFlash”	VT_BOOL	进入确认 - 文本闪烁	进入确认 - 文本闪烁	组显示
“MCText”	VT_BOOL	显示文本	显示文本	组显示
“MessageClass”	VT_I4	消息类型	消息类型	组显示
“Min”	VT_R8	最小值	最小值	组显示
“MinuteNeedleHeight”	VT_I4	MinuteNeedleHeight	MinuteNeedleHeight	组显示
“MinuteNeedleWidth”	VT_I4	MinuteNeedleWidth	MinuteNeedleWidth	组显示
“Moveable”	VT_BOOL	可移动	可移动	打印作业/脚本诊断、画面窗口
“NeedleColor”	VT_UI4	NeedleColor	NeedleColor	WinCC 量表控件，

“NormalColor”	VT_UI4	NormalColor	NormalColor	WinCC 量表控件，
“NumberLines”	VT_I4	可见行数	可见行数	文本列表、复选框、单选框
“ObjectName”	VT_BSTR	对象名称	对象名称	按钮、椭圆、椭圆弧、椭圆扇形、圆、圆弧、扇形、线、多边形、折线、矩形、圆角矩形、静态文本、3D 棒图、打印作业/脚本诊断、I/O 字段、棒图、图形对象、组显示、文本列表、状态显示、复选框、单选框、圆形按钮、滚动条对象、DAClockCtrl、WinCC 量表控件、WinCC 报警控件（WinCC V7 之前）、PButtonCtrl、SliderCtrl、WTVctrlCtrl，
“OffsetLeft”	VT_I4	画面偏移量 X	画面偏移量 X	画面窗口
“OffsetTop”	VT_I4	画面偏移量 Y	画面偏移量 Y	画面窗口
“OnTop”	VT_BOOL	前景	前景	打印作业/脚本诊断、画面窗口
“Operation”	VT_BOOL	允许操作员控制	允许操作员控制	按钮、椭圆、椭圆弧、椭圆扇形、圆、圆弧、扇形、线、多边形、折线、矩形、圆角矩形、静态文本、3D 棒图、I/O 字段、棒图、图形对象、组显示、文本列表、状态显示、复选框、单选框、圆形按钮、滚动条对象、DAClockCtrl、WinCC 量表控件、WinCC 报警控件（WinCC V7 之前）、PButtonCtrl、SliderCtrl、WTVctrlCtrl，

“OperationMessage”	VT_BOOL	操作员输入消息	操作员输入消息	I/O 域、文本列表、复选框、单选框、滚动条对象
“OperationReport”	VT_BOOL	操作员操作报告	操作员操作报告	I/O 域、文本列表、滚动条对象
“Orientation”	VT_BOOL	文本方向	文本方向	静态文本、I/O 域、文本列表、复选框、单选框
“OuterBevelOffset”	VT_I4	OuterBevelOffset	OuterBevelOffset	SliderCtrl
“OuterBevelStyle”	VT_I4	OuterBevelStyle	OuterBevelStyle	SliderCtrl
“OuterBevelWidth”	VT_I4	OuterBevelWidth	OuterBevelWidth	SliderCtrl
“Outline”	VT_BOOL	框线	框线	PButtonCtrl
“OutputFormat”	VT_BSTR	输出格式	输出格式	I/O 域
“OutputValue”	VT_R8	输出值	输出值	I/O 域、文本列表
“PasswordLevel”	VT_UI4	用户级别	密码	按钮、椭圆、椭圆弧、椭圆扇形、圆、圆弧、扇形、线、多边形、折线、矩形、圆角矩形、静态文本、3D 棒图、I/O 域、棒图、图形对象、组显示、文本列表、状态显示、复选框、单选框、圆形按钮、滚动条对象
“PicDeactReferenced”	VT_BOOL	画面取消激活已引用	画面取消激活已引用	圆形按钮
“PicDeactTransparent”	VT_I4	画面取消激活透明颜色	画面取消激活透明颜色	圆形按钮
“PicDeactUseTransColor”	VT_BOOL	画面取消激活透明颜色打开	画面取消激活透明颜色打开	圆形按钮
“PicDownReferenced”	VT_BOOL	画面打开参考	画面打开参考	圆形按钮
“PicDownTransparent”	VT_I4	画面打开透明颜色	画面打开透明颜色	圆形按钮
“PicDownUseTransColor”	VT_BOOL	画面打开透明颜色打开	画面打开透明颜色打开	圆形按钮

“PicReferenced”	VT_BOOL	参考画面	参考画面	图形对象
“PicTransColor”	VT_I4	画面透明颜色	画面透明颜色	图形对象
“Picture”	VT_USER DEFINED	画面	画面	DAClockCtrl
“PictureBack”	VT_USER DEFINED	PictureBack	PictureBack	SliderCtrl
“PictureDeactivated”	VT_BSTR	画面状态关闭	画面状态关闭	圆形按钮
“PictureDown”	VT_BSTR	画面状态开	画面状态开	按钮、圆形按钮
“PictureName”	VT_BSTR	画面名称	画面名称	画面窗口、图形对象
“PictureSelected”	VT_USER DEFINED	已选择画面	已选择画面	PButtonCtrl
“PictureUnselected”	VT_USER DEFINED	未选择画面	未选择画面	PButtonCtrl
“PictureThumb”	VT_USER DEFINED	PictureThumb	PictureThumb	SliderCtrl
“PictureUp”	VT_BSTR	画面状态关	画面状态关	按钮、圆形按钮
“PicUpReferenced”	VT_BOOL	画面关闭参考	画面关闭参考	圆形按钮
“PicUpTransparent”	VT_I4	画面关闭透明颜色	画面关闭透明颜色	圆形按钮
“PicUpUseTransColor”	VT_BOOL	画面关闭透明颜色打开	画面关闭透明颜色打开	圆形按钮
“PicUseTransColor”	VT_BOOL	画面透明颜色打开	透明颜色打开	图形对象
“PointCount”	VT_I4	角的数目	角的数目	多边形、折线
“Position”	VT_I4	定位	定位	SliderCtrl
“PredefinedAngles”	VT_I4	角度设置	角度设置	3D 棒图
“Pressed”	VT_BOOL	已按下	已按下	圆形按钮
“Process”	VT_R8	过程驱动连接	过程驱动连接	3D 棒图、滚动条对象、棒图
“Process”	VT_R8	选择框	选择框	复选框、单选框

“ProjectPath”	VT_BSTR	ProjectPath	ProjectPath	WinCC 报警控件（WinCC V7 之前）
“Radius”	VT_I4	半径	半径	圆、圆弧、扇形、圆形按钮
“RadiusHeight”	VT_I4	Y 半径	Y 半径	椭圆、椭圆弧、椭圆扇形
“RadiusWidth”	VT_I4	X 半径	X 半径	椭圆、椭圆弧、椭圆扇形
“RangeMax”	VT_I4	RangeMax	RangeMax	SliderCtrl
“RangeMin”	VT_I4	RangeMin	RangeMin	SliderCtrl
“ReferenceRotationLeft”	VT_I4	旋转参考坐标 X	旋转参考坐标 X	线、多边形、折线
“ReferenceRotationTop”	VT_I4	旋转参考坐标 Y	旋转参考坐标 Y	线、多边形、折线
“RefreshTimerPeriod”	VT_I4	RefreshTimerPeriod	RefreshTimerPeriod	WTVctrlCtrl
“Relevant”	VT_BOOL	相关组	相关组	组显示
“RightComma”	VT_I4	小数点右边的位数	小数点右边的位数	棒图
“RotationAngle”	VT_I4	旋转角	旋转角	线、多边形、折线
“RoundCornerHeight”	VT_I4	角半径 Y	角半径 Y	圆角矩形
“RoundCornerWidth”	VT_I4	角半径 X	角半径 X	圆角矩形
“SameSize”	VT_BOOL	同样大小	同样大小	组显示
“ScaleColor”	VT_I4	标尺颜色	标尺颜色	棒图
“ScaleTicks”	VT_I4	标尺标记	标尺标记	棒图
“Scaling”	VT_BOOL	标尺	标尺	棒图
“ScalingType”	VT_I4	棒图标尺	棒图标尺	棒图
“ScrollBars”	VT_BOOL	滚动条	滚动条	画面窗口
“SecondNeedleHeight”	VT_I4	SecondNeedleHeight	SecondNeedleHeight	DAClockCtrl
“SecondNeedleWidth”	VT_I4	SecondNeedleWidth	SecondNeedleWidth	DAClockCtrl
“SelBGColor”	VT_I4	选择背景色	选择背景色	文本列表

“SelectionMode”	VT_I2	SelectionMode	SelectionMode	WinCC 报警控件（WinCC V7 之前）
“SelTextColor”	VT_I4	选择字体颜色	选择字体颜色	文本列表
“ServerName”	VT_BSTR	服务器名称	服务器名称	DAClockCtrl、WinCC 量表控件、WinCC 报警控件（WinCC V7 之前）、PButtonCtrl、SliderCtrl、WTVctrlCtrl，
“SignificantMask”	VT_UI4	位模式组显示	位模式组显示	组显示
“Sizeable”	VT_BOOL	可调整大小	可调整大小	打印作业/脚本诊断、画面窗口
“ShowBar”	VT_BOOL	ShowBar	ShowBar	SliderCtrl
“ShowDanger”	VT_BOOL	ShowDanger	ShowDanger	WinCC 量表控件，
“ShowDecimalPoint”	VT_BOOL	ShowDecimalPoint	ShowDecimalPoint	WinCC 量表控件，
“ShowNormal”	VT_BOOL	ShowNormal	ShowNormal	WinCC 量表控件，
“ShowPeak”	VT_BOOL	ShowPeak	ShowPeak	WinCC 量表控件，
“ShowPosition”	VT_BOOL	ShowPosition	ShowPosition	SliderCtrl
“ShowThumb”	VT_BOOL	ShowThumb	ShowThumb	SliderCtrl
“StartAngle”	VT_I4	起始角度	起始角度	椭圆弧、椭圆扇形、圆弧、扇形
“Template”	VT_BSTR	模板	模板	打印作业/脚本诊断
“TemplateName”	VT_BSTR	TemplateName	TemplateName	WinCC 报警控件（WinCC V7 之前）
“文本”	VT_BSTR	文本	文本	按钮、静态文本、复选框、单选框
“ThumbBackColor”	VT_UI4	ThumbBackColor	ThumbBackColor	SliderCtrl
“TicColor”	VT_UI4	TicColor	TicColor	WinCC 量表控件，
“TicFont”	VT_USER DEFINED	TicFont	TicFont	WinCC 量表控件，
“Ticks”	VT_BOOL	标记	标记	DAClockCtrl
“TicksColor”	VT_UI4	TicksColor	TicksColor	DAClockCtrl

“TickStyle”	VT_I4	TickStyle	TickStyle	SliderCtrl
“TicOffset”	VT_R8	TicOffset	TicOffset	WinCC 量表控件，
“TicTextColor”	VT_UI4	TicTextColor	TicTextColor	WinCC 量表控件，
“TicTextOffset”	VT_R8	TicTextOffset	TicTextOffset	WinCC 量表控件，
“TicWidth”	VT_R8	TicWidth	TicWidth	WinCC 量表控件，
“TitleCut”	VT_BOOL	TitleCut	TitleCut	WinCC 报警控件（WinCC V7 之前）
“Toggle”	VT_BOOL	自锁	自锁	圆形按钮
“ToleranceHigh”	VT_R8	限制 TH	限制 TH	棒图
“ToleranceLow”	VT_R8	限制 TL	限制 TL	棒图
“ToolTipText”	VT_BSTR	工具提示文本	工具提示文本	按钮、椭圆、椭圆弧、椭圆扇形、圆、圆弧、扇形、线、多边形、折线、矩形、圆角矩形、静态文本、3D 棒图、I/O 字段、棒图、图形对象、组显示、文本列表、状态显示、复选框、单选框、圆形按钮、滚动条对象、DAClockCtrl、WinCC 量表控件、WinCC 报警控件（WinCC V7 之前）、PButtonCtrl、SliderCtrl、WTVctrlCtrl，

“Top”	VT_I4	Y 位置	Y 位置	按钮、椭圆、椭圆弧、椭圆扇形、圆、圆弧、扇形、线、多边形、折线、矩形、圆角矩形、静态文本、3D 棒图、打印作业/脚本诊断、画面窗口、I/O 字段、棒图、图形对象、组显示、文本列表、状态显示、复选框、单选框、圆形按钮、滚动条对象、DAClockCtrl、WinCC 量表控件、WinCC 报警控件（WinCC V7 之前）、PButtonCtrl、SliderCtrl、WTVctrlCtrl，
“Trend”	VT_BOOL	趋势	趋势	棒图
“TrendColor”	VT_I4	趋势颜色	趋势颜色	棒图
“TypeAlarmHigh”	VT_BOOL	类型 AH	类型 AH	棒图
“TypeAlarmLow”	VT_BOOL	类型 AL	类型 AL	棒图
“TypeLimitHigh4”	VT_BOOL	类型 RH4	类型 RH4	棒图
“TypeLimitHigh5”	VT_BOOL	类型 RH5	类型 RH5	棒图
“TypeLimitLow4”	VT_BOOL	类型 RL4	类型 RL4	棒图
“TypeLimitLow5”	VT_BOOL	类型 RL5	类型 RL5	棒图
“TypeToleranceHigh”	VT_BOOL	类型 TH	类型 TH	棒图
“TypeToleranceLow”	VT_BOOL	类型 TL	类型 TL	棒图
“TypeWarningHigh”	VT_BOOL	类型 WH	类型 WH	棒图
“TypeWarningLow”	VT_UI4	类型 WL	类型 WL	棒图

“UnitColor”	VT_USER DEFINED	UnitColor	UnitColor	WinCC 量表控件，
“UnitFont”	VT_BSTR	UnitFont	UnitFont	WinCC 量表控件，
“UnitText”	VT_BSTR	UnitText	UnitText	WinCC 量表控件，
“UnitOffset”	VT_R8	UnitOffset	UnitOffset	WinCC 量表控件，
“UnselBGColor”	VT_I4	列表背景颜色	列表背景颜色	文本列表
“UnselTextColor”	VT_I4	列表字体颜色	列表字体颜色	文本列表
“UpdateCycle”	VT_I4	更新周期	更新周期	画面窗口
“UseRefreshTimer”	VT_BOOL	UseRefreshTimer	UseRefreshTimer	WTVctrlCtrl
“UserValue1”	VT_UI4	用户值 1	用户值 1	组显示
“UserValue2”	VT_UI4	用户值 2	用户值 2	组显示
“UserValue3”	VT_UI4	用户值 3	用户值 3	组显示
“UserValue4”	VT_UI4	用户值 4	用户值 4	组显示
“Value”	VT_R4	值	值	WinCC 量表控件，
“ValueMax”	VT_R4	ValueMax	ValueMax	WinCC 量表控件，
“ValueMin”	VT_R4	ValueMin	ValueMin	WinCC 量表控件，
“VideoEnabled”	VT_BOOL	VideoEnabled	VideoEnabled	WTVctrlCtrl
“VideoSource”	VT_I2	VideoSource	VideoSource	WTVctrlCtrl
“可见”	VT_BOOL	显示	显示	所有对象（包括画面）
“Warning”	VT_R4	警告	警告	WinCC 量表控件，
“WarningColor”	VT_UI4	WarningColor	WarningColor	WinCC 量表控件，
“WarningHigh”	VT_R8	限制 WH	限制 WH	棒图
“WarningLow”	VT_R8	限制 WL	限制 WL	棒图
“Width”	VT_I4	Width	Width	所有对象，以下除外：应用程序窗口、画面窗口
“WindowBorder”	VT_BOOL	边框	边框	应用程序窗口、画面窗口、打印作业/脚本诊断
“WindowsStyle”	VT_BOOL	窗口样式	窗口样式	按钮、滚动条对象
“WindowType”	VT_I2	WindowType	WindowType	WinCC 报警控件（WinCC V7 之前）
“WinTVInformation”	VT_BSTR		WinTVInformation	WTVctrlCtrl

“WithAxes”	VT_BOOL	WithAxes	WithAxes	SliderCtrl
“WithLabels”	VT_BOOL	WithLabels	WithLabels	SliderCtrl
“ZeroPoint”	VT_UI4	零点	零点	3D 棒图、棒图
“ZeroPointValue”	VT_R8	零点值	零点值	3D 棒图、棒图
缩放	VT_I4	缩放因子	缩放因子	画面窗口

3.3.1.7 OCX 的 API 调用

概述

API 可在单独的应用程序中使用。 但是，对集成到多个图形画面的 OCX 进行 API 调用时，可能出现问题（包括可能的死锁）。

如果必须在图形画面中使用 OCX，则必须检查并确保也可通过 IndustrialX 处理此任务。 此选项设计用于完成此目的。

如果确定 API 必须用于此任务，则注意以下说明：

报警记录和变量记录的运行系统 API 函数尤其会受此影响。

一个原因是在运行过程中 API 通过窗口消息从 WinCC 程序请求数据并等待应答。 如果 OCX 在 WinCC 等程序中运行，则其无法处理请求，因为其处于 API 调用过程中。

解决方案：

如果发生此情况，通常将 API 调用置于其它线程中足可解决问题。

但是，仅当问题实际存在时，才可使用此解决方案。 例如，有些其它调用没有在主线程中运行时，它们会遇到问题。

3.3.2 结构

3.3.2.1 LINKINFO

声明

```
typedef struct {
    LINKTYPE    LinkType;
    DWORD       dwCycle;
    TCHAR       szLinkName[256];
    }
    LINKINFO;
```

成员

LinkType

"trigger.h"文件中定义的 enum LinkType 值应该用于 LINKTYPE 类型。

enum LinkType	当前定义的值	链接
BUBRT_LT_NOLINK	0（定义的起始值）	无链接
BUBRT_LT_VARIABLE_DIRECT	1（顺序）	直接变量
BUBRT_LT_VARIABLE_INDIRECT	2（顺序）	间接变量
BUBRT_LT_ACTION	3（顺序）	C 操作
BUBRT_LT_ACTION_WIZARD	4（顺序）	动态对话框
BUBRT_LT_DIRECT_CONNECTION	5（顺序）	直接连接
BUBRT_LT_ACTION_WIZARD_INPROC	6（顺序）	动态对话框

当使用 PDLRTSetLink 时，仅可使用 BUBRT_LT_VARIABLE_DIRECT 和 BUBRT_LT_VARIABLE_INDIRECT。所有的链接类型都可用于 PDLRTGetLink。

说明

要对 trigger.h 中的 enum LinkType 定义进行后续扩展，必须始终在末尾添加新常量，否则上述值将更改。因此，应始终使用枚举的定义而不是值。

dwCycle

更新的周期时间。dwCycle 标识更新周期列表内的顺序。

“改变时”	索引：0
“250ms”	索引：1
:	:
“用户周期 5”	索引：15
“窗口周期”	索引：235
“画面周期”	索引：255

szLinkName

变量连接的名称

所需文件

pdlrtapi.h

trigger.h

API 函数

PDLRTGetLink (页 2253)	查询对象属性和变量之间的链接
PDLRTSetLink (页 2255)	定义对象属性和变量之间的链接

参见

PDLRTGetLink (页 2253)

PDLRTSetLink (页 2255)

3.3.2.2 MULTILINK

声明

```
typedef struct {
    DWORD               wArraySize;
    LPMULTILINKINFO     pLinkArray;
    }
    MULTILINK;
```

成员

wArraySize

链接数（MULTILINKINFO 类型的结构数）。

pLinkArray

指向有关对象属性和变量之间链接的 MULTILINKINFO (页 2215) 类型的第一个 wArraySize 结构的指针。

所需文件

pdlrtapi.h

API 函数

PDLRTSetMultiLink (页 2258)	定义对象属性和变量之间的链接（多个变量）

参见

PDLRTSetMultiLink (页 2258)

MULTILINKINFO (页 2215)

3.3.2.3 MULTILINKINFO

声明

```
typedef struct {
    char*       pszObjectName,
    char*       pszPropertyName,
    LINKTYPE    LinkType;
    DWORD       dwCycle;
    char*       pszLinkName;
    }
    MULTILINKINFO;
```

成员

lpszObjectName

指向对象名称的指针。

lpszPropertyName

指向要链接的对象属性名称的指针。

LinkType

标识连接类型。

LT_VARIABLE_DIRECT	直接连接对象属性和变量
LT_VARIABLE_INDIRECT	间接连接对象属性和变量

dwCycle

更新的周期时间。dwCycle 标识更新周期列表内的顺序。

“改变时”	索引：0
“250ms”	索引：1
:	:
“用户周期 5”	索引：15

pszLinkName

指向连接名称的指针

备注

MULTILINKINFO 结构是 MULTILINK (页 2214) 结构的一部分。

所需文件

pdlrtapi.h

参见

MULTILINK (页 2214)

3.3.2.4 FOCUSINFO

说明

具有运行系统光标的画面对象存储在结构中。

声明

```
typedef struct {
    WCHAR    szPicture[256];
    WCHAR    szObject[256];
    }
    FOCUSINFO;
```

成员

szPicture

画面名称

szObject

对象名称

API 函数

PDLRTGetFocus (页 2236)	查询输入焦点

参见

PDLRTGetFocus (页 2236)

3.3.3 常规函数

3.3.3.1 PDLRTClosePicture

使用

在运行时模式下关闭画面。

声明

```
BOOL PDLRTClosePicture(
    ADRMODE           adrMode,
    LPCSTR            lpszPictureName,
    LPCSTR            lpszWName,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    PCMN_ERROR        pError );
```

参数

adrMode

该参数为将来开发预留，必须预设为 0。

lpszPictureName

指向画面的已组态名称（不包括路径或扩展名）的指针。

该条目区分大小写。

lpszWName

要关闭的窗口对象的名称。

pfn

指向回调函数的指针。调用此函数可通知用户请求成功或失败。

如果传送 pfn = NULL，则调用是同步的。调用应用程序设置为等待，直到 PDLRT 报告失败或成功为止。应优先将此类调用用于相互依赖的调用顺序。

如果后续 API 调用需要此函数完成并且此函数与回调异步使用，则 API 调用应与信号量同步。

可以对各 API 函数使用单独的回调。

pvUser

指向传送给回调函数的应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

画面已关闭。

FALSE

错误。

所需文件

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

相关函数

PDLRT_CALLBACK (页 2231)	运行时函数的回调函数

参见

PDLRT_CALLBACK (页 2231)

函数概览 (页 2180)

3.3.3.2 PDLRTDisableClosePicture

使用

阻止在运行时模式下关闭画面。可以多次调用此函数。仅当对 PDLRTEnableClosePicture 函数的调用达相同次数时，才可再次关闭画面。

声明

```
BOOL PDLRTDisableClosePicture(
    ADRMODE           adrMode,
    LPCSTR            lpszPictureName,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    PCMN_ERROR        pError );
```

参数

adrMode

该参数为将来开发预留，必须预设为 0。

lpszPictureName

指向画面的已组态名称（不包括路径或扩展名）的指针。

该条目区分大小写。

pfn

指向回调函数的指针。调用此函数可通知用户请求成功或失败。

如果传送 pfn = NULL，则调用是同步的。调用应用程序设置为等待，直到 PDLRT 报告失败或成功为止。应优先将此类调用用于相互依赖的调用顺序。

如果后续 API 调用需要此函数完成并且此函数与回调异步使用，则 API 调用应与信号量同步。

可以对各 API 函数使用单独的回调。

pvUser

指向传送给回调函数的应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

pError

指向 CMN_ERROR. 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

函数已成功完成。

FALSE

错误。

所需文件

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

相关函数

PDLRT_CALLBACK (页 2231)	运行时函数的回调函数

参见

PDLRT_CALLBACK (页 2231)

函数概览 (页 2180)

3.3.3.3 PDLRTEnableClosePicture

使用

启用在运行时模式下关闭画面。

但是，仅当对 PDLRTEnableClosePicture 函数的调用达到对 PDLRTDisableClosePicture 函数的调用次数时，才可关闭画面。

记录此前执行的任何 PDLRTClosePicture，然后在最后一个 PDLRTEnableClosePicture 执行完后执行它。

声明

```
BOOL PDLRTEnableClosePicture(
    ADRMODE           adrMode,
    LPCSTR            lpszPictureName,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    PCMN_ERROR        pError );
```

参数

adrMode

该参数为将来开发预留，必须预设为 0。

lpszPictureName

指向画面的已组态名称（不包括路径或扩展名）的指针。

该条目区分大小写。

pfn

指向回调函数的指针。调用此函数可通知用户请求成功或失败。

如果传送 pfn = NULL，则调用是同步的。调用应用程序设置为等待，直到 PDLRT 报告失败或成功为止。应优先将此类调用用于相互依赖的调用顺序。

如果后续 API 调用需要此函数完成并且此函数与回调异步使用，则 API 调用应与信号量同步。

可以对各 API 函数使用单独的回调。

pvUser

指向传送给回调函数的应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

函数已成功完成。

FALSE

错误。

所需文件

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

相关函数

PDLRT_CALLBACK (页 2231)	运行时函数的回调函数

参见

PDLRT_CALLBACK (页 2231)

函数概览 (页 2180)

3.3.3.4 PDLRTGotoPicture

使用

加载指定画面。

声明

```
BOOL PDLRTGotoPicture (
    PDLRTGotoPict   nextPict,
    PCMN_ERROR      pError );
```

参数

nextPict

此方法的函数通过此参数控制。可能为以下值：

PDLRTPictureHome	加载起始画面。
PDLRTPictureNext	加载画面存储器中的下一个画面。
PDLRTPicturePrev	加载画面存储器中的上一个画面。
PDLRTPictureStored	加载画面存储器中存储的画面。
PDLRTStorePicture	将当前画面存储到画面存储器中。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

画面已加载。

FALSE

错误。

错误消息

PDLRT_NO_PIC	未选择画面

所需文件

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

相关函数

PDLRTPictureNavigation (页 2228)	启用与禁用画面浏览

参见

PDLRTPictureNavigation (页 2228)

函数概览 (页 2180)

3.3.3.5 PDLRTInquireFreeArea

使用

确定是否可以在画面中放置给定矩形而不覆盖“受保护区域”。例如，“受保护区域”是前景中的其它画面。

声明

```
BOOL PDLRTInquireFreeArea(
    LPRECT            lpScreenRect,
    DWORD             dwModus,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    PCMN_ERROR        pError );
```

参数

lpScreenRect

指向窗口特定 RECT 类型结构的指针。

dwModus

查询模式：提供多个可根据需要进行或运算的查询选项：

PDLRT_IQ_ONLY	（值：0x01）	根据需要仅返回结果“匹配/不匹配”
PDLRT_IQ_MODY_POSX	（值：0x02）	如果无法放置给定矩形，则可修改矩形的 X 位置以便可以放置矩形。

PDLRT_IQ_MODY_POSY	（值：0x04）	如果无法放置给定矩形，则可修改矩形的 Y 位置以便可以放置矩形。
PDLRT_IQ_MODY_HEIGHT	（值：0x08）	如果无法放置给定矩形，则可修改矩形的高度以便可以放置矩形。
PDLRT_IQ_MODY_WIDTH	（值：0x10）	如果无法放置给定矩形，则可修改矩形的宽度以便可以放置矩形。

pfn

指向回调函数的指针。调用此函数可通知用户请求成功或失败。

如果传送 pfn = NULL，则调用是同步的。调用应用程序设置为等待，直到 PDLRT 报告失败或成功为止。应优先将此类调用用于相互依赖的调用顺序。

如果后续 API 调用需要此函数完成并且此函数与回调异步使用，则 API 调用应与信号量同步。

可以对各 API 函数使用单独的回调。

pvUser

指向传送给回调函数的应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

窗口区域空闲。

FALSE

错误。

所需文件

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

相关函数

PDLRT_CALLBACK (页 2231)	运行时函数的回调函数

参见

PDLRT_CALLBACK (页 2231)

函数概览 (页 2180)

3.3.3.6 PDLRTOpenPicture

使用

此函数用于在运行时执行起始画面更改。

声明

```
BOOL PDLRTOpenPicture(
    ADRMODE           adrMode,
    LPCSTR            lpszPictureName,
    LPCSTR            lpszWName,
    LPCSTR            lpszPictureFileName,
    DWORD             dwWinStyle,
    LONG              lxPos,
    LONG              lyPos,
    LONG              lWitdh,
    LONG              lHeight,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    PCMN_ERROR        pError );
```

参数

adrMode

指向画面的已组态名称（不包括路径或扩展名）的指针。

该条目区分大小写。

lpszWName

目前必须给此参数提供画面名称。

lpszPictureFileName

目前必须给此参数提供画面名称。

dwWinStyle

该参数为将来开发预留，必须预设为 0。

lxPos

该参数为将来开发预留，必须预设为 0。

lyPos

该参数为将来开发预留，必须预设为 0。

lWidth

该参数为将来开发预留，必须预设为 0。

lHeigth

该参数为将来开发预留，必须预设为 0。

pfn

指向回调函数的指针。调用此函数可通知用户请求成功或失败。

如果传送 pfn = NULL，则调用是同步的。调用应用程序设置为等待，直到 PDLRT 报告失败或成功为止。应优先将此类调用用于相互依赖的调用顺序。

如果后续 API 调用需要此函数完成并且此函数与回调异步使用，则 API 调用应与信号量同步。

可以对各 API 函数使用单独的回调。

pvUser

指向传送给回调函数的应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

起始画面更改已执行。

FALSE

错误。

错误消息

PDLRT_PICTURE_ALREADY_OPEN	画面已打开
PDLRT_NO_PIC	未选择画面
PDLRT_BAD_OLE_CONVERSION	通过 OLE 自动转换时出错

所需文件

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

相关函数

PDLRT_CALLBACK (页 2231)	运行时函数的回调函数

参见

PDLRT_CALLBACK (页 2231)

函数概览 (页 2180)

3.3.3.7 PDLRTPictureNavigation

使用

打开和关闭画面浏览。关闭时，无法再随着画面更改将画面添加到堆栈；旧状态被冻结。

声明

```
BOOL PDLRTPictureNavigation (
    PDLRT_PNFLAGS    flags,
    PCMN_ERROR       pError );
```

参数

flags

此函数通过此参数控制。可能为以下值：

PDLRT_PNF_ENABLE	打开画面浏览。
PDLRT_PNF_DISABLE	关闭画面浏览。
PDLRT_PNF_ENABLE_LOAD_LAST_PICTURE	打开画面浏览并加载画面存储器中的最后一个画面。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

函数已成功完成。

FALSE

错误。

原因：画面浏览已打开或关闭。

所需文件

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

相关函数

PDLRTGotoPicture (页 2222)	加载指定画面

参见

PDLRTGotoPicture (页 2222)

函数概览 (页 2180)

3.3.3.8 PDLRTShowApp

使用

此函数用于将可能位于背景中的运行系统窗口提到桌面前景。

声明

```
BOOL PDLRTShowApp (
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    PCMN_ERROR        pError );
```

参数

pfn

指向回调函数的指针。调用此函数可通知用户请求成功或失败。

如果传送 pfn = NULL，则调用是同步的。调用应用程序设置为等待，直到 PDLRT 报告失败或成功为止。应优先将此类调用用于相互依赖的调用顺序。

如果后续 API 调用需要此函数完成并且此函数与回调异步使用，则 API 调用应与信号量同步。

可以对各 API 函数使用单独的回调。

pvUser

指向传送给回调函数的应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

运行系统窗口位于前景中。

FALSE

错误。

所需文件

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

相关函数

PDLRT_CALLBACK (页 2231)	回调函数

参见

PDLRT_CALLBACK (页 2231)

函数概览 (页 2180)

3.3.3.9 PDLRT_CALLBACK

说明

如果要异步通知应用程序有关运行时 API 函数的执行情况，则必须提供 PDLRT_CALLBACK 类型的回调函数。

所有 PDLRT 函数将此函数用作回调函数。

声明

```
void ( * PDLRT_CALLBACK ) (
    LPVOID        pvUser,
    PCMN_ERROR    pError );
```

参数

pvUser

指向应用程序特定数据的指针。此指针用作 API 函数的参数并再次原样返回到这里。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

此回调没有返回值。

备注

如果调用 API 函数而没有回调，例如，pfn = NULL、，则调用是同步的。调用应用程序设置为等待，直到 PDLRT 报告失败或成功为止。应优先将此类调用用于相互依赖的调用顺序。

如果后续 API 调用需要此函数完成并且此函数与回调异步使用，则 API 调用应与信号量同步。若不同步，也会存在风险：在调用过程已结束或离开有效性范围时，异步函数仍将继续通过数据指针写入不再有效的数据，这将导致异常处理。

可以对各 API 函数使用单独的回调。

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

在极少数情况下，在函数调用返回之前可能已传送回调。

所需文件

pdlrtapi.h

参见

PDLRTClosePicture (页 2217)

PDLRTDisableClosePicture (页 2219)

PDLRTEnableClosePicture (页 2220)

PDLRTInquireFreeArea (页 2224)

PDLRTOpenPicture (页 2226)

PDLRTShowApp (页 2230)

PDLRTSetLink (页 2255)

PDLRTSetMultiLink (页 2258)

PDLRTGetLink (页 2253)

PDLRTSetPropEx (页 2248)

PDLRTGetPropEx (页 2245)

PDLRTGetDefPropEx (页 2242)

PDLRTSetFocus (页 2240)

PDLRTSetCursorKeys (页 2238)

PDLRTGetFocus (页 2236)

PDLRTGetCursorKeys (页 2233)

函数概览 (页 2180)

3.3.4 用于调整运行系统光标的函数

3.3.4.1 PDLRTGetCursorKeys

使用

此函数查询使用 PDLRTSetCursorKeys 设置的光标控制键。

声明

```
BOOL PDLRTGetCursorKeys (
    long*             pKeyUp,
    long*             pKeyDown,
    long*             pKeyLeft,
    long*             pKeyRight,
    long*             pKeyState,
    long*             pTabMode,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    PCMN_ERROR        pError );
```

参数

KeyUp

此指针提供用于向下移动光标的按键的虚拟键代码 (VK_...)。

pKeyDown

此指针提供用于向上移动光标的按键的虚拟键代码 (VK_...)。

pKeyLeft

此指针提供用于向左移动光标的按键的虚拟键代码 (VK_...)。

pKeyRight

此指针提供用于向右移动光标的按键的虚拟键代码 (VK_...)。

pKeyState

此指针提供键盘状态：

HOTKEYF_SHIFT	（值：0x01）	“SHIFT”键按下
HOTKEYF_CONTROL	（值：0x02）	“CTRL”键按下
HOTKEYF_ALT	（值：0x04）	“ALT”键按下

可以根据需要对这些值进行或运算。

pTabMode

此指针提供 TabMode，此对象在下一个光标控制键激活时获取焦点：

O：	xxScreenxx
1：	xxScreenxx
10：	xxScreenxx
	xxScreenxx

pfn

指向回调函数的指针。调用此函数可通知用户请求成功或失败。

如果传送 pfn = NULL，则调用是同步的。调用应用程序设置为等待，直到 PDLRT 报告失败或成功为止。应优先将此类调用用于相互依赖的调用顺序。

如果后续 API 调用需要此函数完成并且此函数与回调异步使用，则 API 调用应与信号量同步。

可以对各 API 函数使用单独的回调。

pvUser

指向传送给回调函数的应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已确定光标控制键。

FALSE

错误。

所需文件

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

相关函数

PDLRTSetCursorKeys (页 2238)	设置光标控制键
PDLRT_CALLBACK (页 2231)	回调函数

参见

PDLRTSetCursorKeys (页 2238)

PDLRT_CALLBACK (页 2231)

函数概览 (页 2180)

3.3.4.2 PDLRTGetFocus

使用

此函数用于返回画面和具有输入焦点的对象的名称。

声明

```
BOOL PDLRTGetFocus(
    ADRMODE           adrMode,
    LPFOCUSINFO       pFocusInfo,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    PCMN_ERROR        pError );
```

参数

adrMode

adrMode 参数用于设置要编辑的画面的寻址模式。

PDLRT_AM_DEFAULT	o	画面和对象的寻址是相对的
PDLRT_AM_PICTABS	1	画面的寻址是绝对的
PDLRT_AM_OBJABS	2	对象的寻址是绝对的

可以对 PDLRT_AM_PICTABS 和 PDLRT_AM_OBJABS 值进行或运算。

pFocusInfo

指向其中存储了结果的 FOCUSINFO (页 2216) 结构的指针。

pfn

指向回调函数的指针。调用此函数可通知用户请求成功或失败。

如果传送 pfn = NULL，则调用是同步的。调用应用程序设置为等待，直到 PDLRT 报告失败或成功为止。应优先将此类调用用于相互依赖的调用顺序。

如果后续 API 调用需要此函数完成并且此函数与回调异步使用，则 API 调用应与信号量同步。

可以对各 API 函数使用单独的回调。

pvUser

指向传送给回调函数的应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已确定输入焦点。

FALSE

错误。

所需文件

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

相关函数

PDLRTSetFocus (页 2240)	设置输入焦点
PDLRT_CALLBACK (页 2231)	回调函数

参见

FOCUSINFO (页 2216)

PDLRTSetFocus (页 2240)

PDLRT_CALLBACK (页 2231)

函数概览 (页 2180)

3.3.4.3 PDLRTSetCursorKeys

使用

此函数用于设置光标控制键。

声明

```
BOOL PDLRTSetCursorKeys (
    long              KeyUp,
    long              KeyDown,
    long              KeyLeft,
    long              KeyRight,
    long              KeyState,
    long              TabMode,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    PCMN_ERROR        pError );
```

参数

KeyUp

用于向下移动光标的按键的虚拟键代码 (VK_...) 。

KeyDown

用于向上移动光标的按键的虚拟键代码 (VK_...) 。

KeyLeft

用于向左移动光标的按键的虚拟键代码 (VK_...) 。

KeyRight

用于向右移动光标的按键的虚拟键代码 (VK_...) 。

KeyState

KeyState 提供键盘状态：

HOTKEYF_SHIFT	（值：0x01）	“SHIFT”键按下
HOTKEYF_CONTROL	（值：0x02）	“CTRL”键按下
HOTKEYF_ALT	（值：0x04）	“ALT”键按下

可以根据需要对这些值进行或运算。

TabMode

TabMode 指定在下一个光标控制键激活时获取焦点的对象：

0：	xxScreenxx
1：	xxScreenxx
10：	xxScreenxx
	xxScreenxx

pfn

指向回调函数的指针。调用此函数可通知用户请求成功或失败。

如果传送 pfn = NULL，则调用是同步的。调用应用程序设置为等待，直到 PDLRT 报告失败或成功为止。应优先将此类调用用于相互依赖的调用顺序。

如果后续 API 调用需要此函数完成并且此函数与回调异步使用，则 API 调用应与信号量同步。

可以对各 API 函数使用单独的回调。

pvUser

指向传送给回调函数的应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已定义光标控制键。

FALSE

错误。

所需文件

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

相关函数

PDLRTGetCursorKeys (页 2233)	查询光标控制键
PDLRT_CALLBACK (页 2231)	回调函数

参见

PDLRTGetCursorKeys (页 2233)

PDLRT_CALLBACK (页 2231)

函数概览 (页 2180)

3.3.4.4 PDLRTSetFocus

使用

此函数用于设置输入焦点。此对象通过 lpszPictureName 和 lpszObjectName 确定。

声明

```
BOOL PDLRTSetFocus (
    ADRMODE           adrMode,
    LPCSTR            lpszPictureName,
    LPCSTR            lpszObjectName,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    PCMN_ERROR        pError );
```

参数

adrMode

adrMode 参数用于设置要编辑的画面的寻址模式。

PDLRT_AM_DEFAULT	o	画面和对象的寻址是相对的
PDLRT_AM_PICTABS	1	画面的寻址是绝对的
PDLRT_AM_OBJABS	2	对象的寻址是绝对的

可以对 PDLRT_AM_PICTABS 和 PDLRT_AM_OBJABS 值进行或运算。

lpszPictureName

根据 adrMode 定义的寻址模式指向画面的已组态名称（无扩展名）的指针。该条目区分大小写。

lpszObjectName

根据 adrMode 定义的寻址模式指向已组态对象名称的指针。

pfn

指向回调函数的指针。调用此函数可通知用户请求成功或失败。

如果传送 pfn = NULL，则调用是同步的。调用应用程序设置为等待，直到 PDLRT 报告失败或成功为止。应优先将此类调用用于相互依赖的调用顺序。

如果后续 API 调用需要此函数完成并且此函数与回调异步使用，则 API 调用应与信号量同步。

可以对各 API 函数使用单独的回调。

pvUser

指向传送给回调函数的应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已设置输入焦点

FALSE

错误

错误消息

PDLRT_NO_PIC	未选择画面
PDLRT_NO_OBJ	找不到对象

所需文件

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

相关函数

PDLRTGetFocus (页 2236)	查询输入焦点
PDLRT_CALLBACK (页 2231)	回调函数

参见

PDLRTGetFocus (页 2236)

PDLRT_CALLBACK (页 2231)

函数概览 (页 2180)

3.3.5 用于处理对象属性的函数

3.3.5.1 PDLRTGetDefPropEx

使用

请求默认属性值。如果请求的数据类型与属性的数据类型不匹配，则在向 pvProp 用户缓冲区传送期间根据请求的格式更改此值（如果可能）。

声明

```
BOOL PDLRTGetDefPropEx(
    ADRMODE           adrMode,
    LPCSTR            lpszPictureName,
    LPCSTR            lpszObjectName,
    LPCSTR            lpszPropName,
    VARTYPE           vt,
    LPVOID            pvProp,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    DWORD             dwFlags,
    LPVOID            pData,
    PCMN_ERROR        pError );
```

参数

adrMode

adrMode 参数用于设置要编辑的画面的寻址模式。

PDLRT_AM_DEFAULT	0	画面和对象的寻址是相对的
PDLRT_AM_PICTABS	0	画面的寻址是绝对的
PDLRT_AM_OBJABS	2	对象的寻址是绝对的

可以对 PDLRT_AM_PICTABS 和 PDLRT_AM_OBJABS 值进行或运算。

lpszPictureName

根据 adrMode 定义的寻址模式指向画面的已组态名称的指针。该条目区分大小写。

lpszObjectName

指向已组态对象名称的指针，根据通过 adrMode 定义的寻址模式。

lpszPropName

指向对象属性的已组态名称的指针。

vt

pvProp 下传送的值的数据类型。允许的类型在编译器的 include "oaidl.h" 和 "wtypes.h" 文件中定义。可以将 WinCC include 文件 APLIB\Defines.h 用于脚本。

pvProp

指向存储属性值的变量的指针。通过 vt 确定值的数据类型：

vt	pvProp
VT_BSTR	BSTR*
VT_LPSTR	LPSTR*
VT_LPWSTR	LPWSTR*
VT_UI4	LONG*
VT_xxxx	pvProp 必须指向缓冲区

对于具有缓冲功能的类型（例如，BSTR），函数会分配缓冲区，但调用应用程序必须再次释放该缓冲区。

pfn

指向回调函数的指针。调用此函数可通知用户请求成功或失败。

如果传送 pfn = NULL，则调用是同步的。调用应用程序设置为等待，直到 PDLRT 报告失败或成功为止。应优先将此类调用用于相互依赖的调用顺序。

如果后续 API 调用需要此函数完成并且此函数与回调异步使用，则 API 调用应与信号量同步。

可以对各 API 函数使用单独的回调。

pvUser

指向传送给回调函数的应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

dwFlags

该参数为将来开发预留，必须预设为 0。

pData

该参数为将来开发预留，必须预设为 NULL。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已确定对象属性的默认值。

FALSE

错误。

备注

要释放分配的内存区域，使用 SysFreeString(...) 的 VT_BSTR Windows API 函数以及 VT_LPWSTR 和 LocalFree(...) 的 VT_LPSTR Windows API 函数。

错误消息

PDLRT_NO_PROP	找不到属性
PDLRT_BAD_OLE_CONVERSION	通过 OLE 自动转换时出错

所需文件

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

相关函数

PDLRT_CALLBACK (页 2231)	回调函数

参见

PDLRT_CALLBACK (页 2231)

函数概览 (页 2180)

3.3.5.2 PDLRTGetPropEx

使用

请求当前属性值。通过 lpszPictureName、lpszObjectName 和 lpszPropName 确定源。

如果请求的数据类型与属性的数据类型不匹配，则在向 pvProp 用户缓冲区传送期间根据请求的格式更改此值（如果可能）。

声明

```
BOOL PDLRTGetPropEx(
    ADRMODE           adrMode,
    LPCSTR            lpszPictureName,
    LPCSTR            lpszObjectName,
    LPCSTR            lpszPropName,
    VARTYPE           vt,
    LPVOID            pvProp,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    DWORD             dwFlags,
    LPVOID            pData,
    PCMN_ERROR        pError );
```

参数

adrMode

adrMode 参数用于设置要编辑的画面的寻址模式。

PDLRT_AM_DEFAULT	0	画面和对象的寻址是相对的
PDLRT_AM_PICTABS	0	画面的寻址是绝对的
PDLRT_AM_OBJABS	2	对象的寻址是绝对的

可以对 PDLRT_AM_PICTABS 和 PDLRT_AM_OBJABS 值进行或运算。

lpszPictureName

根据 adrMode 定义的寻址模式指向画面的已组态名称的指针。该条目区分大小写。

lpszObjectName

根据 adrMode 定义的寻址模式指向对象的已组态名称的指针。

lpszPropName

指向对象属性的已组态名称的指针。

vt

pvProp 下传送的值的数据类型。允许的类型在编译器的 include "oaidl.h" 和 "wtypes.h" 文件中定义。可以将 WinCC include 文件 APLIB\Defines.h 用于脚本。

pvProp

指向存储属性值的变量的指针。通过 vt 确定值的数据类型：

vt	pvProp
VT_BSTR	BSTR*
VT_LPSTR	LPSTR*
VT_LPWSTR	LPWSTR*
VT_UI4	LONG*
VT_xxxx	pvProp 必须指向缓冲区

对于具有缓冲功能的类型（例如，BSTR），函数会分配缓冲区，但调用应用程序必须再次释放该缓冲区。

pfn

指向回调函数的指针。调用此函数可通知用户请求成功或失败。

如果传送 pfn = NULL，则调用是同步的。调用应用程序设置为等待，直到 PDLRT 报告失败或成功为止。应优先将此类调用用于相互依赖的调用顺序。

如果后续 API 调用需要此函数完成并且此函数与回调异步使用，则 API 调用应与信号量同步。

可以对各 API 函数使用单独的回调。

pvUser

指向传送给回调函数的应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

dwFlags

该参数为将来开发预留，必须预设为 0。

pData

该参数为将来开发预留，必须预设为 NULL。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已确定对象属性。

FALSE

错误。

备注

要释放分配的存储区域，请对 VT_BSTR 使用 Windows API 函数 SysFreeString(...)，对 VT_LPSTR、VT_LPWSTR 使用 Windows API 函数 LocalFree(...)。

不可使用 VT_DISPATCH 或其它参考；只能使用标准类型以及最多使用一个 VT_VARIANT（对于简单类型的数组属性）。

错误消息

PDLRT_NO_PIC	未选择画面
PDLRT_BAD_OLE_CONVERSION	通过 OLE 自动转换时出错
PDLRT_NO_PROP	找不到属性

所需文件

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

相关函数

PDLRTSetPropEx (页 2248)	设置属性
PDLRT_CALLBACK (页 2231)	回调函数

参见

PDLRT_CALLBACK (页 2231)

PDLRTSetPropEx (页 2248)

函数概览 (页 2180)

3.3.5.3 PDLRTSetPropEx

使用

设置属性。 通过 lpszPictureName、lpszObjectName 和 lpszPropName 确定目标。 如果传送的数据类型与属性的数据类型不匹配，则在向属性进行传送期间更改此值（如果可能）。

只能设置那些也能动态化的属性。

声明

```
BOOL PDLRTSetPropEx(
    ADRMODE           adrMode,
    LPCSTR            lpszPictureName,
    LPCSTR            lpszObjectName,
    LPCSTR            lpszPropName,
    VARTYPE           vt,
    LPVOID            pvProp,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    DWORD             dwFlags,
    LPVOID            pData,
    PCMN_ERROR        pError );
```

参数

adrMode

adrMode 参数用于设置要编辑的画面的寻址模式。

PDLRT_AM_DEFAULT	0	画面和对象的寻址是相对的
PDLRT_AM_PICTABS	0	画面的寻址是绝对的
PDLRT_AM_OBJABS	2	对象的寻址是绝对的

可以对 PDLRT_AM_PICTABS 和 PDLRT_AM_OBJABS 值进行或运算。

lpszPictureName

根据 adrMode 定义的寻址模式指向画面的已组态名称的指针。 该条目区分大小写。

lpszObjectName

根据 adrMode 定义的寻址模式指向已组态对象名称的指针。

lpszPropName

指向对象属性的已组态名称的指针。

vt

pvProp 下传送的值的数据类型。 允许的类型在编译器的 include "oaidl.h" 和 "wtypes.h" 文件中定义。 可以将 WinCC include 文件 APLIB\Defines.h 用于脚本。

pvProp

指向存储属性值的缓冲区的指针。 通过 vt 确定值的数据类型：

vt	pvProp
VT_BSTR	BSTR*
VT_LPSTR	LPSTR*
VT_LPWSTR	LPWSTR*
VT_UI4	LONG*
VT_xxxx	pvProp 必须指向缓冲区

pfn

指向回调函数的指针。 调用此函数可通知用户请求成功或失败。

如果传送 pfn = NULL，则调用是同步的。 调用应用程序设置为等待，直到 PDLRT 报告失败或成功为止。 应优先将此类调用用于相互依赖的调用顺序。

如果后续 API 调用需要此函数完成并且此函数与回调异步使用，则 API 调用应与信号量同步。

可以对各 API 函数使用单独的回调。

pvUser

指向传送给回调函数的应用程序特定数据的指针。 函数不会评估该指针，但在回调函数中该指针会重新可用。

dwFlags

该参数为将来开发预留，必须预设为 0。

pData

该参数为将来开发预留，必须预设为 0。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。 出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已设置对象属性。

FALSE

错误。

备注

不可使用 VT_DISPATCH 或其它参考；只能使用标准类型以及最多使用一个 VT_VARIANT（对于简单类型的数组属性）。

错误消息

PDLRT_NO_PIC	未选择画面
PDLRT_PICTURE_NOT_LOADED	无法加载画面文件
PDLRT_FAILURE_PARAM	错误参数
PDLRT_NO_OBJ	找不到对象
PDLRT_NO_PROP	找不到属性
PDLRT_BAD_OLE_CONVERSION	通过 OLE 自动转换时出错

所需文件

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

相关函数

PDLRTGetPropEx (页 2245)	查询属性
PDLRT_CALLBACK (页 2231)	回调函数

示例

```
#pragma code("pdlrtapi.dll")

#include "apdefap.h"

#define CP_ACP 0
#define MB_PRECOMPOSED   1

void ButtonRT(char* text)
{
    DM_DIRECTORY_INFO dirInfo;
    CMN_ERROR error;
    BOOL bRet;
    char szProjectFile[222] = "";
    WCHAR  wstring[99];
    BSTR bstrval;
    WORD *pw = NULL;
    int i;

    printf("Function  ButtonRT() with text '%s'\r\n", text);

    bRet = DMGetRuntimeProject
(&szProjectFile[0],sizeof(szProjectFile),&error);
    printf("DMGetRuntimeProject: %s\r\n", szProjectFile);

    bRet = PDLRTOpenPicture(0,"Start.PDL",NULL,NULL,0,0,0,0,0,NULL,NULL,
&error);
    printf("PDLRTOpenPicture: %s,%ld, %s\r\n", bRet?"TRUE":"FALSE",
error.dwError1,error.szErrorText);

    bRet = PDLRTSetPropEx(0,"Start.PDL",  "Button4", "Text",VT_LPSTR,
&text,NULL,NULL,0, NULL, &error);
    printf("PDLRTSetPropEx: %s,%ld, %s\r\n", bRet?"TRUE":"FALSE",
error.dwError1,error.szErrorText);

    SysFreeString(  bstrval  );
    printf("SysFreeString done\r\n");
}
```

参见

PDLRT_CALLBACK (页 2231)

PDLRTGetPropEx (页 2245)

函数概览 (页 2180)

3.3.6 处理动态性的函数

3.3.6.1 PDLRTGetLink

使用

查询属性的链接。如果属性具有间接变量连接，则将返回它。如果间接变量连接需要当前激活的直接连接，则可通过在 LINKINFO 结构中指定
LinkType = BUBRT_LT_VARIABLE_DIRECT 进行请求。

声明

```
BOOL PDLRTGetLink (
    ADRMODE           adrMode,
    LPCSTR            lpszPictureName,
    LPCSTR            lpszObjectName,
    LPCSTR            lpszPropName,
    LPLINKINFO        pLink,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    PCMN_ERROR        pError );
```

参数

adrMode

adrMode 参数用于设置要编辑的画面的寻址模式。

PDLRT_AM_DEFAULT	o	画面和对象的寻址是相对的
PDLRT_AM_PICTABS	1	画面的寻址是绝对的
PDLRT_AM_OBJABS	2	对象的寻址是绝对的

可以对 PDLRT_AM_PICTABS 和 PDLRT_AM_OBJABS 值进行或运算。

lpszPictureName

根据 adrMode 定义的寻址模式指向画面的已组态名称（无扩展名）的指针。该条目区分大小写。

lpszObjectName

根据 adrMode 定义的寻址模式指向已组态对象名称的指针。

lpszPropName

指向对象属性名称的指针。

pLink

指向其中存储了链接信息的 LINKINFO (页 2212) 结构的指针。

pfn

指向回调函数的指针。调用此函数可通知用户请求成功或失败。

如果传送 pfn = NULL，则调用是同步的。调用应用程序设置为等待，直到 PDLRT 报告失败或成功为止。应优先将此类调用用于相互依赖的调用顺序。

如果后续 API 调用需要此函数完成并且此函数与回调异步使用，则 API 调用应与信号量同步。

可以对各 API 函数使用单独的回调。

pvUser

指向传送给回调函数的应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已确定链接。

FALSE

错误。

错误消息

PDLRT_NO_LINK	属性不具有动态性

所需文件

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

相关函数

PDLRTSetLink (页 2255)	定义对象属性和变量之间的链接
PDLRT_CALLBACK (页 2231)	回调函数

参见

PDLRTSetLink (页 2255)

PDLRT_CALLBACK (页 2231)

LINKINFO (页 2212)

函数概览 (页 2180)

3.3.6.2 PDLRTSetLink

使用

设置属性和变量之间的链接。

声明

```
BOOL PDLRTSetLink (
    ADRMODE           adrMode,
    LPCSTR            lpszPictureName,
    LPCSTR            lpszObjectName,
    LPCSTR            lpszPropName,
    LPLINKINFO        pLink,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    PCMN_ERROR        pError );
```

参数

adrMode

adrMode 参数用于设置要编辑的画面的寻址模式。

PDLRT_AM_DEFAULT	o	画面和对象的寻址是相对的
PDLRT_AM_PICTABS	1	画面的寻址是绝对的
PDLRT_AM_OBJABS	2	对象的寻址是绝对的

可以对 PDLRT_AM_PICTABS 和 PDLRT_AM_OBJABS 值进行或运算。

lpszPictureName

根据 adrMode 定义的寻址模式指向画面的已组态名称（无扩展名）的指针。该条目区分大小写。

lpszObjectName

根据 adrMode 定义的寻址模式指向已组态对象名称的指针。

lpszPropName

指向对象属性名称的指针。

pLink

指向其中存储了链接信息的 LINKINFO (页 2212) 结构的指针。

因为这里仅允许与变量的链接，所以只能在 LinkType 中使用 BUBRT_LT_VARIABLE_DIRECT 和 BUBRT_LT_VARIABLE_INDIRECT 。

pfn

指向回调函数的指针。调用此函数可通知用户请求成功或失败。

如果传送 pfn = NULL ，则调用是同步的。调用应用程序设置为等待，直到 PDLRT 报告失败或成功为止。应优先将此类调用用于相互依赖的调用顺序。

如果后续 API 调用需要此函数完成并且此函数与回调异步使用，则 API 调用应与信号量同步。

可以对各 API 函数使用单独的回调。

pvUser

指向传送给回调函数的应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已设置链接

FALSE

错误

错误消息

PDLRT_FAILURE_PARAM	错误参数
PDLRT_NO_PIC	未选择画面
PDLRT_NO_OBJ	找不到对象
PDLRT_LINK_NOT_SET	无法链接属性

所需文件

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

相关函数

PDLRTGetLink (页 2253)	确定对象属性和变量之间的链接
PDLRTSetMultiLink (页 2258)	定义对象属性和变量之间的链接（多个变量）
PDLRT_CALLBACK (页 2231)	回调函数

参见

PDLRTGetLink (页 2253)

PDLRTSetMultiLink (页 2258)

PDLRT_CALLBACK (页 2231)

LINKINFO (页 2212)

函数概览 (页 2180)

3.3.6.3 PDLRTSetMultiLink

使用

设置属性和变量之间的多个链接。

声明

```
BOOL PDLRTSetMultiLink (
    ADRMODE           adrMode,
    LPCSTR            lpszPictureName,
    LPMULTILINK       pMultiLink,
    PDLRT_CALLBACK    pfn,
    LPVOID            pvUser,
    PCMN_ERROR        pError );
```

参数

adrMode

adrMode 参数用于设置要编辑的画面的寻址模式。

PDLRT_AM_DEFAULT	o	画面和对象的寻址是相对的
PDLRT_AM_PICTABS	1	画面的寻址是绝对的
PDLRT_AM_OBJABS	2	对象的寻址是绝对的

可以对 PDLRT_AM_PICTABS 和 PDLRT_AM_OBJABS 值进行或运算。

lpszPictureName

根据 adrMode 定义的寻址模式指向画面的已组态名称（无扩展名）的指针。该条目区分大小写。

pMultiLink

指向其中存储了具有链接信息的字段的 MULTILINK (页 2214) 结构的指针。

pfn

指向回调函数的指针。调用此函数可通知用户请求成功或失败。

如果传送 pfn = NULL，则调用是同步的。调用应用程序设置为等待，直到 PDLRT 报告失败或成功为止。应优先将此类调用用于相互依赖的调用顺序。

如果后续 API 调用需要此函数完成并且此函数与回调异步使用，则 API 调用应与信号量同步。

可以对各 API 函数使用单独的回调。

pvUser

指向传送给回调函数的应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

返回值

TRUE

仅当所有链接都已成功设置，才会返回 TRUE 返回值。即使只有一个链接无法设置，也会返回 FALSE。

FALSE

错误。

错误消息

PDLRT_FAILURE_PARAM	错误参数
PDLRT_NO_PIC	未选择画面
PDLRT_NO_OBJ	找不到对象
PDLRT_LINK_NOT_SET	无法链接属性

所需文件

pdlrtapi.h

pdlrt_s.lib

pdlrtapi.dll

相关函数

PDLRTSetLink (页 2255)	定义对象属性和变量之间的链接
PDLRT_CALLBACK (页 2231)	回调函数

参见

PDLRTSetLink (页 2255)

PDLRT_CALLBACK (页 2231)

MULTILINK (页 2214)

函数概览 (页 2180)

3.4 脚本函数

3.4.1 基本知识

3.4.1.1 函数概览

概述

AP_RT_PROC (页 2278)	运行时函数的通知
APActive (页 2291)	激活动作
APCompile (页 2281)	编译源代码
APCompileEx (页 2283)	编译源代码
APConnect (页 2272)	注册应用程序
APDisconnect (页 2274)	注销应用程序
APEndAct (页 2293)	注册应用程序控制中的动作
APFreeResultStruct (页 2295)	释放动作结果内存
APInactive (页 2296)	禁用动作
APSetLanguage (页 2277)	设置错误文本的语言
APStart (页 2298)	启动动作
APTransact (页 2301)	注册要处理的动作
GSCGenCompile (页 2286)	编译动作
GSCGenCompileUserFunctions (页 2288)	编译用户函数
GSCGenGetActionStream (页 2289)	确定动作流

3.4.1.2 结构概览

概述

AP_ACT_KEY (页 2266)	动作标识
AP_ACT_RESULT_STRUCT (页 2268)	动作结果
CREATE_USER_HEADER_FILE (页 2269)	创建用户函数的头文件
GENERATE_COMPILE (页 2270)	生成动作
GET_ACTION_STREAM (页 2271)	确定动作流

3.4.1.3 错误消息

概述

CMN_ERROR 错误结构中的 API 函数可返回以下错误消息：

一般错误消息

AP_ALREADY_CONNECTED	2	应用程序已经与动作控制连接
AP_NO_CONNECTION	3	应用无法建立与动作控制的连接。
AP_ERROR_IPC_SEND	4	与动作控制进行通信时出错。无法发送作业。
AP_FAILURE_UNKNOWN	5	未详细描述错误。
AP_FAILURE_PARAM	6	参数分配不正确
AP_NO_ACT_PROGRAM	7	无法启动动作控制
AP_TIMEOUT	8	超时，目前尚未实现。
AP_ACT_QUIT	9	动作控制已结束。
AP_INSTALL_SERV_ERROR	10	无法安装服务通道
AP_ENDACT_UNKNOWN_ORDER	11	将未知作业号用于 EndAct
AP_ACTION_FAILED	12	在无错误情况下无法执行动作。返回结果无效。
AP_FAILURE_IN_SERVER	13	服务器发出错误信号。
AP_TO_MANY_CLIENTS	14	达到最大连接数。无法建立新连接。
AP_TRANSACTID_UNKNOWN	15	无效事务 ID。如果尝试结束未事先注册的事务，则调用 AP_EndAct 时将发生错误。

AP_NO_MEMORY	16	不再有足够的内存可供此操作使用
AP_TRANSACT_ERROR	17	事务中发生错误；AP_ACT_KEY 的 dwerror 存在动作相关错误
AP_RESULT_TRANS_ERROR	18	结果存在错误。
AP_RESULT_START_ERROR	19	结果存在错误。
AP_NO_UPDATE_WRONG_FORMAT	50	更新选项不适用于该数据格式。动作不包含解释器代码。
AP_ERR_WRONG_FORMAT	202	动作的数据格式不正确。
AP_NO_VALID_FUNCTION_VALUE	203	函数返回值无法转换为 VARIANT 数据类型。

错误

执行动作时可能发生的错误

AP_CISS_ERR_EXIT_OVERFLOW	1001	脚本解释执行中堆栈溢出。取消进一步执行该动作。
AP_CISS_ERR_EXIT_DIVIDE0	1002	执行动作时发生用 0 作除数。该动作被取消。
AP_CISS_ERR_EXIT_UNRESOLVED	1003	执行动作时，引用不存在的符号。
AP_CISS_ERR_EXIT_GPF	1004	执行动作时，访问未定义的内存。
AP_CISS_ERR_EXIT_BREAKPOINT	1005	动作解释器因断点停止。
AP_CISS_ERR_EXIT_STEP	1006	在下一处理步骤中恢复了动作解释器。

报警

动作解释器的报警

AP_CISS_ERR_CREATE_PCH_FROM_PCH	8001	无法从预编译头文件生成预编译头文件。
AP_CISS_ERR_MODULE_IN_USE	8002	无法访问动作。模块当前正在使用。
AP_CISS_ERR_INVALID_PROGRAM	8003	程序无效。
AP_CISS_ERR_INVALID_MODULE	8004	动作无效。
AP_CISS_ERR_CANNOT_CREATE_FILE	8005	动作控制无法创建文件。
AP_CISS_ERR_CANNOT_NO_MEMORY	8006	动作解释器不再具有足够内存。
AP_CISS_ERR_INVALID_FILE_FORMAT	8007	文件格式对动作控制无效。

AP_CISS_ERR_CANNOT_OPEN_FILE	8008	动作控制无法打开文件。
AP_CISS_ERR_PROGRAM_IS_LOCKED	8009	程序目前被动作控制锁定。
AP_CISS_ERR_MODULE_ALREADY_INSERTED	8010	已将动作提供给动作控制进行处理。
AP_CISS_ERR_CONFLICT_WITH_OTHER	8011	在动作执行过程中，与另一动作发生冲突。
AP_CISS_ERR_MODULE_NOT_FOUND	8013	动作控制无法找到动作。
AP_CISS_ERR_FUNCTION_NOT_FOUND	8014	动作控制无法找到函数。
AP_CISS_ERR_INVALID_LINE	8015	指定的行信息无效。
AP_CISS_ERR_INVALID_SCOPE	8016	指定的符号超出有效范围。
AP_CISS_ERR_BUFFER_TOO_SMALL	8017	传送的内存对于动作解释器来说过小。
AP_CISS_ERR_INVALID_TYPE	8018	动作解释器不识别指定的类型。
AP_CISS_ERR_SYMBOL_NOT_FOUND	8019	未找到指定符号。

通知代码

AP_NOTIFY_ERROR	0	执行函数时出错。
AP_NOTIFY_DATA	1	执行函数时未发生错误。
AP_NOTIFY_CODE_TRANSACT	1	确认 AP_TransAct 调用
AP_NOTIFY_CODE_START	2	确认 AP_Start 调用
AP_NOTIFY_CODE_RESULT	3	动作结果
AP_NOTIFY_CODE_DISCONNECT	6	APDisconnect 调用确认
AP_NOTIFY_CODE_ENDACT	7	确认 AP_EndAct 调用
AP_NOTIFY_CODE_ACTIVE	8	确认 AP_Active 调用
AP_NOTIFY_CODE_INACTIVE	9	确认 AP_Inactive 调用
AP_NOTIFY_ERROR_SERVER_QUITT	10	动作控制已关闭的通知。
AP_NOTIFY_CODE_TRANSRESULT	11	错误结果的通知。调用 APTransAct
AP_NOTIFY_CODE_STARTRESULT	12	错误结果的通知。调用 APStart。
AP_NOTIFY_CODE_RESULT_RT	13	优化调用的周期性结果的通知

3.4.1.4 常数

函数/动作 ID

GSC_AP_SFCT	0x00000001	标准函数
GSC_AP_PFCT	0x00000002	项目函数
GSC_AP_GSC	0x00000004	GSC 动作

限值

AP_MAX_TRIG_NAME	21	触发器名称的最大长度

头类型

CMHF_APDEFAP	0x00000001	项目定义头
CMHF_AP_PBIB	0x00000002	项目函数头
CMHF_AP_GLOB	0x00000004	标准函数头
CMHF_AP_ICF	0x00000008	内部函数头
CMHF_AP_ALL	CMHF_APDEFAP | CMHF_AP_PBIB | CMHF_AP_GLOB | CMHF_AP_ICF	
CMHF_AP_USER	0x00000010	用户函数头

触发器类型

AP_TRIG_UNDEFINED	0	触发器尚未初始化。
AP_TRIG_VAR	1	触发器是一个变量。
AP_TRIG_TIMER	2	触发器是一个定时器。
AP_TRIG_UNKNOWN	3	触发器未知。
AP_TRIG_TRANSACT	4	通过 TransAct 调用指定触发器。

循环类型

AP_TRIG_CYCLE	1	周期性
AP_TRIG_NCYCLE	2	非周期性

循环时间

AP_TRIG_CYCLE_01	1	表示控制中心中的循环时间为 250 ms
AP_TRIG_CYCLE_02	2	表示控制中心中的循环时间为 500 ms
AP_TRIG_CYCLE_03	3	表示控制中心中的循环时间为 1 s
AP_TRIG_CYCLE_04	4	表示控制中心中的循环时间为 2 s
AP_TRIG_CYCLE_05	5	表示控制中心中的循环时间为 5 s
AP_TRIG_CYCLE_06	6	表示控制中心中的循环时间为 10 s
AP_TRIG_CYCLE_07	7	表示控制中心中的循环时间为 1 分钟
AP_TRIG_CYCLE_08	8	表示控制中心中的循环时间为 5 分钟
AP_TRIG_CYCLE_09	9	表示控制中心中的循环时间为 10 分钟
AP_TRIG_CYCLE_10	10	表示控制中心中的循环时间为 1 小时
AP_TRIG_CYCLE_11	11	表示控制中心中的循环时间为“用户定义的 1”
AP_TRIG_CYCLE_12	12	表示控制中心中的循环时间为“用户定义的 2”
AP_TRIG_CYCLE_13	13	表示控制中心中的循环时间为“用户定义的 3”
AP_TRIG_CYCLE_14	14	表示控制中心中的循环时间为“用户定义的 4”
AP_TRIG_CYCLE_15	15	表示控制中心中的循环时间为“用户定义的 5”

3.4.2 结构

3.4.2.1 AP_ACT_KEY

声明

```
typedef struct {
    DWORD     dwKeyType;
    DWORD     dwID;
    CHAR      szActionName[AP_MAX_ACTION_NAME + 1];
    DWORD     dwCycle;
    VARIANT   *pVariant;
    DWORD     dwVariantItem;
    DWORD     dwerror;
    LPVOID    lpvUser;
    } AP_ACT_KEY ;
```

成员

dwKeyType

键类型决定是通过名称 (szActionName) 还是 ID (dwID) 对动作进行寻址。

AP_ID_TYPE	0	通过其 ID 标识动作
AP_NAME_TYPE	1	通过其名称标识动作

WinCC 项目中已存在的项目和标准函数通过名称进行标识。最先通过 APTransAct 函数注册的用户自定义函数通过 ID 进行标识。

dwID

与 dwKeyType = AP_ID_TYPE 关联时，动作通过 dwID 启动。

调用 APTransAct 函数时将分配动作 ID。调用 APTransAct 函数时，必须将 dwID 设置为默认值 0。

szActionName

与 dwKeyType = AP_NAME_TYPE 关联时，动作通过 szActionName 启动。

可在通过 APStart 启动动作时使用。

dwCycle

通过名称寻址时，将循环启动动作。该升级周期通过更新周期列表中包含的条目索引定义。

pVariant

指向 VARIANT 数据类型的字段的指针，用于描述动作的参数。支持以下数据类型（作为 VARTYPES）：

unsigned char	VT_UI1
unsigned integer 2 Bytes	VT_UI2 (WORD)
unsigned integer 4 Bytes	VT_UI4 (DWORD)
short	VT_I2
long	VT_I4
float	VT_R4
double	VT_R8
VARIANT_BOOL	VT_BOOL
SCODE	VT_ERROR
CY	VT_CY
DATE	VT_DATE
BSTR	VT_BSTR
BLOB	VT_BLOB

动作的参数描述和返回结果只能以 VARIANT 数据类型的形式出现。

不允许以引用的形式传送。

dwVariantItem

动作接收为参数的 VARIANT 数据类型的数目。

dwerror

APTransAct 中可出现的错误数。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

说明

AP_ACT_KEY 结构明确标识动作。通过 API 调用 APTransAct 将此键分配为 ID。动作通过此键启动。

所需文件

ap_def.h

API 函数

APActive (页 2291)	激活动作
APInactive (页 2296)	禁用动作
APStart (页 2298)	启动动作
APTransact (页 2301)	注册要处理的动作

参见

APActive (页 2291)

APInactive (页 2296)

APStart (页 2298)

APTransact (页 2301)

3.4.2.2 AP_ACT_RESULT_STRUCT

声明

```
typedef struct {
    VARIANT     *ap_result;
    AP_ACT_KEY  apActKey;
    CMN_ERROR   error;
    DWORD       dwreserved;
    } AP_ACT_RESULT_STRUCT ;
```

成员

***ap_result**

返回 VARIANT 数据类型形式的动作结果。

apActKey

用于标识动作的键。AP_ACT_KEY 对应于启动动作的键。

error

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

dwreserved

该参数为将来开发预留，必须预设为 0。

所需文件

ap_def.h

API 函数

APFreeResultStruct (页 2295)	释放动作结果内存
APStart (页 2298)	启动动作

参见

APFreeResultStruct (页 2295)

APStart (页 2298)

3.4.2.3 CREATE_USER_HEADER_FILE

声明

```
typedef struct {
    char*     pszStartDir;
    char*     pszHeaderFileName;
    BOOL      bShowDlg;
    char*     pszWindowText;
    } CREATE_USER_HEADER_FILE ;
```

成员

pszStartDir

指向在其中搜索用户函数（以终止符"\"开始）的起始目录的指针

pszHeaderFileName

指向头文件名称的指针

bShowDlg

显示含有进度显示的对话框

pszWindowText

指向对话框标签的指针。Default:BROWSER

所需文件

capigsc.h

API 函数

GSCGenCompileUserFunctions (页 2288)	编译用户函数

参见

GSCGenCompileUserFunctions (页 2288)

3.4.2.4 GENERATE_COMPILE

声明

```
typedef struct {
    char*     pszProjectName;
    LPACTION  pAction;
    } GENERATE_COMPILE ;
```

成员

pszProjectName

指向项目名称的指针

pAction

指向包含源代码的有效动作流的指针

所需文件

capigsc.h

相关函数

GSCGenCompile (页 2286)	编译动作

参见

GSCGenCompile (页 2286)

3.4.2.5 GET_ACTION_STREAM

声明

```
typedef struct {
    char*     pszPathName;
    DWORD     dwType;
    } GET_ACTION_STREAM, *LPGET_ACTION_STREAM ;
```

成员

pszPathName

函数或动作的完整文件名

dwType

允许的类型：

GSC_AP_SFCT	默认函数
GSC_AP_PFCT	项目函数
GSC_AP_GSC	动作

所需文件

capigsc.h

API 函数

GSCGenGetActionStream (页 2289)	确定动作流

参见

GSCGenGetActionStream (页 2289)

3.4.3 常规函数

3.4.3.1 APConnect

说明

该函数会注册到动作控制中的应用程序。如果在 fpAppBack 中指定了回调函数，则该函数会异步执行，而在 fpAppBack == NULL 动作中会同步执行。

声明

```
BOOL APConnect (
    LPCSTR        lpszAppName,
    AP_RT_PROC    fpAppBack,
    PDWORD        pdwOrderId,
    LPCVOID       lpvUser,
    PCMN_ERROR    pError )
```

参数

lpszAppName

应用程序名称，应用程序通过该名称注册到 DMConnect。因此，必须先执行 DMConnect。APConnect 只能同步执行。

如果在动作中调用脚本编程的函数，则必须为 lpszAppName 指定"AktSteu"值（注意具体的写入方式），因为 WinCC 任务已使用此名称执行过 DMConnect。

fpAppBack

用于接收消息的回调函数。例如，通过此函数报告动作控制已结束的消息。

如果程序注册了 Notify 例程，则必须定期清空消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

pdwOrderId

调用 APTransAct 函数时分配的作业编号。在同步调用中，pdwOrderID 并不重要；在异步作业中，作业编号在回调函数中提供。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

如果不使用 lpvUser，则必须为其分配 NULL。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

应用程序已注册

FALSE

错误

错误消息

AP_NO_ERROR	未发生错误。
AP_NO_CONNECTION	应用无法建立与动作控制的连接。
AP_ERROR_IPC_SEND	与动作控制进行通信时出错。无法发送作业。
AP_TIMEOUT	目前尚未实现超时。
AP_ALREADY_CONNECTED	应用程序已经与动作控制连接。
AP_FAILURE_PARAM	参数提供错误。
AP_TO_MANY_CLIENTS	达到最大连接数。无法建立新连接。
AP_FAILURE_UNKNOWN	未详细描述错误。
AP_ACT_QUIT	动作控制已结束。

所需文件

ap_def.h

apcli_S.lib

apclient.dll

相关函数

DMConnect (页 1909)	设置到数据管理器的连接
APDisconnect (页 2274)	断开与动作控制的连接
APTransact (页 2301)	注册要处理的动作
AP_RT_PROC (页 2278)	回调函数

示例

建立与脚本编程的连接 (页 2303) "AP01.cpp"

参见

DMConnect (页 1909)

APTransact (页 2301)

AP_RT_PROC (页 2278)

APDisconnect (页 2274)

建立与脚本编程的连接 (页 2303)

函数概览 (页 2260)

3.4.3.2 APDisconnect

说明

该函数会注销动作控制中的应用程序。

声明

```
BOOL APDisconnect (
    AP_RT_PROC    fpAppBack,
    PDWORD        pdwOrderId,
    LPCVOID       lpvUser,
    PCMN_ERROR    pError )
```

参数

fpAppBack

用于接收消息的回调函数。使用回调函数时，该作业异步执行。如果 fpAppBack == NULL，该作业将同步执行。

如果程序注册了 Notify 例程，则必须定期清空消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

pdwOrderId

调用 APTransAct 函数时分配的作业编号。在同步调用中，pdwOrderID 并不重要；在异步作业中，作业编号在回调函数中提供。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

应用程序已注销

FALSE

错误

注释

说明

不得在应用程序（EXE、DLL、OCX 等）的析构函数中使用该调用。这可能会由于 Microsoft 特定的机制而导致调用堵塞，进而导致程序堵塞。

错误消息

AP_NO_ERROR	未发生错误。
AP_NO_CONNECTION	应用无法建立与动作控制的连接。

AP_ERROR_IPC_SEND	与动作控制进行通信时出错。无法发送作业。
AP_TIMEOUT	目前尚未实现超时。
AP_ALREADY_CONNECTED	应用程序已经与动作控制连接。
AP_FAILURE_PARAM	参数提供错误。
AP_TO_MANY_CLIENTS	达到最大连接数。无法建立新连接。
AP_FAILURE_UNKNOWN	未详细描述错误。
AP_ACT_QUIT	动作控制已结束。

所需文件

ap_def.h

apcli_S.lib

apclient.dll

相关函数

APConnect (页 2272)	建立与动作控制的连接
APTransact (页 2301)	注册要处理的动作
AP_RT_PROC (页 2278)	回调函数

示例

建立与脚本编程的连接 (页 2303) "AP01.cpp"

参见

APTransact (页 2301)

APConnect (页 2272)

AP_RT_PROC (页 2278)

建立与脚本编程的连接 (页 2303)

函数概览 (页 2260)

3.4.3.3 APSetLanguage

说明

使用此函数设置输出错误文本时使用的语言。

声明

```
BOOL APSetLanguage (
    const DWORD    dwLanguageID )
```

参数

dwLanguageID

符合 Windows 语言设置的语言 ID。根据新设置的语言返回错误文本。如果不支持该语言，则使用默认语言。

返回值

TRUE

错误文本的语言已更改

FALSE

错误

所需文件

ap_def.h

apcli_S.lib

apclient.dll

参见

函数概览 (页 2260)

3.4.3.4 AP_RT_PROC

说明

如果您的应用程序将针对运行系统中 API 函数的执行采取异步通知，则必须提供 AP_RT_PROC 类型的回调函数。

动作编程的所有函数均使用此函数作为回调函数。

声明

```
BOOL ( * AP_RT_PROC) (
    DWORD     dwAP_Notify,
    WORD     dwAP_NotifyCode,
    DWORD     dwError,
    LPVOID    lpvData,
    DWORD     dwItems,
    DWORD     dwOrderId,
    LPVOID    lpvUser );
```

参数

dwAP_Notify

说明回调函数的类型。可能值为 AP_NOTIFY_ERROR 和 AP_NOTIFY_DATA。

dwAP_NotifyCode

如果 dwAP_Notify == AP_NOTIFY_ERROR，则 lpvData 指向包含错误描述的 CMN_ERROR 类型的结构，且 dwAP_NotifyCode 为 NULL。

如果 dwAP_Notify == AP_NOTIFY_DATA，则 dwAP_NotifyCode 包含回调函数更具体的说明：

AP_NOTIFY_CODE_TRANSACT	通过 APTransAct 确认调用
AP_NOTIFY_CODE_START	通过 APStart 确认调用
AP_NOTIFY_CODE_RESULT	动作结果

dwError

错误编号

lpvData

指向提供的数据的指针。数据结构取决于 dwAP_Notify 和 dwAP_NotifyCode。

dwItems

lpvData 中的条目数。

dwOrderId

调用 API 函数时分配的作业编号。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

返回值

TRUE

函数成功

FALSE

错误

注释

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

在极少数情况下，在函数调用返回之前可能已传送回调。

所需文件

ap_def.h

相关函数

APActive (页 2291)	激活动作
APConnect (页 2272)	注册应用程序
APDisconnect (页 2274)	注销应用程序
APEndAct (页 2293)	注册应用程序控制中的动作
APInactive (页 2296)	禁用动作
APStart (页 2298)	启动动作
APTransact (页 2301)	注册要处理的动作

示例

建立与脚本编程的连接 (页 2303) "AP01.cpp"

参见

APConnect (页 2272)

APDisconnect (页 2274)

APActive (页 2291)

APEndAct (页 2293)

APInactive (页 2296)

APStart (页 2298)

APTransact (页 2301)

建立与脚本编程的连接 (页 2303)

函数概览 (页 2260)

3.4.4 用于处理源代码的函数

3.4.4.1 APCompile

说明

对保存在 lpvScode 中的源代码进行编译，并以 P 代码的形式保存在 lpvPcode 中。应用程序必须为 P 代码提供内存。

声明

```
BOOL APCompile (
    LPCSTR          szProjectName,
    PDWORD          pdwOrderId,
    LPCSTR          lpvScode,
    const DWORD     dwScodeSize,
    LPVOID          *lpvPcode,
    PDWORD          dwPcodeSize,
    const HWND      hwndLog,
    const DWORD     dwDebugFlag,
    PDWORD          nErrors,
    PDWORD          nWarnings,
    LPCVOID         lpvUser,
    PCMN_ERROR      pError )
```

参数

szProjectName

包含有效项目路径的字符串。项目路径决定了使用哪个 Include 和预编译头进行编译。

pdwOrderId

调用 APTransAct 函数时分配的作业编号。该作业编号必须由调用方提供。

lpvScode

指向要编译的源代码的指针。

dwScodeSize

源代码的大小（单位为字节）。

lpvPcode

包含成功编译的 P 代码的指针的地址。内存由 APCompile 函数创建，必须由调用方通过 APFreePCode 函数再次释放。

dwPcodeSize

包含编译后 P 代码的大小（单位为字节）。

hwndLog

输出错误消息时所使用的窗口句柄。窗口句柄也可以为 NULL。使用窗口进行输出时，通过 Windows 函数 WM_COPYDATA 发送消息。

dwDebugFlag

该标记确定 P 代码应提供调试信息（位 1 = 1）还是不提供调试消息（位 1 = 0）。基于性能原因，通常应选择"0"。

位 2 = 1：针对 Include 使用服务器设置

位 2 = 0：针对 Include 使用本地设置

nErrors

发生的错误数。nErrors > 0 时不返回 P 代码。

nWarnings

发生的警告数。nWarnings > 0 时仍生成 P 代码。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在 Callback-Funktion 中会重新提供。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

源代码编译成功

FALSE

错误

所需文件

ap_def.h

apcli_S.lib

apclient.dll

相关函数

APCompileEx (页 2283)	编译源代码
APFreePCode	释放源代码内存

参见

APCompileEx (页 2283)

函数概览 (页 2260)

3.4.4.2 APCompileEx

说明

与 APCompile 函数不同，要编译的源代码会分为多个子部分。

在源代码中，void function {instructions} 分为

void function {	指令的起始部分
Instructions	基本部分
}	指令的结尾部分

将对保存在子部分 lpvScodeProlog、lpvScodeBase 和 lpvScodeEpilog 中的源代码进行编译，并以 P 代码的形式保存在 lpvPcode 中。应用程序必须为 P 代码提供内存。

声明

```
BOOL APCompileEx (
    LPCSTR         szProjectName,
    PDWORD         pdwOrderId,
    LPCSTR         lpvScodeProlog,
    const DWORD    dwScodePrologSize,
    LPCSTR         lpvScodeBase,
    const DWORD    dwScodeBaseSize,
    LPCSTR         lpvScodeEpilog,
    const DWORD    dwScodeEpilogSize,
    LPVOID         *lpvPcode,
    PDWORD         dwPcodeSize,
    const HWND     hwndLog,
    const DWORD    dwDebugFlag,
    PDWORD         nErrors,
    PDWORD         nWarnings,
    LPCVOID        lpvUser,
    PCMN_ERROR     pError )
```

参数

szProjectName

包含有效项目路径的字符串。项目路径决定了使用哪个 Include 和预编译头进行编译。

pdwOrderId

调用 APTransAct 函数时分配的作业编号。该作业编号必须由调用方提供。

szProjectName

包含有效项目路径的字符串。项目路径决定了使用哪个 Include 和预编译头进行编译。

pdwOrderId

调用 APTransAct 函数时分配的作业编号。该作业编号必须由调用方提供。

lpvScodeProlog

指向要编译的源代码起始部分的指针。

dwScodePrologSize

起始部分的大小（单位为字节）。

lpvScodeBase

指向要编译的源代码基础部分的指针。

dwScodeBaseSize

基础部分的大小（单位为字节）。

lpvScodeEpilog

指向要编译的源代码结尾部分的指针。

dwScodeEpilogSize

结尾部分的大小（单位为字节）。

lpvPcode

包含成功编译的 P 代码的指针的地址。内存由 APCompile 函数创建，必须由调用方通过 APFreePCode 函数再次释放。

dwPcodeSize

包含编译后 P 代码的大小（单位为字节）。

hwndLog

输出错误消息时所使用的窗口句柄。窗口句柄也可以为 NULL。使用窗口进行输出时，通过 Windows 函数 WM_COPYDATA 发送消息。

dwDebugFlag

该标记确定 P 代码应提供调试信息（位 1 = 1）还是不提供调试消息（位 1 = 0）。基于性能原因，通常应选择"0"。

位 2 = 1：针对 Include 使用服务器设置

位 2 = 0：针对 Include 使用本地设置

nErrors

发生的错误数。nErrors > 0 时不返回 P 代码。

nWarnings

发生的警告数。nWarnings > 0 时仍生成 P 代码。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在 Callback-Funktion 中会重新提供。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

源代码编译成功

FALSE

错误

所需文件

ap_def.h

apcli_S.lib

apclient.dll

相关函数

APCompile (页 2281)	编译源代码
APFreePCode	释放源代码内存

参见

APCompile (页 2281)

函数概览 (页 2260)

3.4.4.3 GSCGenCompile

说明

该函数用于编译动作。

声明

```
LPACTION GSCGenCompile(
    LPGENERATE_COMPILE  lpGenCompile,
    HWND                hWndParent,
    unsigned long*      plErrors,
    unsigned long*      plWarnings,
    AllocAppMem         lpfnAllocAppMem,
    LPCMN_ERROR         lpdmError)
```

参数

lpGenCompile

指向 GENERATE_COMPILE (页 2270) 结构的指针。

hWndParent

进行状态输出的窗口的句柄。

plErrors

指向表示返回错误数的变量的指针

plWarnings

指向表示返回警告数的变量的指针

lpfnAllocAppMem

指向在调用应用程序中用于分配内存的函数的指针

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

指向编译的动作流的指针。即使编译出现故障，也会生成新的动作流。必须根据错误数和警告数检查调用是否成功。

注释

直到下一次启动运行系统时，编译的动作才可用。

所需文件

capigsc.h

gscgr_s.lib

gscgen.dll

参见

GENERATE_COMPILE (页 2270)

函数概览 (页 2260)

3.4.4.4 GSCGenCompileUserFunctions

说明

该函数会重新编译所有特殊的用户函数。

声明

```
BOOL GSCGenCompileUserFunctions (
    LPCREATE_USER_HEADER_FILE    pGenCUHF,
    HWND                         hWndParent,
    LPCMN_ERROR                  lpdmError)
```

参数

pGenCUHF

指向 CREATE_USER_HEADER_FILE (页 2269) 结构的指针。

hWndParent

进行状态输出的窗口的句柄。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

用户函数已编译

FALSE

错误

注释

出于内部原因，此函数始终返回 FALSE，而不返回错误消息，即使函数已生成。

直到下一次启动运行系统时，编译的用户函数才可用。

所需文件

capigsc.h

gscgr_s.lib

gscgen.dll

参见

CREATE_USER_HEADER_FILE (页 2269)

函数概览 (页 2260)

3.4.5 用于处理动作的函数

3.4.5.1 GSCGenGetActionStream

说明

该函数确定用于项目函数、标准函数或动作的动作流。

声明

```
LPACTION GSCGenGetActionStream(
    LPGET_ACTION_STREAM    pGenGAS,
    AllocAppMem            lpfnAllocAppMem,
    LPCMN_ERROR            lpdmError)
```

参数

pGenGAS

指示用于指定函数或动作的 GET_ACTION_STREAM (页 2271) 类型结构的指针。

lpfnAllocAppMem

指向用于在调用应用程序中分配存储器的函数的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

指向动作流的指针。

所需文件

capigsc.h

gscgr_s.lib

gscgen.dll

参见

GET_ACTION_STREAM (页 2271)

函数概览 (页 2260)

3.4.6 动作编程的函数

3.4.6.1 APActive

说明

只能使用此函数重新激活通过 APInactive 禁用的动作，即相应的触发器被再次监视，或者可通过 APStart 重新激活。

声明

```
BOOL APActive (
    PAP_ACT_KEY    lpapActKey,
    const DWORD    dwItems,
    AP_RT_PROC     fpAppBack,
    PDWORD         pdwOrderId,
    LPCVOID        lpvUser,
    PCMN_ERROR     pError )
```

参数

lpapActKey

指向含有下列结构的数据结构的指针：

AP_ACT_KEY 1
AP_ACT_KEY 2
⋮
AP_ACT_KEY n
动作流 1
⋮
动作流 1
动作流 n

数据结构必须通过调用应用程序提供。

必须先通过 APTransAct 注册动作。AP_ACT_KEY 中的 ID 决定了是通过 ID 还是通过名称启动动作。

dwItems

动作次数，即结构 AP_ACT_KEY (页 2266)。

fpAppBack

用于异步调用的回调函数。使用 NULL 时进行同步调用。

如果程序声明了“通知”例程，必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

pdwOrderId

调用 APTransAct 函数时分配的作业编号。在同步调用中，pdwOrderID 并不重要；在异步作业中，作业编号在回调函数中提供。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

动作已激活

FALSE

错误。

所需文件

ap_def.h

apcli_S.lib

apclient.dll

相关函数

APInactive (页 2296)	禁用动作
APTransact (页 2301)	注册要处理的动作
AP_RT_PROC (页 2278)	回调函数

参见

AP_RT_PROC (页 2278)

AP_ACT_KEY (页 2266)

APInactive (页 2296)

APTransact (页 2301)

函数概览 (页 2260)

3.4.6.2 APEndAct

说明

在动作控制中注销已注册的用于处理的动作。

声明

```
BOOL APEndAct (
    AP_RT_PROC      fpAppBack,
    PDWORD          pdwOrderId,
    const PDWORD    pdwOrderEnd,
    LPCVOID         lpvUser,
    PCMN_ERROR      pError )
```

参数

fpAppBack

用于异步调用的回调函数。使用 NULL 时进行同步调用。

如果程序声明了“通知”例程，必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

pdwOrderId

调用 APTransAct 函数时分配的作业编号。在同步调用中，pdwOrderID 并不重要；在异步作业中，作业编号在回调函数中提供。

pdwOrderEnd

要结束的事务的作业编号。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

动作已注销

FALSE

错误。

所需文件

ap_def.h

apcli_S.lib

apclient.dll

相关函数

APTransact (页 2301)	注册要处理的动作
AP_RT_PROC (页 2278)	回调函数

参见

AP_RT_PROC (页 2278)

APTransact (页 2301)

函数概览 (页 2260)

3.4.6.3 APFreeResultStruct

说明

同步调用 APStart 函数时，将以 AP_ACT_RESULT_STRUCT 类型的结构数组的形式分配动作结果。APStart 分配的内存必须通过 APFreeResultStruct 函数再次释放。

声明

```
BOOL APFreeResultStruct (
    PAP_ACT_RESULT_STRUCT    *lpapars,
    const DWORD              dwItems )
```

参数

lpvPcode

指向 AP_ACT_RESULT_STRUCT (页 2268) 类型的第一个 dwItems 结构的指针。

dwItems

结构 AP_ACT_RESULT_STRUCT 的数目。

返回值

TRUE

释放内存。

FALSE

错误。

所需文件

ap_def.h

apcli_S.lib

apclient.dll

相关函数

APStart (页 2298)	动作处理已开始

参见

APStart (页 2298)

AP_ACT_RESULT_STRUCT (页 2268)

函数概览 (页 2260)

3.4.6.4 APInactive

说明

通过 APTransAct 注册的动作可以通过 APInactive 禁用。不再监视相应的触发器，无法通过 APStart 启动动作。可以通过 APActive 重新激活动作。

声明

```
BOOL APInactive (
    PAP_ACT_KEY     lpActKey,
    const DWORD     dwItems,
    AP_RT_PROC      fpAppBack,
    PDWORD          pdwOrderId,
    LPCVOID         lpvUser,
    PCMN_ERROR      pError )
```

参数

lpapActKey

指向含有下列结构的数据结构的指针：

AP_ACT_KEY 1
AP_ACT_KEY 2
⋮
AP_ACT_KEY n
动作流 1
⋮
动作流 1
动作流 n

数据结构必须通过调用应用程序提供。

必须先通过 APTransAct 注册动作。AP_ACT_KEY 中的 ID 决定了是通过 ID 还是通过名称启动动作。

dwItems

动作次数，即结构 AP_ACT_KEY (页 2266)。

fpAppBack

用于异步调用的回调函数。使用 NULL 时进行同步调用。

如果程序声明了“通知”例程，必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

pdwOrderId

调用 APTransAct 函数时分配的作业编号。在同步调用中，pdwOrderID 并不重要；在异步作业中，作业编号在回调函数中提供。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

动作已禁用

FALSE

错误。

所需文件

ap_def.h

apcli_S.lib

apclient.dll

相关函数

APActive (页 2291)	激活动作
APTransact (页 2301)	注册要处理的动作
AP_RT_PROC (页 2278)	回调函数

参见

AP_RT_PROC (页 2278)

APActive (页 2291)

AP_ACT_KEY (页 2266)

APTransact (页 2301)

函数概览 (页 2260)

3.4.6.5 APStart

说明

启动动作以进行处理。可以同时处理多个动作。AP_ACT_KEY 结构中描述了动作。

声明

```
BOOL APStart (
    AP_ACT_KEY              lpapActKey,
    const DWORD              dwItems,
    AP_RT_PROC               fpAppBack,
    AP_RT_PROC               fpAppResult,
    PDWORD                   pdwOrderId,
    PAP_ACT_RESULT_STRUCT    *lpapars,
    LPCVOID                  lpUser,
    PCMN_ERROR               pError )
```

参数

lpapActKey

指向含有下列结构的数据结构的指针：

数据结构必须通过调用应用程序提供。

必须先通过 APTransAct 注册动作。AP_ACT_KEY 中的 ID 决定了是通过 ID 还是通过名称启动动作。

dwItems

动作次数，即结构 AP_ACT_KEY (页 2266)。

fpAppBack

用于异步调用的回调函数。使用 NULL 时进行同步调用。

如果程序声明了“通知”例程，必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

fAppResult

用于循环动作结果的回调函数。回调函数中提供 AP_ACT_RESULT_STRUCT (页 2268) 类型的结构。在同步调用中，结果分配给 lpapars。

pdwOrderId

调用 APTransAct 函数时分配的作业编号。在同步调用中，pdwOrderID 并不重要；在异步作业中，作业编号在回调函数中提供。

lpapars

指向同步调用中的动作结果的指针。将根据启动动作的次数返回 AP_ACT_RESULT_STRUCT (页 2268) 类型结构的数组。该函数分配的内存必须通过 APFreeResultStruct 函数再次释放。

在异步调用中，lpapars = NULL

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

pError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

动作已启动

FALSE

错误。

注释

对于 APStart，只能启动之前未通过 APInactive 禁用的动作。

所需文件

ap_def.h

apcli_S.lib

apclient.dll

相关函数

APFreeResultStruct (页 2295)	释放内存
AP_RT_PROC (页 2278)	回调函数

参见

AP_RT_PROC (页 2278)

APFreeResultStruct (页 2295)

AP_ACT_KEY (页 2266)

AP_ACT_RESULT_STRUCT (页 2268)

函数概览 (页 2260)

3.4.6.6 APTransact

应用程序

声明

参数

xxx

xxx

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统会将错误信息写入该结构。

返回值

TRUE

.

FALSE

错误

注释

所需文件

ap_def.h

apcli_S.lib

apclient.dll

相关函数

示例

Auto-Hotspot "DM01.cpp"

参见

APConnect (页 2272)

APDisconnect (页 2274)

AP_RT_PROC (页 2278)

APActive (页 2291)

APEndAct (页 2293)

APInactive (页 2296)

AP_ACT_KEY (页 2266)

函数概览 (页 2260)

3.4.7 示例

3.4.7.1 建立与脚本编程的连接

概述

```
// ========================================================================
// Filename:.......... ap01.c
// ========================================================================
// : Modul with examples of AP_API
// ************************************************************************
// Copyright (C) 1995/96 SIEMENS AG, AUT 913 All rights reserved
// ************************************************************************
#include "stdafx.h"
#include "ap01.h" // if console application

//{{ODK_EXAMPLE}Establish connection to script programming (AP)}
//{{FUNCTION}APConnect (AP)}
//{{FUNCTION}APDisconnect (AP)}
//{{FUNCTION}AP_RT_PROC (AP)}
//{{FUNCTION}(END)}
// Establish connection to script programming (AP)
// ========================================================================
// Function: AprConnect(void) ODK AP CS
// ========================================================================
// short : Establish connection to script programming
// :
// ========================================================================
BOOL MyAPRTCallback(DWORD dwAP_Notify, DWORD dwAP_NotifyCode, DWORD dwError,
                 LPVOID lpvData, DWORD dwItems, DWORD dwOrderID, LPVOID lpvUser)
{
    lpvUser;
    dwOrderID;
    lpvData;
    TCHAR szText[255];
     _sntprintf_s( szText,_countof(szText), _TRUNCATE, _T("AprNotCon:: AP= %d ;
               dwAP_Notify, dwAP_NotifyCode, dwError, dwItems);
    ODKTrace(szText);
    return(TRUE );
}

void MyApConnect(void)
{
    TCHAR szText[255];
    CMN_ERROR Error;
    BOOL ret= FALSE;
    DWORD dwOrderID = 0;
    TCHAR szApp[255];
    VOID* pUser = NULL;
    _tcsncpy_s(szApp, _countof(szApp), _T("MyODKApp_23"), _TRUNCATE); // must be the same
AppName as by DMConnect
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = APConnect(szApp, MyAPRTCallback, &dwOrderID, pUser, &Error);
    if(FALSE == ret)
    {
```

```
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in APConnect: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("APConnect"));
    }
    ODKTrace(szText);
    //printf("%s\r\n"szText);
    void MyAPDisconnect()
    VOID* pUser = NULL;
    CMN_ERROR Error;
    TCHAR szText[255];
    BOOL ret = FALSE;
    DWORD dwOrderID = 0;
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = APDisconnect(NULL, &dwOrderID, pUser, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in APDisconnect: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("APDisconnect"));
    }
    ODKTrace(szText);
}
// ----------------------------------------------------------------------
//{{ODK_EXAMPLE}(END)}
```

参见

APConnect (页 2272)

APDisconnect (页 2274)

AP_RT_PROC (页 2278)

3.5 用户管理的函数

3.5.1 基本知识

3.5.1.1 函数概览

概述

PWGEN_ENUM_GROUPS_CALLBACK (页 2330)	列出用户组（回调）
PWGEN_ENUM_LEVELS_CALLBACK (页 2338)	列出授权级别（回调）
PWGEN_ENUM_USERS_CALLBACK (页 2325)	列出用户（回调）
PWGENAddGroup (页 2327)	创建用户组
PWGENAddPermLevel (页 2332)	创建授权级别
PWGENAddUser (页 2315)	创建用户
PWGENAddUserEx (页 2317)	创建用户
PWGENChangePassword (页 2319)	更改密码
PWGENCheckPermission (页 2333)	验证用户的用户授权
PWGENCheckUser (页 2320)	检查用户
PWGENConnect (页 2312)	建立与数据库的连接。
PWGENDeletePermLevel (页 2335)	删除授权级别
PWGENDeleteUser (页 2322)	删除用户/删除用户组
PWGENDisconnect (页 2313)	终止与数据库的连接
PWGENEnumGroups (页 2329)	列出用户组
PWGENEnumPermLevels (页 2336)	列出授权级别
PWGENEnumUsers (页 2324)	列出用户
PWGENReadUserPerm (页 2339)	确定用户授权
PWRTCheckPermission (页 2341)	检查区域授权
PWRTCheckPermissionOnPicture (页 2342)	检查授权级别
PWRTGetCurrentUser (页 2347)	确定登录名称

PWRTGetLoginPriority (页 2348)	GetLoginPriority
PWRTIsLoggedInByCard (页 2349)	通过芯片卡登录
PWRTLogin (页 2350)	通过对话框登录
PWRTLogout (页 2352)	注销
PWRTLogoutEx (页 2353)	注销
PWRTPermissionLevelDialog (页 2343)	选择授权级别
PWRTPermissionLevelDialogEx (页 2344)	选择授权级别
PWRTPermissionToString (页 2345)	确定文本说明的授权级别
PWRTSilentLogin (页 2354)	登录
PWRTSilentLoginEx (页 2356)	登录

3.5.1.2 结构概览

概述

PWGEN_GROUPINFO (页 2309)	用户组信息
PWGEN_LEVELINFO (页 2310)	授权级别信息
PWGEN_USERINFO (页 2311)	用户信息

3.5.1.3 错误消息

概述

CMN_ERROR 错误结构中的 API 函数可返回以下错误消息：

RT 的错误消息：

PWRT_CAP_SHMERROR	1	共享内存错误
PWRT_CAP_NOLOGIN	2	无用户登录
PWRT_CAP_NOPERM	3	用户不具有授权。
PWRT_CAP_NOAREA	4	无法找到区域。
PWRT_GPA_SHMERROR	1	共享内存错误
PWRT_GPA_NOLOGIN	2	无用户登录

CS V5.0 和更高版本的错误消息：

PWGEN_API_ERR_SUPPLY	0x10000001L	发生未指明错误。
PWGEN_API_NO_MEMORY	0x10000001L	内存不足。
PWGEN_API_NOT_SUPPORTED	0x10000003L	不支持该函数
PWGEN_API_INVALID_PARAM	0x10000004L	函数参数不正确或缺失。
PWGEN_API_NO_INTERFACE	0x10000005L	未连接到 UserAdminASO-Interface。
PWGEN_API_I_ENUM	0x10000006L	枚举时出错
PWGEN_API_I_READ	0x10000007L	读取时出错
PWGEN_API_I_CREATE	0x10000008L	创建新数据时出错
PWGEN_API_I_MODIFY	0x10000009L	修改数据时出错
PWGEN_API_I_DELETE	0x1000000AL	删除数据时出错
PWGEN_API_TERMINATED	0x1000000FL	处理被过早取消。
PWGEN_API_NO_CONNECTION	0x10000010L	PWGENConnect 尚未执行。
PWGEN_API_IS_CONNECTED	0x10000011L	PWGENConnect 已执行。
PWGEN_API_NO_USER	0x10000020L	未找到用户
PWGEN_API_MAX_USER	0x10000021L	已达到最大用户数（约 128）。
PWGEN_API_EXIST_USER	0x10000022L	所选用户已存在。
PWGEN_API_NO_GROUP	0x10000030L	未找到用户组。
PWGEN_API_MAX_GROUP	0x10000031L	已达到最大用户组数（约 10）。
PWGEN_API_EXIST_GROUP	0x10000032L	所选用户组已存在。
PWGEN_API_NO_LEVEL	0x10000040L	未找到授权级别。
PWGEN_API_MAX_LEVEL	0x10000041L	已达到最大授权级别数（约 1000）。
PWGEN_API_EXIST_LEVEL	0x10000042L	所选授权级别已存在。
PWGEN_API_NO_PERM	0x10000050L	未分配授权。
PWGEN_API_MAX_PERM	0x10000051L	已达到最大授权级别数。
PWGEN_API_EXIST_PERM	0x10000052L	授权已存在。
PWGEN_API_NO_AREA	0x10000060L	未分配范围。
PWGEN_API_MAX_AREA	0x10000061L	已达到最大区域数（约 32）。
PWGEN_API_EXIST_AREA	0x10000062L	区域已存在。
PWGEN_API_NO_PASSWORD	0x10000070L	无密码或无有效密码。
PWGEN_API_MAX_PASSWORD	0x10000071L	已达到最大密码数。
PWGEN_API_EXIST_PASSWORD	0x10000072L	密码已分配。

3.5.1.4 常数

字符串和数组大小

MAX_LOGIN	25
MAX_PASS	25
MAX_LEVEL	70

3.5.2 结构

3.5.2.1 PWGEN_GROUPINFO

声明

```
typedef struct {
    TCHAR    name[MAX_LOGIN];
    int      expiration_time;
    }
    PWGEN_GROUPINFO;
```

成员

name

用户组名称

expiration_time

自动注销时间（单位为分钟）

所需文件

usegenap.h

API 函数

PWGEN_ENUM_GROUPS_CALLBACK (页 2330)	列出用户组

参见

PWGEN_ENUM_GROUPS_CALLBACK (页 2330)

3.5.2.2 PWGEN_LEVELINFO

声明

```
typedef struct {
    int      levelNumber;
    DWORD    dwTextID;
    }
    PWGEN_LEVELINFO;
```

成员

levelNumber

授权数

dwTextID

用于标识授权描述的 ID。

所需文件

usegenap.h

API 函数

PWGEN_ENUM_LEVELS_CALLBACK (页 2338)	列出授权级别

参见

PWGEN_ENUM_LEVELS_CALLBACK (页 2338)

3.5.2.3 PWGEN_USERINFO

声明

```
typedef struct {
    TCHAR    login[MAX_LOGIN];
    TCHAR    group[MAX_LOGIN];
    int      expiration_time;
    }
    PWGEN_USERINFO;
```

成员

login

用户的登录名称

group

用户所属组的名称

expiration_time

自动注销时间（单位为分钟）

所需文件

usegenap.h

API 函数

PWGEN_ENUM_USERS_CALLBACK (页 2325)	列出用户

参见

PWGEN_ENUM_USERS_CALLBACK (页 2325)

3.5.3 常规函数

3.5.3.1 PWGENConnect

说明

该函数用于连接到当前打开项目的数据库。

声明

```
BOOL PWGENConnect (
    LPCTSTR        DSNName,
    LPCMN_ERROR    err)
```

参数

DSNName

项目的数据源名称。不对 DSNName 参数进行评估。可以传递空字符串，但不可以传递 ZERO。始终使用当前打开项目的数据库。

err

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

连接已建立

FALSE

错误

注释

每个应用程序只能执行一次 Connect。如果重复执行 Connect，会返回 PWGEN_API_IS_CONNECTED 错误消息。

错误消息

PWGEN_API_INVALID_PARAM	函数参数不正确或缺失。
PWGEN_API_ERR_SUPPLY	发生未指明错误。
PWGEN_API_NO_INTERFACE	未连接到 UserAdminASO 接口。
PWGEN_API_NO_CONNECTION	PWGENConnect 尚未执行。
PWGEN_API_IS_CONNECTED	PWGENConnect 已执行。

所需文件

usegenap.h

usegen.lib

usegen.dll

相关函数

PWGENDisconnect (页 2313)	终止与数据库的连接

参见

PWGENDisconnect (页 2313)

函数概览 (页 2306)

3.5.3.2 PWGENDisconnect

说明

终止与当前打开项目的数据库的连接。

声明

```
BOOL PWGENDisconnect (
    LPCMN_ERROR    err)
```

参数

err

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

连接被终止

FALSE

错误

注释

应在应用程序关闭之前执行 PWGENDisconnect，否则内部使用的 UserAdminASO 将不再启用。

说明

调用不可用在应用程序（EXE、DLL、OCX 等）的析构函数中。由于 Microsoft 特定的机制，这可能导致调用冻结，从而导致程序崩溃。

错误消息

PWGEN_API_ERR_SUPPLY	发生未指明错误。
PWGEN_API_NO_INTERFACE	未连接到 UserAdminASO 接口。
PWGEN_API_NO_CONNECTION	PWGENConnect 尚未执行。

所需文件

usegenap.h

usegen.lib

usegen.dll

相关函数

PWGENConnect (页 2312)	建立与数据库的连接。

参见

PWGENConnect (页 2312)

函数概览 (页 2306)

3.5.4 用于处理用户的函数

3.5.4.1 PWGENAddUser

说明

该函数会创建新用户，包括空的用户授权矩阵。可以使用 PWGENAddUserEx 函数将组授权传递给要创建的用户。

声明

```
BOOL PWGENAddUser (
    LPCTSTR         username,
    LPCTSTR         password,
    LPCTSTR         group,
    int             expiration_time,
    LPCMN_ERROR     error )
```

参数

username

用户登录名称

password

用户密码

group

用户应添加至的组

expiration_time

自动注销时间（单位为分钟）如果选择的 expiration_time 超出允许范围，则将被设置为 0。

error

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

用户已创建

FALSE

错误

错误消息

PWGEN_API_INVALID_PARAM	函数参数不正确或缺失。
PWGEN_API_ERR_SUPPLY	发生未指明错误。
PWGEN_API_NO_INTERFACE	未连接到 UserAdminASO 接口。
PWGEN_API_NO_CONNECTION	PWGENConnect 尚未执行。
PWGEN_API_NO_MEMORY	内存不足。
PWGEN_API_I_ENUM	枚举出错。
PWGEN_API_EXIST_USER	所选用户已存在。
PWGEN_API_MAX_USER	已达到最大用户数（约 128）。
PWGEN_API_NO_GROUP	无法找到用户组。
PWGEN_API_I_CREATE	创建新数据时出错。

所需文件

usegenap.h

usegen.lib

usegen.dll

相关函数

PWGENAddUserEx (页 2317)	创建用户

参见

PWGENAddUserEx (页 2317)

函数概览 (页 2306)

3.5.4.2 PWGENAddUserEx

说明

该函数用于创建新用户。根据 copy_group_protection 参数，可以将组授权传递给要创建的用户。

声明

```
BOOL PWGENAddUserEx (
    LPCTSTR         username,
    LPCTSTR         password,
    LPCTSTR         group,
    int             expiration_time,
    BOOL            copy_group_permissions,
    LPCMN_ERROR     error )
```

参数

username

用户登录名称

password

用户密码

group

用户应添加至的组

expiration_time

自动注销时间（单位为分钟）如果选择的 expiration_time 超出允许范围，则将被设置为 0。

copy_group_permissions

如果 copy_group_permission = TRUE，则会将组的用户授权传递给新创建的用户。

error

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

用户已创建

FALSE

错误

错误消息

PWGEN_API_INVALID_PARAM	函数参数不正确或缺失。
PWGEN_API_ERR_SUPPLY	发生未指明错误。
PWGEN_API_NO_INTERFACE	未连接到 UserAdminASO 接口。
PWGEN_API_NO_CONNECTION	PWGENConnect 尚未执行。
PWGEN_API_NO_MEMORY	内存不足。
PWGEN_API_I_ENUM	枚举出错。
PWGEN_API_EXIST_USER	所选用户已存在。
PWGEN_API_MAX_USER	已达到最大用户数（约 128）。
PWGEN_API_NO_GROUP	无法找到用户组。
PWGEN_API_I_CREATE	创建新数据时出错。

所需文件

usegenap.h

usegen.lib

usegen.dll

相关函数

PWGENAddUser (页 2315)	创建用户

参见

PWGENAddUser (页 2315)

函数概览 (页 2306)

3.5.4.3 PWGENChangePassword

说明

该函数可用于更改通过用户名指定的用户密码。

声明

```
BOOL PWGENChangePassword (
        LPCTSTR         username,
        LPCTSTR         oldpassword,
        LPCTSTR         newpassword,
        LPCMN_ERROR     error )
```

参数

username

用户登录名称

oldpassword

旧用户密码

newpassword

新用户密码

error

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

密码已更改

FALSE

错误

错误消息

PWGEN_API_INVALID_PARAM	函数参数不正确或缺失。
PWGEN_API_NO_PASSWORD	无密码或无有效密码。
PWGEN_API_ERR_SUPPLY	发生未指明错误。
PWGEN_API_NO_INTERFACE	未连接到 UserAdminASO 接口。
PWGEN_API_NO_CONNECTION	PWGENConnect 尚未执行。
PWGEN_API_NO_MEMORY	内存不足。
PWGEN_API_I_ENUM	枚举出错。
PWGEN_API_NO_USER	无法找到用户。
PWGEN_API_I_READ	读取时出错。
PWGEN_API_I_MODIFY	更改数据时出错。

所需文件

usegenap.h

usegen.lib

usegen.dll

参见

函数概览 (页 2306)

3.5.4.4 PWGENCheckUser

说明

该函数会检查当前 WinCC 项目中是否存在指定用户，并检查给定的密码是否正确。

声明

```
BOOL PWGENCheckUser (
    LPCTSTR        username,
    LPCTSTR        password,
    LPCMN_ERROR    error )
```

参数

username

用户的名称。

password

属于指定用户的密码。

error

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

用户存在且密码正确。

FALSE

错误

错误消息

PWGEN_API_INVALID_PARAM	函数参数不正确或缺失。
PWGEN_API_ERR_SUPPLY	发生未指明错误。
PWGEN_API_NO_INTERFACE	未连接到 UserAdminASO 接口。
PWGEN_API_NO_CONNECTION	PWGENConnect 尚未执行。
PWGEN_API_I_ENUM	枚举出错。
PWGEN_API_NO_USER	无法找到用户。
PWGEN_API_NO_PASSWORD	无密码或无有效密码。

所需文件

usegenap.h

usegen.lib

usegen.dll

参见

函数概览 (页 2306)

3.5.4.5 PWGENDeleteUser

说明

删除指定用户或用户组，包括授权数据。

声明

```
BOOL PWGENDeleteUser (
    LPCTSTR        username,
    BOOL           is_user,
    LPCMN_ERROR    error )
```

参数

username

要删除的用户的编号。

is_user

is_user 参数用于区分应删除用户还是用户组。

TRUE	应删除用户。
FALSE	应删除用户组。

error

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

用户或用户组已删除

FALSE

错误

注释

如果尝试删除管理员，该函数将返回 PWGEN_API_EXIST_USER 错误。

错误消息

PWGEN_API_INVALID_PARAM	函数参数不正确或缺失。
PWGEN_API_ERR_SUPPLY	发生未指明错误。
PWGEN_API_NO_INTERFACE	未连接到 UserAdminASO 接口。
PWGEN_API_NO_CONNECTION	PWGENConnect 尚未执行。
PWGEN_API_I_ENUM	枚举出错。
PWGEN_API_I_DELETE	删除数据时出错。
PWGEN_API_NO_USER	无法找到用户。
PWGEN_API_NO_GROUP	无法找到用户组。
PWGEN_API_EXIST_USER	所选用户已存在。
PWGEN_API_NO_MEMORY	内存不足。

所需文件

usegenap.h

usegen.lib

usegen.dll

参见

函数概览 (页 2306)

3.5.4.6 PWGENEnumUsers

说明

该函数读取已组态用户、针对每个用户调用回调函数并在 dwCount 中返回用户数。

声明

```
BOOL PWGENEnumUsers (
    LPDWORD                         dwCount,
    PWGEN_ENUM_USERS_CALLBACK       cfn,
    PVOID                           userdata,
    LPCMN_ERROR                     error )
```

参数

dwCount

指向应存储用户数的位置的指针。

cfn

用于接收信息的回调函数。如果 cfn == NULL，则只对用户进行计数。

userdata

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

error

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

用户已列出

FALSE

错误

错误消息

PWGEN_API_INVALID_PARAM	函数参数不正确或缺失。
PWGEN_API_ERR_SUPPLY	发生未指明错误。
PWGEN_API_NO_INTERFACE	未连接到 UserAdminASO 接口。
PWGEN_API_NO_CONNECTION	PWGENConnect 尚未执行。
PWGEN_API_I_ENUM	枚举出错。
PWGEN_API_NO_MEMORY	内存不足。

所需文件

usegenap.h

usegen.lib

usegen.dll

相关函数

PWGEN_ENUM_USERS_CALLBACK (页 2325)	列出用户（回调）

参见

PWGEN_ENUM_USERS_CALLBACK (页 2325)

函数概览 (页 2306)

3.5.4.7 PWGEN_ENUM_USERS_CALLBACK

说明

为了评估系统列出的用户信息，必须提供 PWGEN_ENUM_USERS_CALLBACK 类型的回调函数。

声明

```
BOOL ( * PWGEN_ENUM_USERS_CALLBACK) (
    LPWGEN_USERINFO            lpUserInfo,
    PVOID                    lpUser);
```

参数

lpUserInfo

指向具有用户数据的 PWGEN_USERINFO (页 2311) 类型结构的指针。

lpUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

用户已列出

FALSE

错误

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

usegenap.h

相关函数

PWGENEnumUsers (页 2324)	列出用户

参见

PWGENEnumUsers (页 2324)

PWGEN_USERINFO (页 2311)

函数概览 (页 2306)

3.5.5 用于处理用户组的函数

3.5.5.1 PWGENAddGroup

说明

该函数会创建新用户组，包括空的用户授权矩阵。

声明

```
BOOL PWGENAddGroup (
    LPCTSTR         username,
    int             expiration_time,
    LPCMN_ERROR     error )
```

参数

username

要创建的组的名称。

expiration_time

自动注销时间。如果选择的 expiration_time 超出允许范围，则将被设置为 0。

error

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

用户组已创建

FALSE

用户组已存在

注释

使用 PWGENDeleteUser 函数可删除现有用户组。

错误消息

PWGEN_API_INVALID_PARAM	函数参数不正确或缺失。
PWGEN_API_ERR_SUPPLY	发生未指明错误。
PWGEN_API_NO_INTERFACE	未连接到 UserAdminASO 接口。
PWGEN_API_NO_CONNECTION	PWGENConnect 尚未执行。
PWGEN_API_I_ENUM	枚举出错。
PWGEN_API_EXIST_GROUP	所选用户组已存在。
PWGEN_API_MAX_GROUP	已达到最大用户组数（约 10）。
PWGEN_API_NO_MEMORY	内存不足。
PWGEN_API_I_CREATE	创建新数据时出错。

所需文件

usegenap.h

usegen.lib

usegen.dll

相关函数

PWGENDeleteUser (页 2322)	创建用户

参见

PWGENDeleteUser (页 2322)

函数概览 (页 2306)

3.5.5.2 PWGENEnumGroups

说明

该函数读取已组态用户组、针对每个组调用回调函数并在 dwCount 中返回组数。

声明

```
BOOL PWGENEnumGroups (
    LPDWORD                         dwCount,
    PWGEN_ENUM_GROUPS_CALLBACK      cfn,
    PVOID                           userdata,
    LPCMN_ERROR                     error )
```

参数

dwCount

指向应存储用户组数的位置的指针。

cfn

用于接收信息的回调函数。如果 cfn == NULL，则只对用户组进行计数。

userdata

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

error

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

用户组已列出

FALSE

错误

错误消息

PWGEN_API_INVALID_PARAM	函数参数不正确或缺失。
PWGEN_API_ERR_SUPPLY	发生未指明错误。
PWGEN_API_NO_INTERFACE	未连接到 UserAdminASO 接口。
PWGEN_API_NO_CONNECTION	PWGENConnect 尚未执行。
PWGEN_API_I_ENUM	枚举出错。
PWGEN_API_NO_MEMORY	内存不足。

所需文件

usegenap.h

usegen.lib

usegen.dll

相关函数

PWGEN_ENUM_GROUPS_CALLBACK (页 2330)	列出用户组（回调）

参见

PWGEN_ENUM_GROUPS_CALLBACK (页 2330)

函数概览 (页 2306)

3.5.5.3 PWGEN_ENUM_GROUPS_CALLBACK

说明

为了评估系统列出的用户组信息，必须提供 PWGEN_ENUM_GROUPS_CALLBACK 类型的回调函数。

声明

```
BOOL ( * PWGEN_ENUM_GROUPS_CALLBACK) (
    LPWGEN_GROUPINFO    lpGroupInfo,
    PVOID               lpUser);
```

参数

lpUserInfo

指向具有用户组数据的 PWGEN_GROUPINFO (页 2309) 类型结构的指针。

lpUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

用户组已列出

FALSE

错误

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

usegenap.h

相关函数

PWGENEnumGroups (页 2329)	列出用户组

参见

PWGENEnumGroups (页 2329)

PWGEN_GROUPINFO (页 2309)

函数概览 (页 2306)

3.5.6 用于处理授权的函数

3.5.6.1 PWGENAddPermLevel

说明

为所有用户创建新授权。

声明

```
BOOL PWGENAddPermLevel (
    DWORD          txtID,
    int            number,
    LPCMN_ERROR    error )
```

参数

txtID

授权文本的文本 ID。

number

要创建的授权的编号。

error

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

授权级别已创建

FALSE

错误

错误消息

PWGEN_API_INVALID_PARAM	函数参数不正确或缺失。
PWGEN_API_ERR_SUPPLY	发生未指明错误。
PWGEN_API_NO_INTERFACE	未连接到 UserAdminASO 接口。
PWGEN_API_NO_CONNECTION	PWGENConnect 尚未执行。
PWGEN_API_I_ENUM	枚举出错。
PWGEN_API_NO_MEMORY	内存不足。
PWGEN_API_MAX_LEVEL	已达到最大授权级别数 (1000)。
PWGEN_API_EXIST_LEVEL	所选授权级别已存在。
PWGEN_API_I_CREATE	创建新数据时出错。

所需文件

usegenap.h

usegen.lib

usegen.dll

参见

函数概览 (页 2306)

3.5.6.2 PWGENCheckPermission

说明

该函数用于检查用户是否具有指定授权级别的操作员权限。

声明

```
BOOL PWGENCheckPermission(
    LPCTSTR        username,
    DWORD          permlevel,
    LPCMN_ERROR    error)
```

参数

username

用户名

permlevel

要检查的授权的编号

error

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

对应于用户授权级别的操作员授权可用。

FALSE

错误

错误消息

PWGEN_API_INVALID_PARAM	函数参数不正确或缺失。
PWGEN_API_ERR_SUPPLY	发生未指明错误。
PWGEN_API_NO_INTERFACE	未连接到 UserAdminASO 接口。
PWGEN_API_NO_CONNECTION	PWGENConnect 尚未执行。
PWGEN_API_I_ENUM	枚举出错。
PWGEN_API_NO_MEMORY	内存不足。
PWGEN_API_NO_LEVEL	无法找到授权级别。

PWGEN_API_I_READ	读取时出错。
PWGEN_API_NO_PERM	未授权。

所需文件

usegenap.h

usegen.lib

usegen.dll

参见

函数概览 (页 2306)

3.5.6.3 PWGENDeletePermLevel

说明

删除用户授权矩阵中的 levelNumber 指定的授权级别。

声明

```
BOOL PWGENDeletePermLevel (
    int             levelNumber,
    LPCMN_ERROR     error )
```

参数

levelNumber

要删除的授权的编号

error

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

授权级别已删除

FALSE

错误

错误消息

PWGEN_API_INVALID_PARAM	函数参数不正确或缺失。
PWGEN_API_ERR_SUPPLY	发生未指明错误。
PWGEN_API_NO_INTERFACE	未连接到 UserAdminASO 接口。
PWGEN_API_NO_CONNECTION	PWGENConnect 尚未执行。
PWGEN_API_I_ENUM	枚举出错。
PWGEN_API_NO_MEMORY	内存不足。
PWGEN_API_NO_LEVEL	无法找到授权级别。
PWGEN_API_I_DELETE	删除数据时出错。

所需文件

usegenap.h

usegen.lib

usegen.dll

参见

函数概览 (页 2306)

3.5.6.4 PWGENEnumPermLevels

说明

该函数读取已组态授权级别、针对每个级别调用回调函数并在 dwCount 中返回授权级别数。

声明

```
BOOL PWGENEnumPermLevels (
    LPDWORD                         dwCount,
    PWGEN_ENUM_LEVELS_CALLBACK      cfn,
    PVOID                           userdata,
    LPCMN_ERROR                     error )
```

参数

dwCount

指向应存储授权级别数的位置的指针。

cfn

用于接收信息的回调函数。如果 cfn == NULL，则只对授权进行计数。

userdata

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

error

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

授权级别已列出

FALSE

错误

错误消息

PWGEN_API_INVALID_PARAM	函数参数不正确或缺失。
PWGEN_API_ERR_SUPPLY	发生未指明错误。
PWGEN_API_NO_INTERFACE	未连接到 UserAdminASO 接口。
PWGEN_API_NO_CONNECTION	PWGENConnect 尚未执行。

PWGEN_API_I_ENUM	枚举出错。
PWGEN_API_NO_MEMORY	内存不足。

所需文件

usegenap.h

usegen.lib

usegen.dll

相关函数

PWGEN_ENUM_LEVELS_CALLBACK (页 2338)	列出授权（回调）

参见

PWGEN_ENUM_LEVELS_CALLBACK (页 2338)

函数概览 (页 2306)

3.5.6.5 PWGEN_ENUM_LEVELS_CALLBACK

说明

为了评估系统列出的授权级别，必须提供 PWGEN_ENUM_LEVELS_CALLBACK 类型的回调函数。

声明

```
BOOL ( * PWGEN_ENUM_LEVELS_CALLBACK) (
    LPWGEN_LEVELINFO    lpLevelInfo,
    PVOID               lpUser);
```

参数

lpLevelInfo

指向具有授权级别数据的 PWGEN_LEVELINFO (页 2310) 类型结构的指针。

lpUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

授权级别已列出

FALSE

错误

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

usegenap.h

相关函数

PWGENEnumPermLevels (页 2336)	列出授权级别

参见

PWGENEnumPermLevels (页 2336)

PWGEN_LEVELINFO (页 2310)

函数概览 (页 2306)

3.5.6.6 PWGENReadUserPerm

说明

可以使用此函数来确定用户拥有特定授权级别的操作员授权的区域。

声明

```
BOOL PWGENReadUserPerm(
    LPCTSTR    username,
    BOOL       is_user,
    int        levelnumber,
    LPBYTE     freigabe,
    LPDWORD    areaperms )
```

参数

username

用户或用户组的名称

is_user

is_user 表示 username 指的是用户还是用户组。

TRUE	用户
FALSE	用户组

levelnumber

要读取的授权的编号

freigabe

如果 freigabe = 1，则用户对所有区域具有此授权级别的操作员授权。

areaperms

这些区域的特点是具有 32 位的 areaperm。如果用户具有某个区域的授权，则会置位相应位。最低有效位对应于第一个区域。

返回值

TRUE

操作员授权已确定

FALSE

错误

所需文件

usegenap.h

usegen.lib

usegen.dll

参见

函数概览 (页 2306)

3.5.6.7 PWRTCheckPermission

说明

检查登录用户是否拥有给定授权的操作员授权。

声明

```
BOOL PWRTCheckPermission (
    DWORD    permlevel,
    DWORD    suppress_messagebox )
```

参数

permlevel

要检查的授权级别的编号

suppress_messagebox

如果 suppress_messagebox != 0，则不会显示任何对话框。

返回值

TRUE

已授权

FALSE

未授权

所需文件

pwrt_api.h

pass_s.lib

useadmin.dll

示例

PWRT 检查许可 (页 2358) "PWRTBunch.cpp"

参见

PWRT 检查许可 (页 2358)

函数概览 (页 2306)

3.5.6.8 PWRTCheckPermissionOnPicture

说明

检查登录用户是否拥有给定授权的操作员授权。

声明

```
BOOL PWRTCheckPermissionOnPicture (
    DWORD          permlevel,
    LPCTSTR        picture_name,
    DWORD          suppress_messagebox,
    LPCMN_ERROR    lperr )
```

参数

permlevel

待检查权限的编号。

picture_name

含有待测试对象的画面的名称。

suppress_messagebox

如果 suppress_messagebox != 0，则不会显示任何对话框。

lperr

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已授权

FALSE

未授权

所需文件

pwrt_api.h

pass_s.lib

useadmin.dll

示例

检查画面的某等级的权限 (页 2360) "PWRTBunch.cpp"

参见

检查画面的某等级的权限 (页 2360)

函数概览 (页 2306)

3.5.6.9 PWRTPermissionLevelDialog

说明

应打开对话框，利用 PWRTPermisssionLevelDialogEx 函数选择权限，因为之后会在移动对话框窗口时执行重画。

声明

```
LONG PWRTPermissionLevelDialog (
    )
```

参数

无

相关函数

PWRTPermissionLevelDialogEx (页 2344)	通过对话框选择权限

示例

通过指定可能出现的错误在对话框中进行授权级别查询 (页 2362) "PWRTBunch.cpp"

参见

通过指定可能出现的错误在对话框中进行授权级别查询 (页 2362)

PWRTPermissionLevelDialogEx (页 2344)

PWRTPermissionToString (页 2345)

函数概览 (页 2306)

3.5.6.10 PWRTPermissionLevelDialogEx

说明

保护 WinCC 中通过对话框生成的对象的可用性以选择等级。PWRTPermissionLevelDialogEx 显示相应的对话框，该对话框提供可供选择的可用权限。

声明

```
LONG PWRTPermissionLevelDialogEx (
    HWND         hParentWnd,
    CMN_ERROR    *lpErr)
```

参数

hParentWnd

对话框父窗口的句柄。

lpErr

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

0-999 授权级别

-1：对话框已通过“取消”(Dialog) 关闭。

所需文件

pwrt_api.h

pass_s.lib

useadmin.dll

示例

通过指定可能出现的错误在对话框中进行授权级别查询 (页 2362) "PWRTBunch.cpp"

参见

PWRTPermissionLevelDialog (页 2343)

通过指定可能出现的错误在对话框中进行授权级别查询 (页 2362)

函数概览 (页 2306)

3.5.6.11 PWRTPermissionToString

说明

确定与授权等级关联的描述。

声明

```
BOOL PWRTPermissionToString (
    LONG      perm,
    LPTSTR    string,
    int       bufsize)
```

参数

perm

要确定其描述的授权级别。

string

指向用于记录描述的缓冲区的指针。

bufsize

缓冲区的大小

返回值

TRUE

描述已成功传送。

FALSE

无法确定描述。

所需文件

pwrt_api.h

pass_s.lib

useadmin.dll

相关函数

PWRTPermissionLevelDialog (页 2343)	选择授权级别

示例

获取与许可编号关联的字符串 (页 2361)。“PWRTBunch.cpp”

参见

PWRTPermissionLevelDialog (页 2343)

获取与许可编号关联的字符串 (页 2361)

函数概览 (页 2306)

3.5.7 用于登录、注销的函数

3.5.7.1 PWRTGetCurrentUser

说明

确定当前登录用户的用户名。

声明

```
BOOL PWRTGetCurrentUser (
    LPTSTR    dest,
    int       bufsize )
```

参数

dest

用于接收用户名的数据缓冲区

bufsize

数据缓冲区的大小（字节）

返回值

TRUE

用户名已成功传送

FALSE

无用户登录

所需文件

pwrt_api.h

pass_s.lib

useadmin.dll

示例

返回当前用户的名称 (页 2363) "PWRTBunch.cpp"

参见

返回当前用户的名称 (页 2363)

函数概览 (页 2306)

3.5.7.2 PWRTGetLoginPriority

说明

不通过对话框进行的登录检查。

声明

```
LONG PWRTGetLoginPriority (
    )
```

参数

无

返回值

优先级：

LOGIN_STANDARD	（值：0)
LOGIN_CARD	（值：1)
LOGIN_KEYSWITCH	（值：2)

所需文件

pwrt_api.h

pass_s.lib

useadmin.dll

示例

查询当前登录优先级 (页 2364) "PWRTBunch.cpp"

参见

查询当前登录优先级 (页 2364)

函数概览 (页 2306)

3.5.7.3 PWRTIsLoggedInByCard

说明

确定用户是否通过智能卡登录。

声明

```
BOOL PWRTIsLoggedInByCard (
    )
```

参数

无

返回值

TRUE

用户已通过智能卡登录

FALSE

用户未通过智能卡登录，或用户未登录

所需文件

pwrt_api.h

pass_s.lib

useadmin.dll

示例

检查用户是否已通过智能卡登录 (页 2364) "PWRTBunch.cpp"

参见

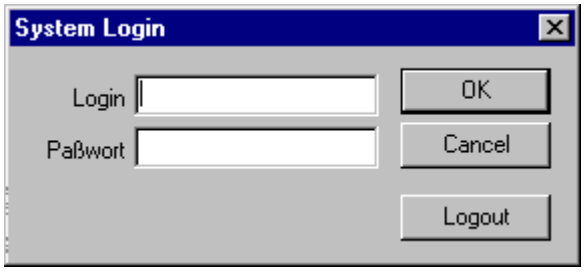
检查用户是否已通过智能卡登录 (页 2364)

函数概览 (页 2306)

3.5.7.4 PWRTLogin

说明

显示登录对话框，并在登录成功后将用户数据加载到共享内存中。



声明

```
BOOL PWRTLogin (
    TCHAR    monitor )
```

参数

monitor

显示对话框的监视器。该值不是数字值，以 TCHAR 格式指定，即“1”用于监视器 1。

返回值

TRUE

对话框已打开。该函数不会等待对话框关闭。

FALSE

对话框未打开

注释

除了手动输入登录数据之外，还可以使用智能卡登录。读卡器直接连接到 OS。如果函数执行时未连接有效智能卡，则登录对话框将自动消失。

所需文件

pwrt_api.h

pass_s.lib

useadmin.dll

相关函数

PWRTSilentLogin (页 2354)	登录

示例

PWRT 登录 - WinCC 自身提供的对话框 (页 2366) "PWRTBunch.cpp"

参见

PWRTSilentLogin (页 2354)

PWRT 登录 - WinCC 自身提供的对话框 (页 2366)

PWRTSilentLoginEx (页 2356)

函数概览 (页 2306)

3.5.7.5 PWRTLogout

说明

该函数会导致注销时标记在共享内存中被置位。

声明

```
BOOL PWRTLogout (
    )
```

参数

无

返回值

TRUE

注销已成功传递至 PassDBRT。

FALSE

注销遭到拒绝（例如 PassDBRT 不可用、ServiceMode 等）。

所需文件

pwrt_api.h

pass_s.lib

useadmin.dll

相关函数

PWRTLogoutEx (页 2353)	注销具有优先级的用户

示例

PWRT 注销 (页 2367) "PWRTBunch.cpp"

参见

PWRT 注销 (页 2367)

PWRTLogoutEx (页 2353)

函数概览 (页 2306)

3.5.7.6 PWRTLogoutEx

说明

除非指定的优先级过低，否则该函数会导致注销时标记在共享内存中被置位。

声明

```
BOOL PWRTLogoutEx (
        int    nLevel
        )
```

参数

nLevel

用户的优先级。

返回值

TRUE

注销已成功传递至 PassDBRT。

FALSE

注销遭到拒绝（例如 PassDBRT 不可用、ServiceMode 等）。

所需文件

pwrt_api.h

pass_s.lib

useadmin.dll

相关函数

PWRTLogout (页 2352)	用户注销

示例

通过优先级进行静默注销 (页 2368) "PWRTBunch.cpp"

参见

通过优先级进行静默注销 (页 2368)

PWRTLogout (页 2352)

函数概览 (页 2306)

3.5.7.7 PWRTSilentLogin

说明

与 PWRTLogin 相反，登录不会在对话框中进行。用户数据（用户名和密码）会直接传递到该函数。

声明

```
BOOL PWRTSilentLogin (
    LPCTSTR    login,
    LPCTSTR    password )
```

参数

login

用户的名称。

password

用户密码

返回值

TRUE

登录成功

FALSE

登录被拒绝

注释

如果函数执行时未连接有效智能卡，则会拒绝登录。

所需文件

pwrt_api.h

pass_s.lib

useadmin.dll

相关函数

PWRTLogin (页 2350)	通过对话框登录
PWRTSilentLoginEx (页 2356)	采用优先级登录

示例

不通过对话框登录 (页 2369) "PWRTBunch.cpp"

参见

PWRTLogin (页 2350)

PWRTSilentLoginEx (页 2356)

不通过对话框登录 (页 2369)

函数概览 (页 2306)

3.5.7.8 PWRTSilentLoginEx

说明

与 PWRTLogin 相反，登录不会在对话框中进行。用户数据（用户名、密码和优先级）会直接传递到该函数。

声明

```
BOOL PWRTSilentLoginEx (
    LPCTSTR    login,
    LPCTSTR    password,
    int        nLevel)
```

参数

login

用户的名称。

password

用户密码

nLevel

用户的优先级

返回值

TRUE

登录成功

FALSE

登录被拒绝

注释

如果函数执行时未连接有效智能卡，则会拒绝登录。

所需文件

pwrt_api.h

pass_s.lib

useadmin.dll

相关函数

PWRTLogin (页 2350)	通过对话框登录
PWRTSilentLogin (页 2354)	不通过对话框登录

示例

通过优先级进行静默登录 (页 2370) "PWRTBunch.cpp"

参见

PWRTSilentLogin (页 2354)

通过优先级进行静默登录 (页 2370)

PWRTLogin (页 2350)

函数概览 (页 2306)

3.5.8 示例

3.5.8.1 PWRT 检查许可

示例

```
//{{ODK_EXAMPLE}PWRT check permission. (USE)}
//{{FUNCTION}PWRTCheckPermission (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::Pwrtcheckpermission()
{
    BOOL bRet;
    CString csOut;
    CString csPermLevName;
    CGetText l_PermissionLevel("Insert the permission level number:");
    if(l_PermissionLevel.DoModal()==IDOK)
    {
        ////////////////////////////////////////////////////////////////////////////////
//////
        bRet = PWRTCheckPermission( l_PermissionLevel.m_lNumber, 0L ); // 0 = show the
message box
        ////////////////////////////////////////////////////////////////////////////////
//////
        ////////////////////////////////////////////////////////////////////////////////
//////
        bRet &= PWRTPermissionToString(l_PermissionLevel.m_lNumber,
csPermLevName.GetBuffer(1024), 1024);
        ////////////////////////////////////////////////////////////////////////////////
//////
        if(!bRet)
        {
            m_pView->Print("ERROR: ", FSIZE_FUNCMARK);
            m_pView->Print("PWRTCheckPermission.\n", FSIZE_PARAMMARK, FALSE, TRUE);
            csOut.Format("Level = %ld (\"%s\") - Access denied.\n",
            l_PermissionLevel.m_lNumber,
            csPermLevName);
            m_pView->Print(csOut, FSIZE_SUBMARK);
            m_pView->Print("\n");
            return;
        }
        else
        {
            csOut.Format("Level = %ld (\"%s\")\n",
            l_PermissionLevel.m_lNumber,
            csPermLevName);
            m_pView->Print("PWRTCheckPermission\n", FSIZE_FUNCMARK, TRUE);
            m_pView->Print("Access approved:\n", FSIZE_PARAMMARK, FALSE, TRUE);
            m_pView->Print(csOut, FSIZE_SUBMARK);
```

```
            m_pView->Print("\n");
        }
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

PWRTCheckPermission (页 2341)

3.5.8.2 检查画面的某等级的权限

示例

```
//{{ODK_EXAMPLE}Checks admission of a certain level for a picture. (USE)}
//{{FUNCTION}PWRTCheckPermissionOnPicture (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::Pwrtcheckpermissiononpicture()
{
    BOOL bRet;
    CMN_ERROR err;
    CString csOut;
    CGetText l_PermissionLevel("Enter the permission level:");
    CGetText l_PictureName("Enter the picture name:", FALSE);
    if(l_PermissionLevel.DoModal()==IDCANCEL)
    {
        return;
    }
    if(l_PictureName.DoModal()==IDCANCEL)
    {
        return;
    }
    ////////////////////////////////////////////////////////////////////////////////
//
    bRet=PWRTCheckPermissionOnPicture(l_PermissionLevel.m_lNumber, l_PictureName.m_csText,
0, &err);
    ////////////////////////////////////////////////////////////////////////////////
//
    if(bRet)
    {
        csOut.Format("Picture: \"%s\"\nLevel = %ld\n",
        l_PictureName.m_csText,
        l_PermissionLevel.m_lNumber);
        m_pView->Print("PWRTCheckPermissionOnPicture\n", FSIZE_FUNCMARK, TRUE);
        m_pView->Print("Access approved:\n", FSIZE_PARAMMARK, FALSE, TRUE);
        m_pView->Print(csOut, FSIZE_SUBMARK);
        m_pView->Print("\n");
    }
    else
    {
        m_pView->PrintError(&err, "PWRTCheckPermissionOnPicture");
        m_pView->PrintError("Access denied", "PWRTCheckPermissionOnPicture");
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

PWRTCheckPermissionOnPicture (页 2342)

3.5.8.3 获取与许可编号关联的字符串

概述

```
//{{ODK_EXAMPLE}Gets a string associated with the permission number. (USE)}
//{{FUNCTION}PWRTPermissionToString (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::Pwrtpermissiontostring()
{
    BOOL bRet;
    CString csOut;
    CString csPermLevName;
    CGetText l_PermissionLevel("Insert the permission level number:");
    if(l_PermissionLevel.DoModal()==IDOK)
    {
        ////////////////////////////////////////////////////////////////////////////////
//////
        bRet=PWRTPermissionToString(l_PermissionLevel.m_lNumber,
csPermLevName.GetBuffer(1024), 1024);
        ////////////////////////////////////////////////////////////////////////////////
//////
        if(!bRet)
        {
            m_pView->Print("ERROR: ", FSIZE_FUNCMARK);
            m_pView->Print("PWRTPermissionToString.\n", FSIZE_PARAMMARK, FALSE, TRUE);
            m_pView->Print("\n");
            return;
        }
        csOut.Format("PWRTPermissionToString( %ld, buffer )\n",
l_PermissionLevel.m_lNumber);
        m_pView->Print(csOut, FSIZE_FUNCMARK, TRUE);
        m_pView->Print("Permission level name:\n", FSIZE_PARAMMARK, FALSE, TRUE);
        csOut.Format("buffer = \"%s\"\n", csPermLevName);
        m_pView->Print(csOut, FSIZE_SUBMARK);
        m_pView->Print("\n");
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

PWRTPermissionToString (页 2345)

3.5.8.4 通过指定可能出现的错误在对话框中进行授权级别查询

示例

```
//{{ODK_EXAMPLE}Permission level query through a dialog with specifying a possible error.
(USE)}
//{{FUNCTION}PWRTPermissionLevelDialogEx (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::PWRTxPermissionLevelDialogEx()
{
    long lLevel;
    CMN_ERROR err;
    CString csOut;
    ////////////////////////////////////////////////////////////////////////////////
//
    lLevel=PWRTPermissionLevelDialogEx(*m_pView, &err);
    ////////////////////////////////////////////////////////////////////////////////
//
    if(m_pView->PrintError(&err, "PWRTPermissionLevelDialogEx"))
    {
       m_pView->Print("lLevel = PWRTPermissionLevelDialogEx(...)\n", FSIZE_FUNCMARK, TRUE);
        m_pView->Print("Permission level number chosen:\n", FSIZE_PARAMMARK, FALSE, TRUE);
        csOut.Format("lLevel = %ld\n", lLevel);
        m_pView->Print(csOut, FSIZE_SUBMARK);
        m_pView->Print("\n");
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

PWRTPermissionLevelDialog (页 2343)

PWRTPermissionLevelDialogEx (页 2344)

3.5.8.5 返回当前用户的名称

示例

```
//{{ODK_EXAMPLE}Return the name of the current user. (USE)}
//{{FUNCTION}PWRTGetCurrentUser (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::Pwrtgetcurrentuser()
{
    BOOL bRet;
    CString csUsername;
    CString csOut;
    ////////////////////////////////////////////////////////////////////////////////
//
    bRet=PWRTGetCurrentUser(csUsername.GetBuffer(1024), 1023);
    ////////////////////////////////////////////////////////////////////////////////
//
    if(!bRet)
    {
        m_pView->Print("ERROR: ", FSIZE_FUNCMARK, TRUE);
        m_pView->Print("PWRTGetCurrentUser.\n", FSIZE_PARAMMARK, FALSE, TRUE);
        m_pView->Print("Failed to get current user's name.\n", FSIZE_SUBMARK);
        m_pView->Print("\n");
        return;
    }
    m_pView->Print("PWRTGetCurrentUser\n", FSIZE_FUNCMARK, TRUE);
    m_pView->Print("Current user's name:\n", FSIZE_PARAMMARK, FALSE, TRUE);
    csOut.Format("Name = \"%s\"\n", csUsername);
    m_pView->Print(csOut, FSIZE_SUBMARK);
    m_pView->Print("\n");
}
//{{ODK_EXAMPLE}(END)}
```

参见

PWRTGetCurrentUser (页 2347)

3.5.8.6 查询当前登录优先级

示例

```
//{{ODK_EXAMPLE}Queries the current login priority. (USE)}
//{{FUNCTION}PWRTGetLoginPriority (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::Pwrtgetloginpriority()
{
    long lPriority;
    CString csOut;
    CMap<long, LONG, CString, CString> l_map(3);
    l_map[LOGIN_STANDARD]=CString("LOGIN_STANDARD");
    l_map[LOGIN_CARD]=CString("LOGIN_CARD");
    l_map[LOGIN_KEYSWITCH]=CString("LOGIN_KEYSWITCH");
    ////////////////////////////////////////////////////////////////////////////////
//
    lPriority=PWRTGetLoginPriority();
    ////////////////////////////////////////////////////////////////////////////////
//
    if(lPriority!=-1)
    {
        m_pView->Print("PWRTGetLoginPriority\n", FSIZE_FUNCMARK, TRUE);
        m_pView->Print("Priority returned:\n", FSIZE_PARAMMARK, FALSE, TRUE);
        csOut.Format("Priority level = %s\n", l_map[lPriority]);
        m_pView->Print(csOut, FSIZE_SUBMARK);
        m_pView->Print("\n");
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

PWRTGetLoginPriority (页 2348)

3.5.8.7 检查用户是否已通过智能卡登录

概述

```
//{{ODK_EXAMPLE}Checks for the user has been logged on by card. (USE)}
//{{FUNCTION}PWRTIsLoggedInByCard (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::Pwrtisloggedinbycard()
{
    BOOL bRet;
    CString csOut;
    ////////////////////////////////////////////////////////////////////////////////
//
    bRet=PWRTIsLoggedInByCard();
    ////////////////////////////////////////////////////////////////////////////////
//
    if(bRet)
    {
        m_pView->Print("PWRTIsLoggedInByCard\n", FSIZE_FUNCMARK, TRUE);
       m_pView->Print("An user show-up logged in by card.\n", FSIZE_PARAMMARK, FALSE, TRUE);
    }
    else
    {
        m_pView->PrintError(_T("Either no user logged-on-by-card or no user logged on at
all."),
        _T("PWRTIsLoggedInByCard()"));
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

PWRTIsLoggedInByCard (页 2349)

3.5.8.8 PWRT 登录 - WinCC 自身提供的对话框

示例

```
//{{ODK_EXAMPLE} PWRT login - dialog provided by WinCC intself. (USE)}
//{{FUNCTION}PWRTLogin (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::PWRTxLogin()
{
    BOOL bRet;
    ////////////////////////////////////////////////////////////////////////////////
//
    bRet=PWRTLogin(_T('1')); // it always appears on the first screen
    ////////////////////////////////////////////////////////////////////////////////
//
    if(!bRet)
    {
        m_pView->Print("ERROR: ", FSIZE_FUNCMARK, TRUE);
        m_pView->Print("PWRTLogin.\n", FSIZE_PARAMMARK, FALSE, TRUE);
        m_pView->Print("General function failure.\n", FSIZE_SUBMARK);
        m_pView->Print("\n");
        return;
    }
    m_pView->Print("PWRTLogin\n", FSIZE_FUNCMARK, TRUE);
    m_pView->Print("Logging on...\n", FSIZE_PARAMMARK, FALSE, TRUE);
    m_pView->Print("\n");
}
//{{ODK_EXAMPLE}(END)}
```

参见

PWRTLogin (页 2350)

3.5.8.9 PWRT 注销

示例

```
//{{ODK_EXAMPLE}PWRT logout. (USE)}
//{{FUNCTION}PWRTLogout (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::Pwrtlogout()
{
    BOOL bRet;
    ////////////////////////////////////////////////////////////////////////////////
//
    bRet=PWRTLogout(); // it appears on the first screen
    ////////////////////////////////////////////////////////////////////////////////
//
    if(!bRet)
    {
        m_pView->Print("ERROR: ", FSIZE_FUNCMARK, TRUE);
        m_pView->Print("PWRTLogout.\n", FSIZE_PARAMMARK, FALSE, TRUE);
        m_pView->Print("General function failure.\n", FSIZE_SUBMARK);
        m_pView->Print("\n");
        return;
    }
    m_pView->Print("PWRTLogout\n", FSIZE_FUNCMARK, TRUE);
    m_pView->Print("Logging out...\n", FSIZE_PARAMMARK, FALSE, TRUE);
    m_pView->Print("\n");
}
//{{ODK_EXAMPLE}(END)}
```

参见

PWRTLogout (页 2352)

3.5.8.10 通过优先级进行静默注销

示例

```
//{{ODK_EXAMPLE}Silent logout with priority level. (USE)}
//{{FUNCTION}PWRTLogoutEx (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::PWRTxLogoutEx()
{
    BOOL bRet;
    CString csOut;
    CGetText l_PriorityLevel("The priority level:");
    if(l_PriorityLevel.DoModal()==IDCANCEL)
    {
        return;
    }
    ////////////////////////////////////////////////////////////////////////////////
//
    bRet=PWRTLogoutEx( l_PriorityLevel.m_lNumber);
    ////////////////////////////////////////////////////////////////////////////////
//
    if(!bRet)
    {
        m_pView->PrintError( _T("Cannot logout."),
        _T("PWRTLogoutEx"));
    }
    else
    {
        m_pView->Print("PWRTLogoutEx\n", FSIZE_FUNCMARK, TRUE);
        m_pView->Print("Logging out...\n", FSIZE_PARAMMARK, FALSE, TRUE);
        csOut.Format("Priority level = %ld\n", l_PriorityLevel.m_lNumber);
        m_pView->Print(csOut, FSIZE_SUBMARK);
        m_pView->Print("\n");
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

PWRTLogoutEx (页 2353)

3.5.8.11 不通过对话框登录

示例

```
//{{ODK_EXAMPLE}Logs on without using a dialog. (USE)}
//{{FUNCTION}PWRTSilentLogin (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::Pwrtsilentlogin()
{
    BOOL bRet;
    CPassdlg l_Password("Enter name and password:");
    if(l_Password.DoModal() == IDCANCEL)
    {
        return;
    }
    ////////////////////////////////////////////////////////////////////////////////
//
    bRet=PWRTSilentLogin(l_Password.m_csUsername,
    l_Password.m_csPassword);
    ////////////////////////////////////////////////////////////////////////////////
//
    if(!bRet)
    {
        m_pView->PrintError( _T("Access denied or no Runtime project open."),
        _T("PWRTSilentLogin"));
    }
    else
    {
        m_pView->Print("PWRTSilentLogin\n", FSIZE_FUNCMARK, TRUE);
        m_pView->Print("Access approved:\n", FSIZE_PARAMMARK, FALSE, TRUE);
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

PWRTSilentLogin (页 2354)

3.5.8.12 通过优先级进行静默登录

示例

```
//{{ODK_EXAMPLE}Silent login with priority level. (USE)}
//{{FUNCTION}PWRTSilentLoginEx (USE)}
//{{FUNCTION}(END)}
void CPWRTBunch::Pwrtsilentloginex()
{
    BOOL bRet;
    CString csOut;
    CPassdlg l_Password("Enter name and password:");
    CGetText l_PriorityLevel("Enter priority level:");
    if(l_Password.DoModal()==IDCANCEL)
    {
        return;
    }
    if(l_PriorityLevel.DoModal()==IDCANCEL)
    {
        return;
    }
    ////////////////////////////////////////////////////////////////////////////////
//
    bRet=PWRTSilentLoginEx(l_Password.m_csUsername, l_Password.m_csPassword,
l_PriorityLevel.m_lNumber);
    ////////////////////////////////////////////////////////////////////////////////
//
    if(!bRet)
    {
        m_pView->PrintError( _T("Access denied or no Runtime project open."),
        _T("PWRTSilentLogin"));
    }
    else
    {
        m_pView->Print("PWRTSilentLogin\n", FSIZE_FUNCMARK, TRUE);
        m_pView->Print("Access approved:\n", FSIZE_PARAMMARK, FALSE, TRUE);
        csOut.Format("Priority level = %ld\n", l_PriorityLevel.m_lNumber);
        m_pView->Print(csOut, FSIZE_SUBMARK);
        m_pView->Print("\n");
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

PWRTSilentLoginEx (页 2356)

3.6 测试系统的函数

3.6.1 基本知识

3.6.1.1 函数概览

声明

TXT_ENUM_INFOTEXTS_PROC (页 2385)	列出信息文本（回调）
TXT_ENUM_LANGUAGES_PROC (页 2393)	列出已组态语言（回调）
TXTCloseProject (页 2375)	关闭文本库
TXTEnumInfoText (页 2383)	列出信息文本
TXTEnumLanguages (页 2391)	列出已组态语言
TXTGetFont (页 2395)	确定语言的字符集
TXTGetMaxTextID (页 2377)	确定文本 ID（最大）
TXTOpenProject (页 2378)	打开文本库
TXTRTConnect（页 2380)	与文本服务器保持连接
TXTRTDisconnect (页 2381)	与文本服务器的连接释放
TXTRTGetInfoText (页 2388)	读取信息文本
TXTRTGetInfoTextMC (页 2389)	读取文本服务器中的信息文本
TXTRTGetLanguageID (页 2399)	确定已激活语言
TXTRTSetLanguage (页 2400)	对错误消息文本进行语言转换
TXTShowLanguagesDialog (页 2397)	显示语言（已安装）
TXTUpdateRuntime (页 2386)	更新运行系统中的信息文本

3.6.1.2 错误消息

概述

通过 API 函数的 CMN_ERROR 错误结构可返回以下错误消息：

TXT_SYS_ERROR	（值： 0x10000000）	如果在错误结构 CMN_ERROR 的 dwError1 中设置该位，则 dwError2 包含系统错误代码。
TXT_OK	（值： 0x00000000）	未发生错误。
TXT_NO_CONNECT	（值： 0x00000001）	数据库尚未打开。
TXT_CONNECT	（值： 0x00000002）	连接到数据库时出错。
TXT_UPDATE	（值： 0x00000003）	写入到数据库时出错。
TXT_CREATE_KEY	（值： 0x00000004）	无法创建 TextID。
TXT_KEY_NOT_FOUND	（值： 0x00000005）	无法找到指定的 TextID。
TXT_LANGUAGE_NOT_FOUND	（值： 0x00000006）	无法找到指定的语言。
TXT_NO_NEW_RECORD	（值： 0x00000007）	无法创建新纪录（AddRecord 生成错误）
TXT_FONT_NOT_FOUND	（值： 0x00000008）	未找到字体
TXT_TABLE_NOT_FOUND	（值： 0x00000009）	未找到数据库表。 dwError2 字段包含更多详细信息：
TXTVERSIONINFO	（值： 0x00000001）	未找到版本信息表。
TXTLANGUAGES	（值： 0x00000002）	未找到语言表。
TXTTABLE	（值： 0x00000003）	未找到文本表。
TXT_WRONG_PROJECT	（值： 0x00000010）	项目名称错误
TXT_OPEN	（值： 0x00000011）	无法加载数据库。
TXT_PARAM	（值： 0x00000012）	无效参数
TXT_DISCONNECT	（值： 0x00000013）	注销数据库时出错
TXT_CALLBACK	（值： 0x00000014）	回调返回 FALSE
TXT_ALREADY_FOUND	（值： 0x00000015）	文本已存在
TXT_CREATE	（值： 0x00000016）	无法创建数据库表。
TXT_UPDATE_RUNTIME	（值： 0x00000017）	未找到运行系统，或 SendMessage 返回 NULL。

TXT_ALREADY_EXIST	（值： 0x00000018）	语言已存在
TXT_NO_LANGUAGE	（值： 0x00000019）	指定的语言无效
TXT_TEXT_LIBRARY_RUNNING	（值： 0x00000020）	TEXTBIB.EXE 已运行；不允许打开
TXT_TABLES_ALREADY_EXIST	（值： 0x00000021）	数据库表已存在。 不再允许创建。

文本库 RT

TXT_RT_OK	（值： 0x00000000）	无错误
TXT_RT_OPEN_MMF	（值： 0x00000001）	打开内存映射文件时出错
TXT_RT_CREATE_SERVICE_WINDOW	（值： 0x00000002）	创建服务窗口时出错
TXT_RT_KEY_NOT_FOUND	（值： 0x00000003）	未找到 TextID
TXT_RT_NO_READ_ACCESS	（值： 0x00000004）	对内存映射文件的读访问被拒绝
TXT_RT_NO_LANGUAGE_FOUND	（值： 0x00000005）	未找到任何语言
TXT_RT_LANGUAGE_NOT_FOUND	（值： 0x00000006）	未找到语言
TXT_RT_INTERNAL_ERROR	（值： 0x00000007）	内部故障
TXT_RT_INVALIDPARAM	（值： 0x00000008）	参数错误
TXT_RT_NODEFAULTSERVER	（值： 0x00000011）	未组态默认服务器。
TXT_RT_NOLOCALSERVER	（值： 0x00000012）	无可用本地服务器。
TXT_RT_NOSERVER	（值： 0x00000013）	未组态默认服务器，无可用本地服务器。
TXT_RT_NOMC	（值： 0x00000014）	该项目不是多客户端项目（不用于此处）
TXT_RT_NOMCDEFAULTSERVER	（值： 0x00000015）	该项目不是多客户端项目，但给出“@default”作为服务器前缀。

3.6.1.3 语言代码

文本系统的语言代码

WinCC 中只支持 Windows 的 SUBLANG_DEFAULT 语言。 因此，可以将以下值分配给 dwLocaleID 参数：

LANG_ARABIC	0x0401
LANG_AFRIKAANS	0x0436
LANG_ALBANIAN	0x041C
LANG_BASQUE	0x042D
LANG_BULGARIAN	0x0402
LANG_BELARUSIAN	0x0423
LANG_CATALAN	0x0403
LANG_CHINESE	0x0804
LANG_CROATIAN	0x041A
LANG_CZECH	0x0405
LANG_DANISH	0x0406
LANG_DUTCH	0x0413
LANG_ENGLISH	0x0409
LANG_ESTONIAN	0x0425
LANG_FAEROESE	0x0438
LANG_FARSI	0x0429
LANG_FINNISH	0x040B
LANG_FRENCH	0x040C
LANG_GERMAN	0x0407
LANG_GREEK	0x0408
LANG_HEBREW	0x040D
LANG_HUNGARIAN	0x040E
LANG_ICELANDIC	0x040F
LANG_INDONESIAN	0x0421
LANG_ITALIAN	0x0410
LANG_JAPANESE	0x0411
LANG_KOREAN	0x0412

LANG_LATVIAN	0x0426
LANG_LITHUANIAN	0x0427
LANG_NORWEGIAN	0x0414
LANG_POLISH	0x0415
LANG_PORTUGUESE	0x0416
LANG_ROMANIAN	0x0418
LANG_RUSSIAN	0x0419
LANG_SLOVAK	0x041B
LANG_SLOVENIAN	0x0424
LANG_SPANISH	0x040A
LANG_SWEDISH	0x041D
LANG_THAI	0x041E
LANG_TURKISH	0x041F
LANG_UKRAINIAN	0x0422

此处，位于高位的 4 个字节 (0x04..) 指定语言为 SUBLANG_DEFAULT 语言

3.6.2 常规函数

3.6.2.1 TXTCloseProject

说明

成功执行后关闭项目数据库。

声明

```
BOOL TXTCloseProject (
    LPCTSTR        lpszProjectFile,
    LPCMN_ERROR    lpdmError );
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

文本库已关闭。

FALSE

错误

注释

每次调用时，TXTOpenProject 函数都会增大内部参考计数器的值。调用 TXTCloseProject 时，该计数器的值会减小。仅当参考计数器的值返回 0 时，才会关闭文本库。

因此，必须调用相同次数的 TXTCloseProject，否则，应用程序关闭时会出现异常错误。

说明

调用不可用于应用程序的析构函数（EXE、DLL、OCX 等）中。由于 Microsoft 的特定机制，这可能导致调用冻结程序，从而导致程序崩溃。

错误消息

TXT_DISCONNECT	注销数据库时出错

所需文件

text_cs.h

text_cs.lib

text_cs.dll

相关函数

TXTOpenProject (页 2378)	打开文本库

示例

获取信息文本 (页 2402) "TX01.cpp"

枚举信息文本 (页 2405) "TX01.cpp"

参见

TXTOpenProject (页 2378)

获取信息文本 (页 2402)

枚举信息文本 (页 2405)

函数概览 (页 2371)

3.6.2.2 TXTGetMaxTextID

说明

记录 DLL 中保存的最高文本 ID。

声明

```
BOOL TXTGetMaxTextID (
    LPLONG          lplMaxTextID,
    LPCMN_ERROR     lpdmError );
```

参数

lplMaxTextID

指向应保存文本 ID 的缓冲区的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

最大文本 ID 已确定

FALSE

错误

所需文件

text_cs.h

text_cs.lib

text_cs.dll

参见

函数概览 (页 2371)

3.6.2.3 TXTOpenProject

说明

成功执行后打开数据库。

声明

```
BOOL TXTOpenProject (
    LPCTSTR        lpszProjectFile,
    LPCTSTR        lpszDSNName,
    BOOL           fExclusive,
    LPCMN_ERROR    lpdmError );
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

lpszDSNName

指向数据源名称的指针。

fExclusive

如果为 TRUE，则 TEXTBIB.EXE 无法启动，这可能是因为其它程序正在访问文本库。如果 TEXTBIB.EXE 已启动，则会拒绝调用。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

数据库打开

FALSE

错误

注释

如果无可用文本库表，则会生成新的文本库表。

每次调用函数时，参考计数器的值都会增大。必须出现与 TXTCloseProject 相同的数值才可以关闭。否则可能会在应用程序关闭时出现异常错误。

错误消息

TXT_CONNECT	连接到数据库时出错。
TXT_TABLE_NOT_FOUND	未找到数据库表。dwError2 字段包含更多详细信息：
TXT_TEXT_LIBRARY_RUNNING	TEXTBIB.EXE 已启动
TXT_WRONG_PROJECT	项目名称错误

所需文件

text_cs.h

text_cs.lib

text_cs.dll

相关函数

TXTCloseProject (页 2375)	关闭文本库

示例

获取信息文本 (页 2402) "TX01.cpp"

枚举信息文本 (页 2405) "TX01.cpp"

参见

TXTCloseProject (页 2375)

获取信息文本 (页 2402)

枚举信息文本 (页 2405)

函数概览 (页 2371)

3.6.2.4 TXTRTConnect

说明

与文本服务器建立连接。该函数可加速执行 TXTRT 函数。之后就不会在每次调用时都与文本服务器进行连接，并且不会在结束时再次关闭。

声明

```
BOOL TXTRTConnect ( );
```

参数

无

返回值

TRUE

与文本服务器的连接已建立

FALSE

错误

注释

关闭应用程序之前，必须始终使用 TXTRTDisconnect 关闭连接，以免系统中出现后续故障。

所需文件

text_rt.h

text_rt.lib

text_rt.dll

相关函数

TXTRTDisconnect (页 2381)	与文本服务器的连接释放

参见

TXTRTDisconnect (页 2381)

函数概览 (页 2371)

3.6.2.5 TXTRTDisconnect

说明

关闭通过 TXTRTConnect 与文本服务器建立的连接。

声明

```
BOOL TXTRTDisconnect ( );
```

参数

无

返回值

TRUE

与文本服务器的连接已建立

注释

该函数会停止 TXTRT 函数提供的加速功能，并始终返回 TRUE。

之后，后续调用的 TXTRT 函数会在每次调用时与文本服务器建立新连接，然后关闭该连接。

如果使用 TXTRTConnect，并且在关闭应用程序前未执行 TXTRTDisconnect，则之后系统中可能出现故障。

说明

调用不可用于应用程序的析构函数（EXE、DLL、OCX 等）中。由于 Microsoft 的特定机制，这可能导致调用冻结程序，从而导致程序崩溃。

所需文件

text_rt.h

text_rt.lib

text_rt.dll

相关函数

TXTRTConnect（页 2380）	与文本服务器保持连接

参见

TXTRTConnect（页 2380）

函数概览 (页 2371)

3.6.3 用于处理信息文本的函数

3.6.3.1 TXTEnumInfoText

说明

列出满足 lpszFilter 中过滤标准的所有信息文本。

声明

```
BOOL TXTEnumInfoText (
    LPCTSTR                     lpszProjectFile,
    DWORD                       dwLocale,
    LPDWORD                     lpdwItems,
    LPCTSTR                     lpszFilter,
    TXT_ENUM_INFOTEXTS_PROC     lpfnEnum,
    LPVOID                      lpvUser,
    LPCMN_ERROR                 lpdmError );
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

dwLocale

语言的语言代码，应枚举其信息文本。

lpdwItems

指向应用程序的双字的指针，双字包含调用后枚举的信息文本的数目。

lpszFilter

指向 LIKE 运算符的 SQL 语句条件的指针。

lpfnEnum

用于接收信息文本的回调函数。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

信息文本已列出

FALSE

错误

注释

如果 lpszFilter == NULL 或传递的是空字符串，则会枚举语言的所有信息文本。还可以通过 lpdwItems 枚举并包含空的、未分配的信息文本。

错误消息

TXT_CALLBACK	回调返回 FALSE
TXT_LANGUAGE_NOT_FOUND	无法找到指定的语言。
TXT_NO_CONNECT	数据库尚未打开。
TXT_NO_LANGUAGE	指定的语言无效
TXT_WRONG_PROJECT	项目名称错误

所需文件

text_cs.h

text_cs.lib

text_cs.dll

相关函数

TXT_ENUM_INFOTEXTS_PROC (页 2385)	列出信息文本（回调）

示例

枚举信息文本 (页 2405) "TX01.cpp"

参见

TXT_ENUM_INFOTEXTS_PROC (页 2385)

枚举信息文本 (页 2405)

函数概览 (页 2371)

3.6.3.2 TXT_ENUM_INFOTEXTS_PROC

说明

为了评估列出的工具提示，必须提供 TXT_ENUM_INFOTEXTS_PROC 类型的回调函数。

声明

```
BOOL ( * TXT_ENUM_INFOTEXTS_PROC) (
    DWORD      dwTextID,
    LPCTSTR    lpszInfoText,
    LPVOID     lpvUser );
```

参数

dwTextID

lpszInfotext 引用的文本的 ID。

lpszInfoText

指向通过调用函数传递的工具提示的指针。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举

FALSE

取消枚举

注释

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

text_cs.h

text_cs.lib

text_cs.dll

相关函数

TXTEnumInfoText (页 2383)	列出信息文本

示例

枚举信息文本 (页 2405) "TX01.cpp"

参见

TXTEnumInfoText (页 2383)

枚举信息文本 (页 2405)

函数概览 (页 2371)

3.6.3.3 TXTUpdateRuntime

说明

文本库中的文本重新加载到运行系统中。

声明

```
BOOL TXTUpdateRuntime (
    LPCTSTR        lpszProjectFile,
    HWND           hwndParent,
    LPCMN_ERROR    lpdmError );
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

hwndParent

父窗口的句柄

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

信息文本已更新

注释

未更新文本引用变量。该函数始终会返回 TRUE。

所需文件

text_cs.h

text_cs.lib

text_cs.dll

参见

函数概览 (页 2371)

3.6.3.4 TXTRTGetInfoText

说明

成功执行之后，该函数会通过 dwTextID 描述的文本填充传递给 lpszBuffer 的缓冲区。如果传递的缓冲区比文本长度（缓冲区长度，pdwSize 中的字符数）小，则会相应截断文本。返回值仍为 TRUE。如果 NULL 传入 lpszBuffer，则函数会确定所需缓冲区大小并将其存储在 pdwSize 中。

声明

```
BOOL TXTRTGetInfoText (
    DWORD          dwTextID,
    LPTSTR         lpszBuffer,
    LPDWORD        pdwSize,
    LPCMN_ERROR    lpdmError );
```

参数

dwTextID

要读取的文本 ID

lpszBuffer

指向保存信息文本的缓冲区的指针。

pdwSize

指向包含缓冲区大小的 DWORD 的指针

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

信息文本已确定

FALSE

错误

注释

错误消息

TXT_RT_OPEN_MMF	打开内存映射文件时出错
TXT_RT_CREATE_SERVICE_WINDOW	创建服务窗口时出错
TXT_KEY_NOT_FOUND	无法找到指定的 TextID。
TXT_RT_NO_READ_ACCESS	对内存映射文件的读访问被拒绝

所需文件

text_rt.h

text_rt.lib

text_rt.dll

相关函数

AUTOHOTSPOT	读取文本服务器中的工具提示

示例

获取信息文本 (页 2402) "TX01.cpp"

参见

获取信息文本 (页 2402)

函数概览 (页 2371)

3.6.3.5 TXTRTGetInfoTextMC

说明

从指定文本服务器或 C 语言文本列表获取文本（如工具提示）。成功执行之后，该函数会通过 dwTextID 描述的文本填充传递给 lpszBuffer 的缓冲区。如果传递的缓冲区比文本长度（缓冲区长度，pdwSize 中的字符数）小，则会相应截断文本。返回值仍为 TRUE。如果 NULL 传入 lpszBuffer，则函数会确定所需缓冲区大小并将其存储在 pdwSize 中。

声明

```
BOOL TXTRTGetInfoTextMC (
    DWORD            dwTextID,
    LPTSTR            lpszBuffer,
    LPDWORD            pdwSize,
     LPTSTR            lpszServer,
    LPCMN_ERROR        lpdmError );
```

参数

dwTextID

要读取的文本 ID

lpszBuffer

指向保存信息文本的缓冲区的指针。

pdwSize

指向包含缓冲区大小的 DWORD 的指针

lpszServer

指向文本服务器符号名称（不带服务器定界符 ::）的指针

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

信息文本已确定

FALSE

错误

注释

该函数只用于多客户端项目

错误消息

TXT_RT_OPEN_MMF	打开内存映射文件时出错
TXT_RT_CREATE_SERVICE_WINDOW	创建服务窗口时出错
TXT_KEY_NOT_FOUND	无法找到指定的 TextID。
TXT_RT_NO_READ_ACCESS	对内存映射文件的读访问被拒绝
TXT_RT_NODEFAULTSERVER	未组态默认服务器。
TXT_RT_NOLOCALSERVER	无可用本地服务器。
TXT_RT_NOSERVER	未组态默认服务器，且无可用本地服务器。
TXT_RT_NOMCDEFAULTSERVER	该项目不是多客户端项目，但给出"@default"作为服务器前缀。

所需文件

text_rt.h

text_rt.lib

text_rt.dll

相关函数

AUTOHOTSPOT	读取工具提示

参见

函数概览 (页 2371)

3.6.4 用于处理语言的函数

3.6.4.1 TXTEnumLanguages

说明

为所有已组态语言调用传递的回调函数。

声明

```
BOOL TXTEnumLanguages (
    LPCTSTR                     lpszProjectFile,
    LPDWORD                     lpdwItems,
    TXT_ENUM_LANGUAGES_PROC     lpfnEnum,
    LPVOID                      lpvUser,
    LPCMN_ERROR                 lpdmError );
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

lpdwItems

指向应用程序的双字的指针，双字包含调用后枚举的语言数目。

lpfnEnum

用于接收已组态语言的回调函数。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

组态语言已列出

FALSE

错误

错误消息

TXT_CALLBACK	回调返回 FALSE
TXT_NO_CONNECT	数据库尚未打开。
TXT_WRONG_PROJECT	项目名称错误

所需文件

text_cs.h

text_cs.lib

text_cs.dll

相关函数

TXT_ENUM_LANGUAGES_PROC (页 2393)	列出已组态语言（回调）

示例

枚举信息文本 (页 2405) "TX01.cpp"

参见

TXT_ENUM_LANGUAGES_PROC (页 2393)

枚举信息文本 (页 2405)

函数概览 (页 2371)

3.6.4.2 TXT_ENUM_LANGUAGES_PROC

说明

为了评估列出的语言，必须提供 TXT_ENUM_LANGUAGES_PROC 类型的回调函数。

声明

```
BOOL ( * TXT_ENUM_LANGUAGES_PROC) (
    DWORD      dwLocaleID,
    LPCTSTR    lpszName,
    LPVOID     lpvUser );
```

参数

dwLocaleID

语言代码

lpszName

指向语言名称的指针。

lpvUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

注释

说明

仅应在这里复制数据（如可能）。以下回调中的函数调用类型可导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

text_cs.h

相关函数

TXTEnumLanguages (页 2391)	列出已组态语言

示例

枚举信息文本 (页 2405) "TX01.cpp"

参见

TXTEnumLanguages (页 2391)

枚举信息文本 (页 2405)

函数概览 (页 2371)

3.6.4.3 TXTGetFont

说明

成功执行后，会使用所需语言的字体集填充传入 lplf 的 LOGFONT 结构。

声明

```
BOOL TXTGetFont (
    LPCTSTR         lpszProjectFile,
    DWORD           dwLocale,
    LPLOGFONT       lplf,
    LPCMN_ERROR     lpdmError );
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

dwLocale

语言的语言编码，应获取其字体

lplf

指向具有字符集相关信息的 Windows 特定 LOGFONT 类型结构的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

字符集已确定

FALSE

错误

错误消息

TXT_NO_CONNECT	数据库尚未打开。
TXT_NO_LANGUAGE	指定的语言无效
TXT_LANGUAGE_NOT_FOUND	无法找到指定的语言。
TXT_WRONG_PROJECT	项目名称错误

所需文件

text_cs.h

text_cs.lib

text_cs.dll

参见

函数概览 (页 2371)

3.6.4.4 TXTShowLanguagesDialog

说明

打开显示可用语言的对话框。如果单击“确定”(OK) 关闭该对话框，则所选语言会存储在 lpdwLocale 中。此外，相应字体会传入 LOGFONT 结构。

声明

```
BOOL TXTShowLanguagesDialog (
    LPCTSTR                lpszProjectFile,
    HWND                 hwndParent,
    LPDWORD                lpdwLocale,
    LPLOGFONT             lplf,
    LPCMN_ERROR            lpdmError );
```

参数

lpszProjectFile

指向项目文件名称（包括路径和扩展名）的指针。

hwndParent

对话框父窗口的句柄。该参数默认设置为 NULL。

dwLocale

指向要安装语言的代码的指针。

lplf

指向具有字符集相关信息的 Windows 特定 LOGFONT 类型结构的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

语言已选择

FALSE

出错或单击“取消”(Cancel) 关闭对话框

注释

为了提供对话框中安装的语言，可以通过该函数枚举数据库中的语言。如果出现错误，则 TXT_CALLBACK 错误消息会存储在错误结构中。

错误消息

TXT_CALLBACK	回调返回 FALSE
TXT_NO_CONNECT	数据库尚未打开。
TXT_WRONG_PROJECT	项目名称错误

所需文件

text_cs.h

text_cs.lib

text_cs.dll

参见

函数概览 (页 2371)

3.6.4.5 TXTRTGetLanguageID

说明

成功执行后，会使用内存映射文件中加载的原始语言 ID 填充传入 lpchLanguageID 的缓冲区。

声明

```
BOOL TXTRTGetLanguageID (
    LPBYTE          lpchLanguageID,
    LPCMN_ERROR     lpdmError );
```

参数

lpchLanguageID

指向存储原始语言 ID 的缓冲区的指针。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

激活的语言已确定

FALSE

错误

错误消息

TXT_RT_OPEN_MMF	打开内存映射文件时出错
TXT_RT_CREATE_SERVICE_WINDOW	创建服务窗口时出错
TXT_RT_NO_LANGUAGE_FOUND	未找到任何语言

所需文件

text_rt.h

text_rt.lib

text_rt.dll

参见

函数概览 (页 2371)

3.6.4.6 TXTRTSetLanguage

说明

设置错误消息的语言。如果该语言不可用，则会设置默认语言（德语），并会返回 FALSE。然后不再支持该函数，并始终会返回值 TRUE。

声明

```
BOOL TXTRTSetLanguage (
    DWORD          dwLocaleID,
    LPCMN_ERROR    lpdmError );
```

参数

dwLocaleID

要设置语言的语言代码。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

错误消息的语言已转换

所需文件

text_rt.h

text_rt.lib

text_rt.dll

参见

函数概览 (页 2371)

3.6.5 示例

3.6.5.1 获取信息文本

示例

```
//{{ODK_EXAMPLE}Insert new Infotext (TXT)}
//{{FUNCTION}TXTOpenProject (TXT)}
//{{FUNCTION}TXTNewInfoText (TXT)}
//{{FUNCTION}TXTSetInfoText (TXT)}
//{{FUNCTION}TXTCloseProject (TXT)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: MyTxtSetInfotext(void) ODK DM CS
// ========================================================================
// Abst.   :    Insert New Text
// ========================================================================
void MyTxtNewInfotext(void)
{
    TCHAR szText[255];
    CMN_ERROR Error;
    BOOL ret = FALSE;
    DWORD dwLocale = 0x0407; // german
    TCHAR szFilter[]="S%"; // all texts with 'S'
    DWORD dwSize =254;
    DWORD dwTextID = 0;
    ret = MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile and
g_szDSNName
    if(TRUE == ret)
    {
        memset(&Error, 0, sizeof(CMN_ERROR));
        ret = TXTOpenProject(g_szProjectFile, g_szDSNName, FALSE, &Error);
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TXTOpenProject:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
            ODKTrace(szText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TXTOpenProject"));
            ODKTrace(szText);
            dwTextID = 380;
            dwLocale = 0x0407; // german
            _tcsncpy_s(szText, _countof(szText), _T("NewInfoText_ODK"), _TRUNCATE);
            memset(&Error, 0, sizeof(CMN_ERROR));
```

```
            ret = TXTGetInfoText(g_szProjectFile, dwTextID, dwLocale, szText, &dwSize,
&Error);
            if(FALSE == ret)
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TXTNewInfoText: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                    Error.dwError1, Error.dwError2, Error.szErrorText);
            }
            else
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TXTNewInfoText"));
            }
            ODKTrace(szText);
            dwLocale=0x0409; // english
            //dwTextID = 380; // use ID from TXTNewInfoText
            _tcsncpy_s(szText, _countof(szText), _T("InfoText_ODK"), _TRUNCATE);
            memset(&Error, 0, sizeof(CMN_ERROR));
           ret = TXTSetInfoText(g_szProjectFile, NULL, dwTextID, dwLocale, szText, &Error);
            if(FALSE == ret)
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TXTSetInfoText: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                    Error.dwError1, Error.dwError2, Error.szErrorText);
            }
            else
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TXTSetInfoText"));
            }
            ODKTrace(szText);
            memset(&Error, 0, sizeof(CMN_ERROR));
            ret = TXTCloseProject(g_szProjectFile, &Error);
            if(FALSE == ret)
            {
                 _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TXTCloseProject: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                    Error.dwError1, Error.dwError2, Error.szErrorText);
            }
            else
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TXTCloseProject"));

                ODKTrace(szText);
            }
        }
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

TXTCloseProject (页 2375)

TXTOpenProject (页 2378)

TXTRTGetInfoText (页 2388)

3.6.5.2 枚举信息文本

示例

```
//{{ODK_EXAMPLE}Enumerate texts (TXT)}
//{{FUNCTION}TXTOpenProject (TXT)}
//{{FUNCTION}TXTCloseProject (TXT)}
//{{FUNCTION}TXTEnumLanguages (TXT)}
//{{FUNCTION}TXT_ENUM_LANGUAGES_PROC (TXT)}
//{{FUNCTION}TXTEnumInfoText (TXT)}
//{{FUNCTION}TXT_ENUM_INFOTEXTS_PROC (TXT)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: MyTxtEnums(void) ODK DM CS
// ========================================================================
// Abst. : Enumerate Texts
// ========================================================================

BOOL MyTXTEnumLanguagesCallback (DWORD dwLocID, LPCTSTR lpszName, LPVOID lpvUser)
{
    lpvUser;
    TCHAR szText[255];
    BOOL ret = TRUE;
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   InfoLang: ID=%x ;
    ODKTrace(szText);
    return(ret);
}

BOOL MyTXTEnumInfotextsCallback(DWORD dwTextID, LPCTSTR lpszInfoText, LPVOID lpvUser)
{
    lpvUser;
    TCHAR szText[255];
    BOOL ret = TRUE;
     _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   InfoText: ID=%.4d ; Tx=%s "),
dwTextID, lpszInfoText);
    ODKTrace(szText);
    return(ret);
}

void MyTxtEnums(void)
{
    TCHAR szText[255];
    CMN_ERROR Error;
    BOOL ret = FALSE;
    DWORD dwLocale = 0x0407; // german
    DWORD dwItems = 0;
    DWORD dwUser = 0;
    TCHAR szFilter[]="S%"; // all texts with 'S'
    //DWORD dwSize =254;
```

```
    ret = MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile and
g_szDSNName
    if(TRUE == ret)
    {
        memset(&Error, 0, sizeof(CMN_ERROR));
        ret = TXTOpenProject(g_szProjectFile, g_szDSNName, FALSE, &Error);
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TXTOpenProject:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
            ODKTrace(szText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TXTOpenProject"));
            ODKTrace(szText);
            dwItems = 0;
            memset(&Error, 0, sizeof(CMN_ERROR));
            ret = TXTEnumLanguages(g_szProjectFile, &dwItems, MyTXTEnumLanguagesCallback,
&dwUser, &Error);
            if(FALSE == ret)
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TXTOpenProject: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
                ODKTrace(szText);
            }
            dwItems = 0;
            memset(&Error, 0, sizeof(CMN_ERROR));
            ret = TXTEnumInfoText(g_szProjectFile, dwLocale, &dwItems, szFilter,
MyTXTEnumInfotextsCallback, &dwUser, &Error);
            if(FALSE == ret)
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TXTEnumInfoText: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                    Error.dwError1, Error.dwError2, Error.szErrorText);
                ODKTrace(szText);
            }
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TXTEnumInfoText:
NItems=%d ; LANG=%x ; Filter=%s "),
            dwItems, dwLocale, szFilter);
            ODKTrace(szText);
            memset(&Error, 0, sizeof(CMN_ERROR));
            ret = TXTCloseProject(g_szProjectFile, &Error);
            if(FALSE == ret)
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TXTCloseProject: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                    Error.dwError1, Error.dwError2, Error.szErrorText);
                ODKTrace(szText);
```

```
                }
            }
        }
    }
//{{ODK_EXAMPLE}(END)}
```

参见

TXTCloseProject (页 2375)

TXTOpenProject (页 2378)

TXTEnumInfoText (页 2383)

TXT_ENUM_INFOTEXTS_PROC (页 2385)

TXT_ENUM_LANGUAGES_PROC (页 2393)

TXTEnumLanguages (页 2391)

3.7 报表系统函数

3.7.1 基本知识

3.7.1.1 函数概览

概述

RPJAttach (页 2422)	建立连接
RPJCallJobMethod (页 2448)	调用打印作业方法
RPJCreateJob (页 2434)	创建打印作业
RPJCreatePropertyHandle (页 2437)	创建打印作业属性的句柄
RPJDeleteJob (页 2435)	删除打印作业
RPJDeletePropertyHandle (页 2439)	删除打印作业属性的句柄
RPJDetach (页 2423)	断开连接
RPJGetJobMethodAt (页 2450)	确定打印作业方法的名称
RPJGetJobNameAt (页 2440)	确定打印作业的名称

RPJGetJobPropertyAt (页 2452)	确定打印作业属性的名称
RPJGetJobProps (页 2454)	确定打印作业属性
RPJGetNumJobMethods (页 2451)	确定打印作业方法的数目
RPJGetNumJobProperties (页 2456)	确定打印作业属性的数目
RPJGetNumJobs (页 2442)	确定打印作业的数目
RPJGetNumProjectProperties (页 2425)	确定项目中属性的数目
RPJGetProjectPropertyAt (页 2426)	确定项目中使用索引的属性名称和类型
RPJGetProjectProperty (页 2428)	确定项目中属性的值
RPJGetProperty (页 2457)	确定打印作业属性
RPJJobLock (页 2443)	锁定打印作业，避免其他访问
RPJJobUnlockAll (页 2447)	删除所有打印作业锁定
RPJJobUnlock (页 2445)	删除打印作业锁定
RPJMemFree (页 2424)	释放内存
RPJProjectLock (页 2430)	锁定对项目中打印作业列表的访问
RPJProjectUnlockAll (页 2433)	删除项目中所有打印作业列表锁定
RPJProjectUnlock (页 2431)	删除项目中打印作业列表锁定
RPJPropertyClear (页 2460)	删除打印作业属性的句柄
RPJSetProperty (页 2461)	指定打印作业属性

3.7.1.2 常数

布局属性 (CS)

根

- ObjectName
- Context

几何形状

- Left
- Top
- Width
- Height
- RoundCornerWidth

- RoundCornerHeight
- StartAngle
- EndAngle
- Radius
- RadiusWidth
- RadiusHeight
- Columns
- Index
- PointCount
- ActualPointLeft
- ActualPointTop
- PaperSize
- Orientation
- DynMarginLeft
- DynMarginRight
- DynMarginTop
- DynMarginBottom
- PrintMarginLeft
- PrintMarginRight
- PrintMarginTop
- PrintMarginBottom

样式

- BorderStyle
- BorderWidth
- FillStyle

颜色

- BorderBackColor
- BorderColor
- BackColor

- FillColor
- ForeColor

字符

- FontName
- FontSize
- FontBold
- FontItalic
- FontUnderline
- Text
- AlignmentLeft
- AlignmentTop
- WordWrap

其它

- FirstPage
- LastPage
- MetaFileName
- LayoutFileName
- Format
- List
- Tag
- DataType
- OutFormat
- Calculation
- PageBreak

打印作业的属性

AbsoluteSelectionFrom	
AbsoluteSelectionTo	

CycleSpan	
DestinationFile	
EnableCycle	
EnableStart	
EndPage	该信息使您能够在特定页面处结束报告的打印输出。
JobName	打印作业的名称在项目中必须是唯一的，并符合 Windows 惯例。
LayoutName	可以使用布局名称为打印作业分配布局。
PrinterName 1	可在此指定打印输出最初要发送至的打印机。
PrinterName 2	如果第一台打印机不可用，在此指定要使用的打印机。
PrinterName 3	如果前两台打印机不可用，在此指定要使用的打印机。
RelativeSelectionCount	
RelativeSelectionRange	
StartPage	该信息使您能够在特定页面处开始报告的打印输出。
StartTime	
UseRelative	
UseOutputFile	
UseOutputPrinter	

3.7.1.3 错误消息

概述

CMN_ERROR 错误结构中的 API 函数可返回以下错误消息：

CS 的错误消息

ERR_NOERROR	0	无错误
ERR_ILLEGALPROJECT	1	项目名称/项目路径无效
ERR_NOMEMORY	2	内存错误

ERR_UNKNOWNERROR	3	未知错误
ERR_THREADNOTINITIALIZED	4	使用的线程未初始化。
ERR_OLEEXCEPTION	5	MFC 和 OLE 关联的错误
ERR_NOOLESERVERAVAIL	6	无可用 OLE-Server

RT 的错误消息

ERR_NOERROR	0	无错误
ERR_ILLEGALPROJECT	1	项目名称/项目路径无效
ERR_NOMEMORY	2	内存错误
ERR_UNKNOWNERROR	3	未知错误
ERR_NULLHANDLE	4	无法创建句柄。
ERR_ILLEGALPOINTER	5	打印机不正确或无效
ERR_ILLEGALJOBINDEX	6	打印机作业索引不正确
ERR_UNKNOWNPROPERTY	7	打印机作业属性未知
ERR_UNKNOWNMETHOD	8	打印机作业方法未知

3.7.1.4 对象属性列表

所有对象的属性

说明

还可以通过上下文相关的帮助获取属性名称以及基本描述。 要对其进行访问，请在报表编辑器中相关对象的属性对话框内右键单击。

属性名称	数据类型
“Width”	VT_I4
“Height”	VT_I4
“ObjectName”	VT_BSTR
“PageBreak”	VT_I4
“BorderStyle”	VT_I4

“BorderWidth”	VT_I4
“BorderBackColor”	VT_I4
“BorderColor”	VT_I4
“Left”	VT_I4
“Top”	VT_I4

所有“二维对象”的属性：

“FillStyle”	VT_I4
“BackColor”	VT_I4
“FillColor”	VT_I4

“动态对象”的属性：

“DataLink”	VT_BSTR

以下属性是对象特定的属性！

所有“系统”对象的属性：

属性名称	数据类型
“FontName”	VT_BSTR
“FontSize”	VT_I4
“FontBold”	VT_I4
“FontItalic”	VT_I4
“FontUnderline”	VT_I4
“Format”	VT_BSTR
“AlignmentLeft”	VT_I4
“AlignmentTop”	VT_I4
“WordWrap”	VT_I4

“动态表”的属性：

“FontName”	VT_BSTR
“FontSize”	VT_I4
“FontBold”	VT_I4
“FontItalic”	VT_I4
“FontUnderline”	VT_I4
“Orientation”	VT_I4
“Columns”	VT_BSTR
“ForeColor”	VT_I4
“List”	VT_I4

“动态文本”的属性：

“FontName”	VT_BSTR
“FontSize”	VT_I4
“FontBold”	VT_I4
“FontItalic”	VT_I4
“FontUnderline”	VT_I4
“Orientation”	VT_I4
“AlignmentLeft”	VT_I4
“AlignmentTop”	VT_I4
“WordWrap”	VT_I4
“ForeColor”	VT_I4

“动态画面”的属性：

“DynHeight”	VT_I4

“圆弧”的属性：

“Radius”	VT_I4
“StartAngle”	VT_I4
“EndAngle”	VT_I4

“椭圆弧”的属性：

“StartAngle”	VT_I4
“EndAngle”	VT_I4
“RadiusWidth”	VT_I4
“RadiusHeight”	VT_I4

“圆”的属性：

“Radius”	VT_I4

“椭圆”的属性：

“RadiusWidth”	VT_I4
“RadiusHeight”	VT_I4

“多边形”的属性：

“Polyline”	VT_BSTR
“PointCount”	VT_I4
“Index”	VT_I4
“ActualPointLeft”	VT_I4
“ActualPointTop”	VT_I4

“扇形”的属性：

“Radius”	VT_I4
“StartAngle”	VT_I4
“EndAngle”	VT_I4

“椭圆扇形”的属性：

“StartAngle”	VT_I4
“EndAngle”	VT_I4
“RadiusWidth”	VT_I4
“RadiusHeight”	VT_I4

“折线”的属性：

“Polyline”	VT_BSTR
“PointCount”	VT_I4
“Index”	VT_I4
“ActualPointLeft”	VT_I4
“ActualPointTop”	VT_I4

“圆角矩形”的属性：

“RoundCornerWidth”	VT_I4
“RoundCornerHeight”	VT_I4

“文本对象”的属性：

“FontName”	VT_BSTR
“FontSize”	VT_I4
“FontBold”	VT_I4
“FontItalic”	VT_I4

“FontUnderline”	VT_I4
“Orientation”	VT_I4
“Text”	VT_BSTR
“AlignmentLeft”	VT_I4
“AlignmentTop”	VT_I4
“WordWrap”	VT_I4
“ForeColor”	VT_I4

“已插入布局”的属性：

“LayoutFileName”	VT_BSTR

“布局”的属性：

“PrintMarginLeft”	VT_I4
“PrintMarginRight”	VT_I4
“PrintMarginTop”	VT_I4
“PrintMarginBottom”	VT_I4
“PaperSize”	VT_BSTR
“FirstPage”	VT_I4
“LastPage”	VT_I4
“Orientation”	VT_I4
“Printer1”	VT_BSTR
“Printer2”	VT_BSTR
“Printer3”	VT_BSTR
“BackColor”	VT_I4
“BorderColor”	VT_I4
“FillColor”	VT_I4
“DynMarginLeft”	VT_I4
“DynMarginRight”	VT_I4
“DynMarginTop”	VT_I4

“DynMarginBottom”	VT_I4
“Context”	VT_BSTR

“图元文件”的属性：

“MetaFileName”	VT_BSTR

“变量”对象的属性：

“Tag”	VT_BSTR
“FontSize”	VT_I4
“FontName”	VT_BSTR
“FontBold”	VT_I4
“FontItalic”	VT_I4
“FontUnderline”	VT_I4
“AlignmentLeft”	VT_I4
“AlignmentTop”	VT_I4
“WordWrap”	VT_I4
“ForeColor”	VT_I4
“OutFormat”	VT_BSTR
“DataType”	VT_I4
“Calculation”	VT_VARIANT

3.7.1.5 处理打印作业属性的一般步骤（报表编辑器）

建立并初始化与 RPJAPI.DLL 的连接

RPJAttach.

创建打印作业属性的句柄

要处理打印作业的属性，必须创建打印作业句柄。 该句柄用于标识存储区，借此可以对打印作业属性进行寻址。

函数： RPJCreatePropertyHandle.

读取打印作业属性

打印作业的属性存储在项目数据库中。 要编辑这些属性，必须将其从数据库加载到句柄指定的区域。

函数： RPJGetJobProps.

确定打印作业属性

某打印作业属性通过其名称进行引用。 要确定属性值，必须读取该属性。

函数： RPJGetProperty.

指定打印作业属性

某打印作业属性通过其名称进行引用。 要设置属性，将调用 RPJSetProperty 函数。

存储打印作业属性

经过修改的打印作业属性存储在项目数据库中。 必须指定要存储的打印作业的名称。

函数： RPJSetJobProps.

重新初始化打印作业属性

重新初始化句柄引用的存储区，而不释放句柄。

函数： RPJPropertyClear.

指定打印作业属性

某打印作业属性通过其名称进行引用。 通过调用函数来设置属性。

函数： RPJSetProperty.

存储打印作业属性

经过修改的打印作业属性存储在项目数据库中。 必须指定要存储的打印作业的名称。

函数： RPJSetJobProps.

删除句柄

将删除打印作业属性句柄。 对于每个通过 RPJCreatePropertyHandle 创建的句柄，如果不再需要，应将其删除。

函数： RPJDeletePropertyHandle.

终止到 RPJAPI.DLL 的连接

RPJDetach.

示例： 启动打印预览中的打印作业

概述

要处理打印作业的属性，必须提供打印作业句柄。 该句柄用于标识存储区，借此可以对打印作业属性进行寻址。

建立并初始化与 RPJAPI.DLL 的连接

RPJAttach.

创建句柄：函数

RPJCreatePropertyHandle.

读取打印作业的属性

打印作业的属性： “Control Center Redocumentation”会从数据库加载到 hProp 指定的区域。

函数： RPJGetJobProps.

初始化打印作业方法

通过指定 PREVIEW 方法初始化打印作业。 打印作业本身是由 hProp 指定的。

函数： RPJCallJobMethod.

删除句柄

将删除打印作业属性句柄。 对于每个通过 RPJCreatePropertyHandle 创建的句柄，如果不再需要，应将其删除。

函数： RPJDeletePropertyHandle.

终止到 RPJAPI.DLL 的连接

RPJDetach.

示例

```
void PrintJob(void)
{
    CMN_ERROR err; //error structure
    BOOL ret;
    HPROPERTIES hProp;
    char jobname[200];
    char method[200];
    LPCSTR szProjectName = "c:\\rest\\test.mcp"; //name of project
    strcpy(method, "PREVIEW");
    strcpy(jobname, "Backdokumentation Control Center");
    ret = RPJAttach(&err);
    if (TRUE == ret)
    {
        hProp = RPJCreatePropertyHandle(szProjectName. &err);
        if (NULL == hProp)
        {
            ErrMsg("Error RPJCreatePropertyHandle", &err);
        }
        else
        {
            ret = RPJGetJobProps(hProp, jobname, &err);
            if (FALSE == ret)
            {
                ErrMsg("Error RPJGetJobProbs", &err);
            }
            else
            {
                ret = RPJCallJobMethod(hProp, method, &err);
                if (FALSE == ret)
                {
                   ErrMsg("Error executing RPJCallJobMethod", &err);
                }
                else
                {
                   Msg("Print job started.");
                }
            }
            RPJDeletePropertyHandle(hProp, &err);
        }
        RPJDetach(&err);
    }
    else
    {
        Msg("No connection to report designer DLL");
    }
}
```

3.7.2 用于建立连接的函数

3.7.2.1 RPJAttach

说明

该函数用于建立并初始化到 RPJAPI.DLL 的连接。

声明

```
BOOL RPJAttach (
    CMN_ERROR*    pcmnerror );
```

参数

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

连接已建立。

FALSE

错误

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

相关函数

RPJDetach (页 2423)	终止连接

参见

RPJDetach (页 2423)

函数概览 (页 2407)

3.7.2.2 RPJDetach

说明

该函数用于关闭已建立的到 RPJAPI.DLL 的连接。

声明

```
BOOL RPJDetach (
    CMN_ERROR*    pcmnerror );
```

参数

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

连接再次关闭。

FALSE

错误

注释

说明

调用不可用于应用程序的析构函数（EXE、DLL、OCX 等）中。由于 Microsoft 特定的机制，这可能导致调用冻结，从而导致程序崩溃。

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

相关函数

RPJAttach (页 2422)	建立连接

参见

RPJAttach (页 2422)

函数概览 (页 2407)

3.7.2.3 RPJMemFree

说明

该函数用于关闭已建立的到 RPJAPI.DLL 的连接并释放已分配的内存。

声明

```
BOOL RPJMemFree (
    const PVOID    pvMemBlock,
    CMN_ERROR*     pcmnerror );
```

参数

pvMemBlock

指向之前由 RPJ 函数分配的已释放存储区。

如果 pvMemBlock = NULL，则不会出现任何情况。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

内存已释放。

FALSE

错误

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

参见

函数概览 (页 2407)

3.7.3 用于处理项目属性的函数

3.7.3.1 RPJGetNumProjectProperties

说明

该函数可确定项目的属性数。

声明

```
BOOL RPJGetNumProjectProperties (
    LPCSTR        pszReserved,
    DWORD*        pdwNumProperties
    CMN_ERROR*    pcmnerror );
```

参数

pszReserved

该参数预留给将来开发使用。

pdwNumProperties

指向返回属性数的 DWORD 变量的指针。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

属性数已确定。

FALSE

错误

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

参见

函数概览 (页 2407)

3.7.3.2 RPJGetProjectPropertyAt

说明

该函数可根据项目中的索引确定属性的名称和类型。

声明

```
BOOL RPJGetProjectPropertyAt (
    LPCSTR        pszReserved,
    DWORD         dwPropIndex,
    LPSTR         pszBuffer,
    DWORD         dwCharMax,
    DWORD*        pdwPrpType,
    CMN_ERROR*    pcmnerror );
```

参数

pszReserved

该参数预留以供将来扩展。

dwPropIndex

项目中属性的索引，将从中返回相关信息。

pszBuffer

指向返回属性名称的缓冲区的指针。缓冲区的大小由 dwCharMax 指定。

dwCharMax

缓冲区的最大大小由 pszBuffer 指定。

pdwPropType

指向返回项目中属性类型的 DWORD 变量的指针。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

项目中属性的名称和类型已确定。

FALSE

错误

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

参见

函数概览 (页 2407)

3.7.3.3 RPJGetProjectProperty

说明

该函数可确定属性值。必须传递项目名称和属性名称。

声明

```
BOOL RPJGetProjectProperty (
    LPCSTR        pszProjectName,
    LPCSTR        pszPropName,
    PVOID         pvPropValue,
    VARTYPE       vtPropType,
    DWORD         dwBufferSize,
    CMN_ERROR*    pcmnerror );
```

参数

pszProjectName

指向项目名称（包括路径、项目名称和扩展名）的指针。

pszPropName

项目中属性的名称。

pvPropValue

指向返回属性值的缓冲区的指针。pvPropValue 缓冲区大小由 dwBufferSize 指定。

vtPropType

pvPropValue 缓冲区中属性值的期望类型。

VT_I4	整型或布尔型
VT_LPSTR	ANSI 格式的文本
VT_LPWSTR	UNICODE 格式的文本
VT_DATE	日期/时间

dwBufferSize

属性值应返回至的 pvPropValue 中缓冲区的大小。只能以 BYTE 指定大小，而不能以 CHAR 指定大小。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

属性值已确定。

FALSE

错误

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

参见

函数概览 (页 2407)

3.7.3.4 RPJProjectLock

说明

针对其他方锁定对项目中打印作业列表的访问。必须在 pszLockerNameNew 中指定锁定方的名称。如果出现访问冲突，则会通过 ppszLockerNameCur 返回锁定方的名称。

声明

```
BOOL RPJProjectLock (
    LPCSTR        pszProjectName,
    BOOL          fWriteLock,
    BOOL          fDoNotWait,
    LPCSTR        pszLockerNameNew,
    LPSTR*        ppszLockerNameCur,
    CMN_ERROR*    pcmnerror );
```

参数

pszProjectName

指向项目名称（包括路径、项目名称和扩展名）的指针。

fWriteLock

指定锁定的类型。

TRUE	锁定写操作和读操作。
FALSE	只锁定读操作。

fDoNotWait

TRUE	如果存在锁定，则会立即报告错误。
FALSE	如果存在锁定，首先会进行周期性检查。几秒钟之后，出现超时错误，检查停止。

pszLockerNameNew

包含锁定方的名称。如果出现访问冲突，则会通过 ppszLockerNameCur 返回名称。

ppszLockerNameCur

如果出现访问冲突，则会在 ppszLockerNameCur 中返回当前锁定方的名称。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

打印作业列表已锁定。

FALSE

访问冲突已出现。项目的打印作业列表已锁定。将通过 ppszLockerNameCur 返回锁定方的名称。

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

相关函数

RPJProjectUnlock (页 2431)	删除锁定
RPJProjectUnlockAll (页 2433)	删除所有锁定

参见

RPJProjectUnlock (页 2431)

RPJProjectUnlockAll (页 2433)

函数概览 (页 2407)

3.7.3.5 RPJProjectUnlock

说明

将删除打印作业列表的锁定。

声明

```
BOOL RPJProjectUnlock (
    LPCSTR        pszProjectName,
    CMN_ERROR*    pcmnerror );
```

参数

pszProjectName

指向项目文件名称（包括路径、项目名称和扩展名）的指针。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

打印作业列表的锁定已删除。

FALSE

访问冲突已出现。

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

相关函数

RPJProjectLock (页 2430)	创建锁定
RPJProjectUnlockAll (页 2433)	删除所有锁定

参见

RPJProjectLock (页 2430)

RPJProjectUnlockAll (页 2433)

函数概览 (页 2407)

3.7.3.6 RPJProjectUnlockAll

说明

将删除打印作业列表的所有锁定。

声明

```
BOOL RPJProjectUnlockAll (
    LPCSTR        pszProjectName,
    CMN_ERROR*    pcmnerror );
```

参数

pszProjectName

指向项目文件名称（包括路径、项目名称和扩展名）的指针。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

打印作业列表的所有锁定已删除。

FALSE

访问冲突已出现。

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

相关函数

RPJProjectLock (页 2430)	创建锁定
RPJProjectUnlock (页 2431)	删除锁定

参见

RPJProjectLock (页 2430)

RPJProjectUnlock (页 2431)

函数概览 (页 2407)

3.7.4 用于处理打印作业的函数

3.7.4.1 RPJCreateJob

说明

创建新的打印作业。

声明

```
BOOL RPJCreateJob (
    LPCSTR        pszProjectName,
    LPCSTR        pszJobName,
    CMN_ERROR*    pcmnerror );
```

参数

pszProjectName

指向项目文件名称（包括路径、项目名称和扩展名）的指针。

pszJobName

指向要创建打印作业的名称的指针。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

打印作业已创建。

FALSE

错误

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

相关函数

RPJDeleteJob (页 2435)	删除打印作业

参见

RPJDeleteJob (页 2435)

函数概览 (页 2407)

3.7.4.2 RPJDeleteJob

说明

将删除 pszJobName 指定的打印作业。

声明

```
BOOL RPJDeleteJob (
    LPCSTR        pszProjectName,
    LPCSTR        pszJobName,
    CMN_ERROR*    pcmnerror );
```

参数

pszProjectName

指向项目文件名称（包括路径、项目名称和扩展名）的指针。

pszJobName

指向要删除打印作业的名称的指针。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

打印作业已删除。

FALSE

错误

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

相关函数

RPJCreateJob (页 2434)	创建打印作业

参见

RPJCreateJob (页 2434)

函数概览 (页 2407)

3.7.4.3 RPJCreatePropertyHandle

说明

要编辑打印作业的属性，首先应通过该函数组态该作业的句柄。将在返回值中获取句柄。

声明

```
HPROPERTIES RPJCreatePropertyHandle (
    LPCSTR        pszProjectName,
    CMN_ERROR*    pcmnerror );
```

参数

pszProjectName

指向项目文件名称（包括路径、项目名称和扩展名）的指针。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

打印作业属性的句柄。

如果出现错误，则返回值为 ZERO。

注释

此处提供的句柄是 RPJGetJobProps、RPJGetProperty 和 RPJSetProperty 之类的函数需要的。

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

相关函数

RPJDeletePropertyHandle (页 2439)	删除句柄
RPJPropertyClear (页 2460)	初始化存储区
RPJGetJobProps (页 2454)	确定打印作业属性
RPJGetProperty (页 2457)	确定打印作业属性
RPJSetProperty (页 2461)	指定打印作业属性

示例

显示打印作业预览 (页 2480) "RD02.cpp"

修改打印作业属性 (页 2475) "RD02.cpp"

获取打印作业名称 (页 2469) "RD02.cpp"

获取打印作业方法名称 (页 2466) "RD02.cpp"

获取打印作业属性 (页 2472) "RD02.cpp"

参见

获取打印作业方法名称 (页 2466)

获取打印作业名称 (页 2469)

获取打印作业属性 (页 2472)

修改打印作业属性 (页 2475)

显示打印作业预览 (页 2480)

RPJDeletePropertyHandle (页 2439)

RPJGetProperty (页 2457)

RPJGetJobProps (页 2454)

RPJSetProperty (页 2461)

RPJPropertyClear (页 2460)

函数概览 (页 2407)

3.7.4.4 RPJDeletePropertyHandle

说明

对于每个通过 RPJCreatePropertyHandle 创建的句柄，如果不再需要，应将其删除。

声明

```
BOOL RPJDeletePropertyHandle (
    HPROPERTIES    hproperties,
    CMN_ERROR*     pcmnerror );
```

参数

hproperties

打印作业属性的句柄由 RPJCreatePropertyHandle 函数创建。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

句柄已删除

FALSE

错误

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

相关函数

RPJCreatePropertyHandle (页 2437)	创建句柄

示例

显示打印作业预览 (页 2480) "RD02.cpp"

修改打印作业属性 (页 2475) "RD02.cpp"

获取打印作业名称 (页 2469) "RD02.cpp"

获取打印作业方法名称 (页 2466) "RD02.cpp"

获取打印作业属性 (页 2472) "RD02.cpp"

参见

RPJCreatePropertyHandle (页 2437)

获取打印作业方法名称 (页 2466)

获取打印作业名称 (页 2469)

获取打印作业属性 (页 2472)

修改打印作业属性 (页 2475)

显示打印作业预览 (页 2480)

函数概览 (页 2407)

3.7.4.5 RPJGetJobNameAt

说明

确定 dwJobIndex 指定的打印作业的名称。

声明

```
BOOL RPJGetJobNameAt (
    LPCSTR        pszProjectName,
    DWORD         dwJobIndex,
    LPSTR         pszBuffer,
    DWORD         dwCharMax,
    CMN_ERROR*    pcmnerror );
```

参数

PszProjectName

指向项目文件名称（包括路径、项目名称和扩展名）的指针。

dwJobIndex

应确定其名称的打印作业的索引。

pszBuffer

指向打印作业名称应返回的接收缓冲区的指针。

dwCharMax

接收缓冲区的大小（以字符数表示，包括零终止）。该值必须大于 1。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

打印作业的名称已确定。

FALSE

错误

注释

RPJGetJobNameAt 指示的索引错误可能是由于调用 RPJGetNumJobs 后作业数发生变化引起的。

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

示例

获取打印作业名称 (页 2469) "RD02.cpp"

参见

获取打印作业名称 (页 2469)

函数概览 (页 2407)

3.7.4.6 RPJGetNumJobs

说明

确定当前未决打印作业的数目。调用该函数之后，这一数目可能由于删除或添加打印作业而改变。

声明

```
BOOL RPJGetNumJobs (
    LPCSTR        pszProjectName,
    DWORD*        pdwNumJobs,
    CMN_ERROR*    pcmnerror );
```

参数

pszProjectName

指向项目文件名称（包括路径、项目名称和扩展名）的指针。

pdwNumJobs

指向应存储打印作业数的位置的指针。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

打印作业数已确定。

FALSE

错误

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

示例

获取打印作业名称 (页 2469) "RD02.cpp"

参见

获取打印作业名称 (页 2469)

函数概览 (页 2407)

3.7.4.7 RPJJobLock

说明

针对其他方锁定对指定打印作业的访问。必须在 pszLockerNameNew 中指定锁定方的名称。如果出现访问冲突，则会通过 ppszLockerNameCur 返回锁定方的名称。

声明

```
BOOL RPJJobLock (
    LPCSTR        pszProjectName,
    LPCSTR        pszJobName,
    BOOL          fWriteLock,
    BOOL          fDoNotWait,
    LPCSTR        pszLockerNameNew,
    LPSTR*        ppszLockerNameCur,
    CMN_ERROR*    pcmnerror );
```

参数

pszProjectName

指向项目名称（包括路径、项目名称和扩展名）的指针。

pszJobName

指向要锁定打印作业的名称的指针。

fWriteLock

指定锁定的类型。

TRUE	锁定写操作和读操作。
FALSE	只锁定读操作。

fDoNotWait

TRUE	如果存在锁定，则会立即报告错误。
FALSE	如果存在锁定，首先会进行周期性检查。几秒钟之后，出现超时错误，检查停止。

pszLockerNameNew

包含锁定方的名称。如果出现访问冲突，则会通过 ppszLockerNameCur 返回名称。

ppszLockerNameCur

如果出现访问冲突，则会在 ppszLockerNameCur 中返回当前锁定方的名称。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

打印作业已锁定。

FALSE

访问冲突已出现。项目的打印作业列表已锁定。将通过 ppszLockerNameCur 返回锁定方的名称。

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

相关函数

RPJJobUnlock (页 2445)	删除锁定
RPJJobUnlockAll (页 2447)	删除所有锁定

参见

RPJJobUnlockAll (页 2447)

RPJJobUnlock (页 2445)

函数概览 (页 2407)

3.7.4.8 RPJJobUnlock

说明

将删除打印作业的锁定。

声明

```
BOOL RPJJobUnlock (
    LPCSTR        pszProjectName,
    LPCSTR        pszJobName,
    CMN_ERROR*    pcmnerror );
```

参数

pszProjectName

指向项目文件名称（包括路径、项目名称和扩展名）的指针。

pszJobName

指向打印作业名称的指针。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

打印作业的锁定已删除。

FALSE

访问冲突已出现。

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

相关函数

RPJJobLock (页 2443)	创建锁定
RPJJobUnlockAll (页 2447)	删除所有锁定

参见

RPJJobUnlockAll (页 2447)

RPJJobLock (页 2443)

函数概览 (页 2407)

3.7.4.9 RPJJobUnlockAll

说明

将删除打印作业的所有锁定。

声明

```
BOOL RPJJobUnlockAll (
    LPCSTR        pszProjectName,
    LPCSTR        pszJobName,
    CMN_ERROR*    pcmnerror );
```

参数

pszProjectName

指向项目文件名称（包括路径、项目名称和扩展名）的指针。

pszJobName

指向打印作业名称的指针。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

指定打印作业的所有锁定已删除。

FALSE

访问冲突已出现

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

相关函数

RPJJobLock (页 2443)	创建锁定
RPJJobUnlock (页 2445)	删除锁定

参见

RPJJobUnlock (页 2445)

RPJJobLock (页 2443)

函数概览 (页 2407)

3.7.5 用于处理打印作业方法的函数

3.7.5.1 RPJCallJobMethod

说明

该函数允许指定特定的作业方法。目前有两种方法：PrintJob 和 PreviewJob。

声明

```
RPJCallJobMethod (
    HPROPERTIES    hproperties,
    LPCSTR         pszMethodName
    CMN_ERROR*     pcmnerror );
```

参数

hproperties

打印作业属性的句柄由 RPJCreatePropertyHandle 函数创建。

pszMethodName

指向要使用作业方法的名称的指针。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

打印作业方法已指定

FALSE

错误

注释

为了通过 RPJCallJobMethod 指定打印作业方法，首先需要执行 RPJGetJobProps 函数。

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

示例

显示打印作业预览 (页 2480) "RD02.cpp"

参见

显示打印作业预览 (页 2480)

函数概览 (页 2407)

3.7.5.2 RPJGetJobMethodAt

说明

该函数确定 dwMethodIndex 中指定的打印方法的名称。

声明

```
BOOL RPJGetJobMethodAt (
    DWORD          dwMethodIndex,
    LPSTR          pszBuffer,
    DWORD          dwCharMax,
    CMN_ERROR*     pcmnerror );
```

参数

dwMethodIndex

将确定其名称的打印方法的索引。

pszBuffer

指向返回打印方法名称的缓冲区的指针。

dwCharMax

接收名称的缓冲区大小（以字符数表示，包括零终止）。该值必须大于 1。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

打印作业方法的名称已确定

FALSE

错误

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

示例

获取打印作业方法名称 (页 2466) "RD02.cpp"

参见

获取打印作业方法名称 (页 2466)

函数概览 (页 2407)

3.7.5.3 RPJGetNumJobMethods

说明

该函数确定目前有多少种可用打印作业方法。

声明

```
BOOL RPJGetNumJobMethods (
    DWORD* pdwNumMethods,
    CMN_ERROR* pcmnerror );
```

参数

pdwNumMethods

指向应存储打印作业方法数的位置的指针。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

打印作业方法数已确定。

FALSE

错误

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

示例

获取打印作业方法名称 (页 2466) "RD02.cpp"

参见

获取打印作业方法名称 (页 2466)

函数概览 (页 2407)

3.7.6 用于处理打印作业方法的函数

3.7.6.1 RPJGetJobPropertyAt

说明

确定 dwPropindex 中指定的作业属性的名称和类型。

声明

```
BOOL RPJGetJobPropertyAt (
    DWORD          dwPropIndex,
    LPSTR          pszBuffer,
    DWORD          dwCharMax,
    DWORD*         pdwPropType,
    CMN_ERROR*     pcmnerror );
```

参数

dwPropIndex

应确定其名称的打印作业属性的索引。

pszBuffer

指向接收作业属性名称的缓冲区的指针。

dwCharMax

接收缓冲区的大小（以字符数表示，包括零终止）。该值必须大于 1。

pdwPropType

指向应存储作业属性类型的位置的指针。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

打印作业属性的名称已确定

FALSE

错误

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

示例

获取打印作业属性 (页 2472) "RD02.cpp"

参见

获取打印作业属性 (页 2472)

函数概览 (页 2407)

3.7.6.2 RPJGetJobProps

说明

该函数确定打印作业的属性并将其存储在内部 hProps 缓冲区中。

声明

```
BOOL RPJGetJobProps (
    HPROPERTIES    hproperties,
    LPCSTR         pszJobName,
    CMN_ERROR*     pcmnerror );
```

参数

hproperties

打印作业属性的句柄由 RPJCreatePropertyHandle 函数创建。

pszJobName

指向打印作业名称的指针。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

注释

该函数是 RPJGetProperty 和 RPJSetProperty 等函数的先决条件。

返回值

TRUE

打印作业属性已确定

FALSE

错误

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

相关函数

RPJCreatePropertyHandle (页 2437)	创建句柄
RPJGetProperty (页 2457)	确定打印作业属性
RPJSetProperty (页 2461)	指定打印作业属性

示例

显示打印作业预览 (页 2480) "RD02.cpp"

修改打印作业属性 (页 2475) "RD02.cpp"

参见

RPJGetProperty (页 2457)

RPJSetProperty (页 2461)

修改打印作业属性 (页 2475)

显示打印作业预览 (页 2480)

RPJCreatePropertyHandle (页 2437)

函数概览 (页 2407)

3.7.6.3 RPJGetNumJobProperties

说明

确定可用打印作业属性的数目。

声明

```
BOOL RPJGetNumJobProperties (
    DWORD*        pdwNumProperties,
    CMN_ERROR*    pcmnerror );
```

参数

pdwNumProperties

指向应存储打印作业属性数的位置的指针。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

打印作业属性数已确定。

FALSE

错误

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

示例

获取打印作业属性 (页 2472) "RD02.cpp"

参见

获取打印作业属性 (页 2472)

函数概览 (页 2407)

3.7.6.4 RPJGetProperty

说明

可以使用 RPJGetProperty 来确定 pszPropName 指定的打印作业属性的值之前，首先需要使用 RPJGetJobProps 来存储内部 hProp 缓冲区中的作业属性。

声明

```
BOOL RPJGetProperty (
    HPROPERTIES    hproperties,
    LPCSTR         pszPropName,
    PVOID          pvPropValue,
    VARTYPE        vtPropType,
    DWORD          dwBufferSize,
    CMN_ERROR*     pcmnerror );
```

参数

hproperties

打印作业属性的句柄由 RPJCreatePropertyHandle 函数创建。

pszPropName

指向打印作业属性名称的指针：

AbsoluteSelectionFrom	
AbsoluteSelectionTo	
CycleSpan	
DestinationFile	
EnableCycle	
EnableStart	
EndPage	该信息使您能够在特定页面处停止报告的打印输出。

JobName	打印作业的名称在项目中必须是唯一的，并符合 Windows 惯例。
LayoutName	可以使用布局名称为打印作业分配布局。
PrinterName 1	可在此指定打印输出最初要发送至的打印机。
PrinterName 2	该属性指定在第一台打印机不可用的情况下要使用的打印机。
PrinterName 3	该属性指定在前两台打印机均不可用的情况下要使用的打印机。
RelativeSelectionCount	
RelativeSelectionRange	
StartPage	该信息使您能够在特定页面处启动报表的打印输出。
StartTime	
UseRelative	
UseOutputFile	
UseOutputPrinter	

pvPropValue

指向应接收该名称的缓冲区的指针。

vtPropType

对象属性的类型取决于 pszPropName：

AbsoluteSelectionFrom	VT_DATE
AbsoluteSelectionTo	VT_DATE
CycleSpan	VT_I4
DestinationFile	VT_LPSTR
EnableCycle	VT_I4
EnableStart	VT_I4
EndPage	VT_I4
JobName	VT_LPSTR / VT_LPWSTR
LayoutName	VT_LPSTR / VT_LPWSTR
PrinterName	VT_LPSTR / VT_LPWSTR
PrinterName2	VT_LPSTR / VT_LPWSTR
PrinterName3	VT_LPSTR / VT_LPWSTR
RelativeSelectionCount	VT_I4
RelativeSelectionRange	VT_I4

StartPage	VT_I4
StartTime	VT_DATE
UseRelative	VT_I4
UseOutputFile	VT_I4
UseOutputPrinter	VT_I4

如果想要更改字符串且应用程序以 Unicode 编译，则必须使用 vtProp VT_LPWSTR 作为数据类型。否则，vtProp 具有 VT_LPTSTR 数据类型。

如果 vtProp 的数据类型为 VT_DATE，则 prop 参数必须引用 SYSTEMTIME 结构。

dwBufferSize

prop 引用的缓冲区的大小（单位为字节）。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

打印作业属性已确定

FALSE

错误

注释

如果用于接收作业属性的缓冲区组态得过小，则会填充缓冲区，但函数会返回 ERR_UNKNOWNERROR 错误消息（例如，作业名称和布局名称可能会出现这种情况）。

错误消息

ERR_UNKNOWNERROR	未知错误

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

相关函数

RPJCreatePropertyHandle (页 2437)	创建句柄
RPJGetJobProps (页 2454)	确定打印作业属性
RPJSetProperty (页 2461)	指定打印作业属性

示例

修改打印作业属性 (页 2475) "RD02.cpp"

参见

RPJGetJobProps (页 2454)

RPJSetProperty (页 2461)

RPJCreatePropertyHandle (页 2437)

修改打印作业属性 (页 2475)

函数概览 (页 2407)

3.7.6.5 RPJPropertyClear

说明

如果要初始化句柄引用的存储区但不删除该句柄，可使用该函数。

声明

```
BOOL RPJPropertyClear (
    HPROPERTIES    hproperties,
    CMN_ERROR*     pcmnerror );
```

参数

hproperties

打印作业属性的句柄由 RPJCreatePropertyHandle 函数创建。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

内存已初始化。

FALSE

错误

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

相关函数

RPJCreatePropertyHandle (页 2437)	创建句柄

参见

RPJCreatePropertyHandle (页 2437)

函数概览 (页 2407)

3.7.6.6 RPJSetProperty

说明

可以使用该函数临时更改各个打印作业属性。

可以使用 RPJSetProperty 复位 pszPropName 指定的打印作业属性的值之前，首先需要使用 RPJGetJobProps 来存储内部 hProp 缓冲区中的作业属性。

声明

```
BOOL RPJSetProperty (
    HPROPERTIES        hproperties,
    LPCSTR             pszPropName,
    PVOID             pvPropValue,
    VARTYPE             vtPropType,
    DWORD             dwReserved,
    CMN_ERROR*         pcmnerror );
```

参数

hproperties

打印作业属性的句柄。该句柄由 RPJCreatePropertyHandle 函数创建。

pszPropName

指向打印作业属性名称的指针：

AbsoluteSelectionFrom	
AbsoluteSelectionTo	
CycleSpan	
DestinationFile	
EnableCycle	
EnableStart	
EndPage	该信息使您能够在特定页面处停止报告的打印输出。
JobName	打印作业的名称在项目中必须是唯一的，并符合 Windows 惯例。
LayoutName	可以使用布局名称为打印作业分配布局。
PrinterName 1	可在此指定打印输出最初要发送至的打印机。
PrinterName 2	该属性指定在第一台打印机不可用的情况下要使用的打印机。
PrinterName 3	该属性指定在前两台打印机均不可用的情况下要使用的打印机。
RelativeSelectionCount	
RelativeSelectionRange	

StartPage	该信息使您能够在特定页面处启动报表的打印输出。
StartTime	
UseRelative	
UseOutputFile	
UseOutputPrinter	

pvPropValue

指向对象属性值的指针。

vtPropType

对象属性的类型取决于 pszPropName：

AbsoluteSelectionFrom	VT_DATE
AbsoluteSelectionTo	VT_DATE
CycleSpan	VT_I4
DestinationFile	VT_LPSTR
EnableCycle	VT_I4
EnableStart	VT_I4
EndPage	VT_I4
JobName	VT_LPSTR / VT_LPWSTR
LayoutName	VT_LPSTR / VT_LPWSTR
PrinterName	VT_LPSTR / VT_LPWSTR
PrinterName2	VT_LPSTR / VT_LPWSTR
PrinterName3	VT_LPSTR / VT_LPWSTR
RelativeSelectionCount	VT_I4
RelativeSelectionRange	VT_I4
StartPage	VT_I4
StartTime	VT_DATE
UseRelative	VT_I4
UseOutputFile	VT_I4
UseOutputPrinter	VT_I4

如果想要更改字符串且应用程序以 Unicode 编译，则必须使用 vtProp VT_LPWSTR 作为数据类型。否则，vtProp 具有 VT_LPTSTR 数据类型。

如果 vtProp 的数据类型为 VT_DATE，则 prop 参数必须引用 SYSTEMTIME 结构。

dwReserved

该参数预留以供将来扩展。

pcmnerror

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

打印作业属性暂时已更改

FALSE

错误

注释

可以通过 RPJGetNumJobProperties 和 RPJGetJobPropertyAt 函数确定所有可用打印作业的属性。

所需文件

rpjapi.h

rpjapi.lib

rpjapi.dll

相关函数

RPJCreatePropertyHandle (页 2437)	创建句柄
RPJGetJobProps (页 2454)	确定打印作业属性
RPJGetProperty (页 2457)	确定打印作业属性

示例

修改打印作业属性 (页 2475) "RD02.cpp"

参见

RPJGetJobProps (页 2454)

RPJGetProperty (页 2457)

RPJCreatePropertyHandle (页 2437)

修改打印作业属性 (页 2475)

函数概览 (页 2407)

3.7.7 示例

3.7.7.1 获取打印作业方法名称

示例

```
//{{ODK_EXAMPLE}Get print job methodname (RPT)}
//{{FUNCTION}RPJCreatePropertyHandle (RPT)}
//{{FUNCTION}RPJDeletePropertyHandle (RPT)}
//{{FUNCTION}RPJGetNumJobMethods (RPT)}
//{{FUNCTION}RPJGetJobMethodAt (RPT)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: MyPrintJobPropertyInquire( void ) ODK RD RT
// ========================================================================
// : Get print job methodname (Enum) !
// ========================================================================
void MyRPJGetJobMethodAt(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    CMN_ERROR Error;
    HPROPERTIES hProp = NULL;
    TCHAR jobname[255];
    TCHAR szText[255];
    DWORD d = 0L;
    DWORD i = 0L;
    char buf[500+1];
    DWORD dwBufsize = 500L;
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("--------------------"));
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("MyRPJGetJobMethodAt:"));
    ODKTrace(szText);
    MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJAttach(&Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJAttach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        return;
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJAttach"));
        ODKTrace(szText);
```

```
    }
    _tcsncpy_s(jobname, _countof(jobname), _T("ODK_PRINTJOB"), _TRUNCATE); // print job
    memset(&Error, 0, sizeof(CMN_ERROR));
    hProp = RPJCreatePropertyHandle (/*PROJ_PATH*/g_szProjectFile, &Error);
    if(NULL == hProp)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJCreatePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        ret = RPJDetach(&Error);
        if (FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
        }
        ODKTrace(szText);
        return;
    }
    else
    {
         _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJCreatePropertyHandle =
0x%08lx"), hProp);
        ODKTrace(szText);
    }
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = RPJGetNumJobMethods(&d, &Error);
    if(FALSE == ret)
    {
       _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJGetNumJobMethods:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
    else
    {
        for (i = 0; i < d; i++)
        {
            memset(&Error,0,sizeof(CMN_ERROR));
            ret = RPJGetJobMethodAt(i, buf, dwBufsize, &Error);
            if (ret == FALSE)
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJDeletePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
            }
```

```
            else
            {
                 _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   %d
PrintJobMethod=%s"),i+1,buf);
            }
            ODKTrace(szText);
        }
    }
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = RPJDeletePropertyHandle(hProp, &Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJDeletePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDeletePropertyHandle"));
    }
    ODKTrace(szText);
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = RPJDetach(&Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
    }
    ODKTrace(szText);
}
//{{ODK_EXAMPLE}(END)}
```

参见

RPJCreatePropertyHandle (页 2437)

RPJDeletePropertyHandle (页 2439)

RPJGetJobMethodAt (页 2450)

RPJGetNumJobMethods (页 2451)

3.7.7.2 获取打印作业名称

示例

```
//{{ODK_EXAMPLE}Get print job names (RPT)}
//{{FUNCTION}RPJCreatePropertyHandle (RPT)}
//{{FUNCTION}RPJDeletePropertyHandle (RPT)}
//{{FUNCTION}RPJGetNumJobs (RPT)}
//{{FUNCTION}RPJGetJobNameAt (RPT)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: MyRPJGetJobNameAt( void ) ODK RD RT
// ========================================================================
// : Get all print job names (Enum) !
// ========================================================================
void MyRPJGetJobNameAt(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    CMN_ERROR Error;
    HPROPERTIES hProp = NULL;
    TCHAR jobname[255];
    TCHAR szText[255];
    DWORD d = 0L;
    DWORD i = 0L;
    char buf[500+1];
    DWORD dwBufsize = 500;
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("--------------------"));
    ODKTrace(szText);
     _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("MyRPJGetJobNameAt:"));
    ODKTrace(szText);
    MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJAttach(&Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJAttach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        return;
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJAttach"));
        ODKTrace(szText);
    }
    _tcsncpy_s(jobname, _countof(jobname), _T("ODK_PRINTJOB"), _TRUNCATE); // print job
    memset(&Error, 0, sizeof(CMN_ERROR));
```

```
    hProp = RPJCreatePropertyHandle (/*PROJ_PATH*/g_szProjectFile, &Error);
    if(NULL == hProp)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJCreatePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        ret = RPJDetach(&Error);
        if (FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
        }
        ODKTrace(szText);
        return;
    }
    else
    {
         _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJCreatePropertyHandle =
0x%08lx"), hProp);
        ODKTrace(szText);
    }
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = RPJGetNumJobs(/*PROJ_PATH*/g_szProjectFile, &d, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJGetNumJobs: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
    else
    {
        for (i = 0; i < d; i++)
        {
            memset(&Error,0,sizeof(CMN_ERROR));
            ret = FALSE;
            ret = RPJGetJobNameAt(/*PROJ_PATH*/g_szProjectFile, i, buf, dwBufsize, &Error);
            if (ret == FALSE)
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJDeletePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
            }
            else
            {
```

```
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   %d
PrintJobName=%s"),i+1, buf);
            }
            ODKTrace(szText);
        }
    }
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = FALSE;
    ret = RPJDeletePropertyHandle(hProp, &Error);
    if (ret == FALSE)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJDeletePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDeletePropertyHandle"));
    }
    ODKTrace(szText);
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = RPJDetach(&Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
    }
    ODKTrace(szText);
}
//{{ODK_EXAMPLE}(END)}
```

参见

RPJCreatePropertyHandle (页 2437)

RPJDeletePropertyHandle (页 2439)

RPJGetJobNameAt (页 2440)

RPJGetNumJobs (页 2442)

3.7.7.3 获取打印作业属性

示例

```
//{{ODK_EXAMPLE}Get print job properties (RPT)}
//{{FUNCTION}RPJCreatePropertyHandle (RPT)}
//{{FUNCTION}RPJDeletePropertyHandle (RPT)}
//{{FUNCTION}RPJGetNumJobProperties (RPT)}
//{{FUNCTION}RPJGetJobPropertyAt (RPT)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: MyRPJGetJobPropertyAt( void ) ODK RD RT
// ========================================================================
// : Inquire print job properties (Enum) !
// ========================================================================
void MyRPJGetJobPropertyAt(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    CMN_ERROR Error;
    HPROPERTIES hProp = NULL;
    TCHAR jobname[255];
    TCHAR szText[255];
    DWORD d = 0L;
    DWORD f = 0L;
    DWORD i = 0L;
    char buf[500+1];
    DWORD dwBufsize = 500L;
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("--------------------"));
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("MyRPJGetJobPropertyAt:"));
    ODKTrace(szText);
    MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJAttach(&Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJAttach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        return;
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJAttach"));
        ODKTrace(szText);
    }
    _tcsncpy_s(jobname, _countof(jobname), _T("ODK_PRINTJOB"), _TRUNCATE); // print job
```

```
    memset(&Error, 0, sizeof(CMN_ERROR));
    hProp = RPJCreatePropertyHandle (/*PROJ_PATH*/g_szProjectFile, &Error);
    if(NULL == hProp)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJCreatePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        ret = RPJDetach(&Error);
        if (FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
        }
        ODKTrace(szText);
        return;
    }
    else
    {
         _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJCreatePropertyHandle =
0x%08lx"), hProp);
        ODKTrace(szText);
    }
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = RPJGetNumJobProperties(&d, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJGetNumJobProperties: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
    else
    {
        for (i = 0; i < d; i++)
        {
            memset(&Error,0,sizeof(CMN_ERROR));
            ret = RPJGetJobPropertyAt(i, buf, dwBufsize, &f, &Error);
            if (FALSE == ret)
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJGetJobPropertyAt: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
            }
            else
            {
```

```
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   %d
PrintJobProperty=%s"),i+1, buf);
            }
            ODKTrace(szText);
        }
    }
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = RPJDeletePropertyHandle(hProp, &Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJDeletePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDeletePropertyHandle"));
    }
    ODKTrace(szText);
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = RPJDetach(&Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
    }
    ODKTrace(szText);
}
//{{ODK_EXAMPLE}(END)}
```

参见

RPJCreatePropertyHandle (页 2437)

RPJDeletePropertyHandle (页 2439)

RPJGetJobPropertyAt (页 2452)

RPJGetNumJobProperties (页 2456)

3.7.7.4 修改打印作业属性

示例

```
//{{ODK_EXAMPLE}Modify print job properties (RPT)}
//{{FUNCTION}RPJCreatePropertyHandle (RPT)}
//{{FUNCTION}RPJDeletePropertyHandle (RPT)}
//{{FUNCTION}RPJGetJobProps (RPT)}
//{{FUNCTION}RPJGetProperty (RPT)}
//{{FUNCTION}RPJSetJobProps (RPT)}
//{{FUNCTION}RPJSetProperty (RPT)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: MyModifyPrintJob( void ) ODK RD RT
// ========================================================================
// : Modify print job properties (change starttime) !
// ========================================================================
void MyModifyPrintJob(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    CMN_ERROR Error;
    TCHAR szText[255];
    HPROPERTIES hProp = NULL;
    SYSTEMTIME st;
    LPVOID ptr;
    DWORD typ;
    TCHAR jobname[200];
    TCHAR propname[200];
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("--------------------"));
    ODKTrace(szText);
     _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("MyModifyPrintJob:"));
    ODKTrace(szText);
    MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJAttach(&Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJAttach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        return;
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJAttach"));
        ODKTrace(szText);
    }
```

```
    // read properties
    memset(&Error, 0, sizeof(CMN_ERROR));
    hProp = RPJCreatePropertyHandle (/*PROJ_PATH*/g_szProjectFile, &Error);
    if(NULL == hProp)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJCreatePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        ret = RPJDetach(&Error);
        if (FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
        }
        ODKTrace(szText);
        return;
    }
    else
    {
         _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJCreatePropertyHandle =
0x%08lx"), hProp);
        ODKTrace(szText);
    }
    _tcsncpy_s(jobname, _countof(jobname), _T("ODK_PRINTJOB"), _TRUNCATE);
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJGetJobProps (hProp, jobname, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJGetJobProps: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        ret = RPJDeletePropertyHandle (hProp, &Error);
        if(ret == FALSE)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJDeletePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
          _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDeletePropertyHandle"));
        }
        ODKTrace(szText);
        ret = RPJDetach(&Error);
```

```
        if (FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
        }
        ODKTrace(szText);
        return;
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   RPJGetJobProps"));
        ODKTrace(szText);
    }
    typ = VT_DATE;
    _tcsncpy_s(jobname, _countof(jobname), _T("ODK_PRINTJOB"), _TRUNCATE);
    _tcsncpy_s(propname, _countof(propname), _T("STARTTIME"), _TRUNCATE);
    memset(&st, 0, sizeof(SYSTEMTIME));
    ptr = (LPVOID)&st;
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = RPJGetProperty (hProp, propname, ptr, (VARTYPE)typ, 16, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJGetProperty: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   RPJGetProperty"));
            ODKTrace(szText);
             _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   Jobname=%s
Propname=%s"),jobname,propname);
            ODKTrace(szText);
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   %02d.%02d.%04d
%02d:%02d:%02d"),
                (WORD)st.wDay,
                (WORD)st.wMonth,
                (WORD)st.wYear,
                (WORD)st.wHour,
                (WORD)st.wMinute,
                (WORD)st.wSecond);
    }
    ODKTrace(szText);
    // write properties
    st.wHour = 11;
    st.wMinute = 12;
```

```
    st.wSecond = 13;
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = RPJSetProperty (hProp, propname, ptr, (VARTYPE) typ, 16, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJSetProperty: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   RPJSetProperty"));
            ODKTrace(szText);
             _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("
Jobname=%s ;Propname=%s"),jobname,propname);
            ODKTrace(szText);
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   %02d.%02d.%04d
%02d:%02d:%02d"),
                (WORD)st.wDay,
                (WORD)st.wMonth,
                (WORD)st.wYear,
                (WORD)st.wHour,
                (WORD)st.wMinute,
                (WORD)st.wSecond);
    }
    ODKTrace(szText);
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = RPJSetJobProps (hProp, jobname, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJSetJobProps: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   RPJSetJobProps"));
    }
    ODKTrace(szText);
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = RPJDeletePropertyHandle (hProp, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJDeletePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDeletePropertyHandle"));
    }
```

```
    ODKTrace(szText);
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = RPJDetach(&Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE,
_T("RPJDeletePropertyHandle"));    }
    ODKTrace(szText);
}
//{{ODK_EXAMPLE}(END)}
```

参见

RPJCreatePropertyHandle (页 2437)

RPJDeletePropertyHandle (页 2439)

RPJGetJobProps (页 2454)

RPJGetProperty (页 2457)

RPJSetProperty (页 2461)

3.7.7.5 显示打印作业预览

示例

```
// ========================================================================
// ========================================================================
// : Modul with examples of Report Designer
// ************************************************************************
// Copyright (C) 1995-99 SIEMENS AG, A&D PT1 D2 All rights reserved
// ************************************************************************
#include "stdafx.h" // if MFC classes
#include "RD02.h"
#include "DM01.h"

//{{ODK_EXAMPLE}Show print job preview (RPT)}
//{{FUNCTION}RPJCreatePropertyHandle (RPT)}
//{{FUNCTION}RPJDeletePropertyHandle (RPT)}
//{{FUNCTION}RPJGetJobProps (RPT)}
//{{FUNCTION}RPJCallJobMethod (RPT)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: MyShowPrintJobPreview( void ) ODK RD RT
// ========================================================================
// : Show print job preview
// ========================================================================
void MyShowPrintJobPreview(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    TCHAR szText[255];
    CMN_ERROR Error;
    HPROPERTIES hProp = NULL;
    TCHAR jobname[200];
    TCHAR methode[200];
     _tcsncpy_s(jobname, _countof(jobname), _T("ODK_PRINTJOB"), _TRUNCATE); // print job
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("--------------------"));
    ODKTrace(szText);
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("MyShowPrintJobPreview:"));
    ODKTrace(szText);
    MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJAttach(&Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJAttach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        return;
```

```
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJAttach"));
        ODKTrace(szText);
    }
    memset(&Error, 0, sizeof(CMN_ERROR));
    hProp = RPJCreatePropertyHandle (/*PROJ_PATH*/g_szProjectFile, &Error);
    if(NULL == hProp)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJCreatePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        ret = RPJDetach(&Error);
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
        }
        ODKTrace(szText);
        return;
    }
    else
    {
         _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJCreatePropertyHandle =
0x%08lx"), hProp);
        ODKTrace(szText);
    }
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJGetJobProps (hProp, jobname, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJGetJobProps: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        ret = RPJDeletePropertyHandle (hProp, &Error);
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJDeletePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
```

```
          _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDeletePropertyHandle"));
        }
        ODKTrace(szText);
        ret = RPJDetach(&Error);
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
        }
        ODKTrace(szText);
        return;
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   RPJGetJobProps"));
        ODKTrace(szText);
    }
    _tcsncpy_s(methode, _countof(methode), _T("PREVIEW"), _TRUNCATE); // preview
    // _tcsncpy_s(methode, _countof(methode), _T("PRINTJOB"), _TRUNCATE); // print
    // _tcsncpy_s(methode, _countof(methode), _T("SETSELECTIONALLPAGES"), _TRUNCATE);
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJCallJobMethod (hProp, methode, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJGetJobProps: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   RPJCallJobMethod"));
    }
    ODKTrace(szText);
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJDeletePropertyHandle (hProp, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
RPJDeletePropertyHandle: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDeletePropertyHandle"));
    }
    ODKTrace(szText);
```

```
    memset(&Error, 0, sizeof(CMN_ERROR));
    ret = RPJDetach(&Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in RPJDetach: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
        {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("RPJDetach"));
    }
    ODKTrace(szText);
}
//{{ODK_EXAMPLE}(END)}
```

参见

RPJCreatePropertyHandle (页 2437)

RPJDeletePropertyHandle (页 2439)

RPJCallJobMethod (页 2448)

RPJGetJobProps (页 2454)

3.8 变量和日志函数

3.8.1 基本知识

3.8.1.1 函数概览

概述

TLG_ENUM_ARCHIV_CALLBACK (页 2572)	列出日志（回调）
TLG_ENUM_PROJECT_NAME_CALLBACK (页 2561)	列出项目（回调）
TLG_ENUM_TIME_NAME_CALLBACK (页 2613)	列出时间对象（回调）

TLG_ENUM_VARIABLE_NAME_CALLBACK (页 2564)	列出变量（回调）
TLG_ENUMBACKUP_ENTRIES (页 2621)	列出备份（回调）
TLG_ENUMTABLES (页 2580)	列出日志（回调）
TLG_ENUMTIMES_CALLBACK (页 2616)	列出时间对象（回调）
TLG_ENUMVARIABLES (页 2567)	列出变量（回调）
TLG_GETARCHIVDATA_CALLBACK (页 2585)	读出日志的数据（回调）
TLGChangeLanguage (页 2551)	语言切换
TLGCloseProject (页 2555)	关闭项目
TLGCloseWindow (页 2602)	关闭“打印作业/脚本诊断”
TLGConnect (页 2552)	建立与变量记录 RT 的连接
TLGCSConnectEx (页 2548)	建立与变量记录 CS 的连接
TLGCSConnect (页 2546)	建立与变量记录 CS 的连接
TLGDisconnect (页 2554)	终止与变量记录 RT 的连接
TLGDrawCurvesInDC (页 2604)	显示趋势
TLGEnumArchives (页 2570)	列出日志
TLGEnumArchivsEx (页 2575)	列出日志
TLGEnumArchivs (页 2573)	列出日志
TLGEnumArchivsSel (页 2577)	列出日志
TLGEnumBackupEntries (页 2619)	列出备份
TLGEnumProject (页 2559)	列出项目
TLGEnumTime (页 2612)	列出时间对象
TLGEnumTimes (页 2615)	列出时间对象
TLGEnumVariablesEx (页 2565)	列出变量
TLGEnumVariables (页 2562)	列出变量
TLGExport (页 2622)	导出日志
TLGFreeMemory (页 2582)	释放内存
TLGGetArchivDataEx (页 2587)	读出日志的数据
TLGGetArchivData (页 2583)	读出日志的数据
TLGGetBackupSize (页 2624)	确定导出的数据记录的大小
TLGGetClosestTimeEx (页 2592)	确定记录时间
TLGGetClosestTime (页 2590)	确定记录时间

TLGInsertArchivData (页 2594)	在日志中插入数据
TLGInsertTemplateItem (页 2605)	将条目写入到趋势窗口模板，将条目写入到表窗口模板
TLGLockArchiv (页 2598)	启用日志，锁定日志
TLGLockVariable (页 2599)	启用变量，锁定变量
TLGOpenProject (页 2557)	打开项目
TLGPressToolbarButton (页 2607)	激活工具栏中的按钮
TLGReadArchiv (页 2601)	读取日志参数
TLGReadTime (页 2618)	确定时间对象参数
TLGReadVariable (页 2568)	确定变量参数
TLGSetRulerWindowVisible (页 2609)	同时显示刻度线窗口
TLGShowWindow (页 2610)	显示“打印作业/脚本诊断”

3.8.1.2 结构概览

概述

TLG_ARCHIV_STR (页 2503)	日志参数
TLG_ARCHIVDATARAW (页 2506)	日志数据
TLG_BACKUP_TABLE_INFO (页 2510)	有关导出表的信息
TLG_CURVESCALEX (页 2512)	X 轴标定
TLG_CURVESCALEY (页 2517)	Y 轴标定
TLG_GETARCHIVDATA (页 2521)	过程值日志的日志数据，压缩日志的日志数据
TLG_IO_BACKUP_SELECT (页 2522)	备份函数的选择标准
TLG_PROT_CURVE_INFOS (页 2523)	趋势的报表结构
TLG_TABLE_INFO (页 2526)	表格的报表结构
TLG_SCAL_STR (页 2525)	变量限制
TLG_TABLESCALE (页 2528)	表格列（属性）
TLG_TEMPLATEITEM_INFO (页 2531)	趋势（属性）表格列（属性）
TLG_TIME_STR (页 2533)	时间对象
TLG_TIMEDATA (页 2535)	时间对象
TLG_TPLITEM_CURVE (页 2536)	趋势（属性）

TLG_TPLITEM_INFO (页 2537)	趋势（属性）表格列（属性）
TLG_TPLITEM_TABLE (页 2538)	表格列（属性）
TLG_VAR_STR (页 2539)	变量（属性）
TLG_VARIABLE_INFO (页 2545)	变量

3.8.1.3 错误消息

概述

CMN_ERROR 错误结构中的 API 函数可返回以下错误消息：

TLG_API_ERROR	0x10000000	未指定的错误
TLG_API_NO_TIME_EXIST	0x10000001	不存在时间对象
TLG_API_TIME_NAME_EXIST	0x10000002	已存在使用该名称的时间对象
TLG_API_TIME_NAME_NOT_EXIST	0x10000003	未找到使用该名称的时间对象
TLG_API_NO_ARCHIV_EXIST	0x10000004	无可用日志
TLG_API_ARCHIV_NAME_EXIST	0x10000005	已存在使用该名称的日志
TLG_API_ARCHIV_NAME_NOT_EXIST	0x10000006	未找到使用该名称的日志
TLG_API_INVALID_ARCHIV_TYP	0x10000007	日志类型无效
TLG_API_NO_VARIABLE_EXIST	0x10000008	不存在任何变量
TLG_API_VARIABLE_NAME_EXIST	0x10000009	已存在使用该名称的变量
TLG_API_VARIABLE_NAME_NOT_EXIST	0x1000000A	未找到使用该名称的变量
TLG_API_VARTYPE_MISMATCH	0x1000000B	变量类型不正确

TLG_API_SRC_NO_VARIABLE_EXIST	0x1000000C	SRC： 不存在任何变量
TLG_API_SRC_VARIABLE_NAME_NOT_EXIST	0x1000000D	SRC： 未找到使用该名称的变量
TLG_API_SRC_ARCHIV_NOT_EXIST	0x1000000E	SRC： 未找到日志
TLG_API_NO_FIELD_EXIST	0x1000000F	无可用字段
TLG_API_FIELD_NAME_EXIST	0x10000010	已存在使用该名称的字段
TLG_API_FIELD_NAME_NOT_EXIST	0x10000011	未找到使用该名称的字段
TLG_API_NO_PROJECT_EXIST	0x10000012	不存在项目
TLG_API_PROJECT_NAME_NOT_EXIST	0x10000013	未找到项目名称
TLG_API_PROJECT_IS_ACTIVE	0x20000014	项目已激活
TLG_API_NO_CONNECTION	0x10000015	未连接到激活的项目
TLG_API_RENAME_NAME_EXIST	0x10000016	分配的名称已存在
TLG_API_NAME_WRONG_CHAR	0x10000017	名称包含错误字符
TLG_API_NAME_TOLONG	0x10000018	名称过长
TLG_API_ERR_ARCHIVSTYLE	0x10000019	日志设计中存在错误
TLG_API_NO_TYP_CHANGE_ALLOWED	0x1000001A	不允许更改类型
TLG_API_ERR_SUPPLY	0x1000001B	发生错误
TLG_API_INVALID_PARAM	0x1000001C	参数分配无效/不正确

TLG_API_NOT_SUPPORTED	0x1000001D	不支持该函数
TLG_API_NO_INTERFACE	0x1000001E	对级别较低的 COM 接口的访问失败
TLG_API_NODEFAULTSERVER	0x10000021	未组态默认服务器
TLG_API_NOLOCALSERVER	0x10000022	无可用本地服务器
TLG_API_NOSERVER	0x10000023	未组态默认服务器，且无可用本地服务器
TLG_API_NOMC	0x10000024	无多客户端项目（此处未使用）
TLG_API_NOMCDEFAULTSERVER	0x10000025	无多客户端项目，但指定“@default::”作为服务器前缀（此处未使用）
TLG_API_UNKNOWN_ERROR	0x10000030	未知回调错误 （WinCC V6.2 及更高版本）
TLG_API_ASYNC_TIMEOUT	0x10000031	异步函数由于超时而取消 （WinCC V6.2 及更高版本）
TLG_API_EXPORT_NO_WRITE	0x10000032	导出文件未写入 （WinCC V6.2 及更高版本）
TLG_API_CORRUPT_DATA	0x10000033	数据不可用/存在错误 （WinCC V6.2 及更高版本）
IDS_API_ERROR_CREATING_CLIENT_WND	30001	无法生成客户端窗口
IDS_API_ERROR_WINDOW_NOT_FOUND	30002	无法找到按钮（窗口）

3.8.1.4 常数

运行系统窗口

TLG_CURVE_WINDOW	0x00000001	趋势窗口
TLG_TABLE_WINDOW	0x00000002	表格窗口
TLG_CTRL_WINDOW	0x00000004	TLGRT 控制窗口

导出函数

TLG_BACKUP_EXPORT	0x00000001	导出而不删除源数据。
TLG_BACKUP_EVACUATE	0x00000002	导出且删除源数据。
TLG_BACKUP_DELETE	0x00000004	不导出但删除数据记录。
TLG_BACKUP_RAW	0x00000008	不要单独处理有关更改的日志。

导出/导入格式

TLG_BAKFMT_CSV	0x00000001	CSV 格式（逗号分隔值）

导入函数

TLG_RESTORE_OVERWRITE	0x00000001	导入时覆盖。
TLG_RESTORE_MERGE	0x00000002	导入时通过回调检查错误。
TLG_RESTORE_BREAK	0x00000004	导入由于出错而取消

日志类型

TLG_ARCTYP_USER	0x00000001	用户日志 （WinCC V5.0 及更高版本不再支持）
TLG_ARCTYP_PROCESS	0x00000002	过程值日志
TLG_ARCTYP_COMPRESS	0x00000004	压缩日志

日志类型

TLG_ARCTYP_ALL	0x00000000	循环或顺序日志
TLG_ARCTYP_CIRCULAR	0x00010000	循环日志
TLG_ARCTYP_FOLLOW	0x00020000	顺序日志

日志标记

TLG_API_FLG_FAST_INSERT	0x00000001	WinCC V6.0 SP1 及更高版本： 使用 TLGInsertArchivData 在 SQL Server 中快速插入数据

窗口模板

TLG_TEMPLATE_ALL	0x00000000	所有窗口模板
TLG_TEMPLATE_CURVE	0x00000001	趋势窗口模板
TLG_TEMPLATE_TABLE	0x00000002	表格窗口模板
TLG_TYP_TEMPLATE_CURVE	0x00800001	趋势窗口模板
TLG_TYP_TEMPLATE_TABLE	0x00800002	表格窗口模板

模板

TLG_TEMPLATEITEM_ALL	0x00000000	所有模板项
TLG_TEMPLATEITEM_CURVE	0x00000001	趋势模板项
TLG_TEMPLATEITEM_TABLE	0x00000002	表格模板项

修改模板项

TLG_TI_ACTION_COLOR	0x00000001	更改颜色
TLG_TI_ACTION_SCALE_X	0x00000002	更改 X 标定
TLG_TI_ACTION_SCALE_Y	0x00000004	更改 Y 标定
TLG_TI_ACTION_VISIBLE	0x00000005	设置可见/不可见
TLG_TI_ACTION_RANGE_X	0x00000006	显示 X 轴范围
TLG_TI_ACTION_RANGE_Y	0x00000007	显示 Y 轴范围

限值

TLG_MAX_SQL_SELECT	512	SQL 命令的最大长度
TLG_MAX_TEMPLATE_NAME	32	趋势或表格窗口模板名称的最大长度
TLG_MAX_TEMPLATEITEM_NAME	32	趋势或表格模板名称的最大长度
TLG_MAX_FUNCTION_NAME	32	动作或函数名称的最大长度
TLG_MAX_DLL_NAME	32	DLL 名称的最大长度
TLG_MAX_STD_TEXT_NAME	32	TLGSetTemplateData
TLG_SETDATA_DEFAULT	0x00000000	y=f(x)

TLG_SETDATA_RESET	0x00000001	清空趋势缓冲区。
TLG_SETDATA_TIME_RANGE_X	0x00000002	x 值必须解释为时间范围： y=f(t)。
TLG_SETDATA_TIME_RANGE_Y	0x00000004	y 值必须解释为时间范围： x=f(t)。

趋势形状

TLG_CURVEFORM_STEP	0x00000011	变量值表示为阶跃趋势
TLG_CURVEFORM_POINTS	0x00000012	变量值表示为单个点。
TLG_CURVEFORM_DIRECT	0x00000014	变量值的线性插值
TLG_CURVEFORM_ DIRFILL	0x00000021	
TLG_CURVEFORM_ STEPFILL	0x00000022	

数据范围类型

TLG_DATATYP_TIMERANGE	0x00000001	
TLG_DATATYP_USERARCHIV	0x00000002	
TLG_DATATYP_BLOCKDATA	0x00000004	

时间范围

TLG_TR_MINUTE	1	
TLG_TR_HOUR	2	
TLG_TR_DAY	3	

TLG_TR_WEEK	4	
TLG_TR_MONTH	5	
TLG_TR_YEAR	6	

轴

TLG_AXIS_X_TOP	0x00000001	
TLG_AXIS_X_MIDDLE	0x00000002	
TLG_AXIS_X_BOTTOM	0x00000004	
TLG_AXIS_Y_LEFT	0x00000001	
TLG_AXIS_Y_MIDDLE	0x00000002	
TLG_AXIS_Y_RIGHT	0x00000004	

协议连接

TLG_PROTFLG_TIME_RANGE	0x00000001	
TLG_PROTFLG_DATA_RANGE	0x00000002	
TLG_PROTFLG_DATA	0x00000001	
TLG_PROTFLG_TITLE	0x00000002	
TLG_PROTFLG_TIMEFIELD	0x00000004	
TLG_PROTFLG_DATACOUNT	0x00000008	
TLG_PROTFLG_DATEFIELD	0x00000010	
TAG_PROT_CURVE_FORM_STEP	0x00000001	
TAG_PROT_CURVE_FORM_POINTS	0x00000002	

TAG_PROT_CURVE_FORM_DIRECT	0x00000003	
TAG_PROT_CURVE_FORM_AREA	0x00000004	
TLG_PROTFLG_FIRST_COL	0x00000000	
TLG_PROTFLG_FIRST_COL_TIME	0x00000001	
TLG_PROTFLG_FIRST_COL_DATA	0x00000002	
TLG_PROTFLG_FIRST_COL_DATE	0x00000004	

读取日志数据

TLG_DATA_FROM_POS	0x00000000	
TLG_DATA_FROM_BEGIN	0x00000001	
TLG_DATA_FROM_END	0x00000002	
TLG_DATA_FROM_ACTUAL	0x00000004	
TLG_DATA_FROM_LEFT_ABS	0x00000008	
TLG_DATA_FROM_RIGHT_ABS	0x00000010	

运行系统工具栏上的按钮

用于趋势和表格的按钮

TLG_BASIC_BTN_HELP	0x00000001	用于调用在线帮助的按钮的标识
TLG_BASIC_BTN_DLG	0x00000002	用于打开趋势和表格窗口模板参数分配对话框的按钮的标识。

TLG_BASIC_BTN_FIRST	0x00000004	用于显示日志前几条数据记录的按钮的标识。
TLG_BASIC_BTN_PREV	0x00000008	用于在日志中进行回滚的按钮的标识。
TLG_BASIC_BTN_NEXT	0x00000010	用于在日志中进行前滚的按钮的标识。
TLG_BASIC_BTN_LAST	0x00000020	用于显示日志最后几条数据记录的按钮的标识。
TLG_BASIC_BTN_STARTSTOP	0x00000040	用于暂停或恢复更新的趋势或表格显示画面的按钮的标识。
TLG_BASIC_BTN_PREV_ITEM	0x00000800	前景或表格第一列中的上一趋势。
TLG_BASIC_BTN_NEXT_ITEM	0x00001000	前景或表格第一列中的下一趋势。
TLG_BASIC_BTN_ARC_VAR_SELECT	0x00000100	打开日志变量选择对话框。
TLG_BASIC_BTN_ITEM_SELECT	0x00000200	打开趋势或表格列的选择对话框。
TLG_BASIC_BTN_TIME_SELECT	0x00000400	打开显示时间范围的选择对话框。

趋势按钮

TLG_CURVE_BTN_ZOOMIN	0x01000000	用于激活要缩放区域的缩放功能的按钮的标识。 只能使用鼠标定义缩放部分的大小！
TLG_CURVE_BTN_ZOOMOUT	0x02000000	用于禁用缩放功能的按钮的标识。 这不会导致切换回常规视图。
TLG_CURVE_BTN_LINEAL	0x04000000	用于激活和禁用标尺的按钮的标识，该标尺用于确定测量点坐标。
TLG_CURVE_BTN_1_TO_1	0x08000000	用于切换到常规趋势视图的按钮的标识。

表格按钮

TLG_TABLE_BTN_EDIT	0x00040000	用于激活和禁用编辑功能的按钮的标识。

结构的字段定义

PDE_DB_PRJ_PROJNAME_MAX_LENGHT TLG_DB_PRJ_PROJNAME_MAX_LENGHT	128	
PDE_DB_PRJ_COMMENT_MAX_LENGHT TLG_DB_PRJ_COMMENT_MAX_LENGHT	200	
PDE_DB_PRJ_USERDATE_MAX_LENGHT TLG_DB_PRJ_USERDATE_MAX_LENGHT	12	
PDE_DB_PRJ_USERNAME_MAX_LENGHT TLG_DB_PRJ_USERNAME_MAX_LENGHT	30	
PDE_DB_ARC_NAME_MAX_LENGHT TLG_DB_ARC_NAME_MAX_LENGHT	32	
PDE_DB_ARC_NAME_MAX_LENGHT TLG_DB_ARC_NAME_MAX_LENGHT	32	
TLG_DB_ARC_USER_NAME_MAX_LENGHT	8	
PDE_DB_ARC_SERVERNAME_MAX_LENGHT TLG_DB_ARC_SERVERNAME_MAX_LENGHT	50	
PDE_DB_ARC_COMMENT_MAX_LENGHT TLG_DB_ARC_COMMENT_MAX_LENGHT	100	
PDE_DB_ARC_ACTIONNAME_MAX_LENGHT TLG_DB_ARC_ACTIONNAME_MAX_LENGHT	30	
PDE_DB_VAR_NAME_MAX_LENGHT TLG_DB_VAR_NAME_MAX_LENGHT	64	
TLG_DB6_VAR_NAME_MAX_LENGHT	128	
TLG_DB_MCP_VAR_MAME_MAX_LENGTH	128	
PDE_DB_VAR_COMMENT_MAX_LENGHT TLG_DB_VAR_COMMENT_MAX_LENGHT	100	
PDE_DB_VAR_WRITEBACK_MAX_LENGHT TLG_DB_VAR_WRITEBACK_MAX_LENGHT	64	
TLG_DB6_VAR_WRITEBACK_MAX_LENGHT	128	
PDE_DB_VAR_CYCLENAME_MAX_LENGHT TLG_DB_VAR_CYCLENAME_MAX_LENGHT	30	

PDE_DB_VAR_SIGNALTEX_MAX_LENGHT TLG_DB_VAR_SIGNALTEX_MAX_LENGHT	30	
PDE_DB_VAR_DLLNAME_MAX_LENGHT TLG_DB_VAR_DLLNAME_MAX_LENGHT	30	
PDE_DB_VAR_FUNCNAME_MAX_LENGHT TLG_DB_VAR_FUNCNAME_MAX_LENGHT	30	
PDE_DB_VAR_UNITDIR_MAX_LENGHT TLG_DB_VAR_UNITDIR_MAX_LENGHT	30	
PDE_DB_VAR_UNITSTR_MAX_LENGHT TLG_DB_VAR_UNITSTR_MAX_LENGHT	64	
TLG_DB6_VAR_UNITSTR_MAX_LENGHT	128	
PDE_DB_FUNCNAME_MAX_LENGHT TLG_DB_FUNCNAME_MAX_LENGHT	30	
PDE_DB_DLLNAME_MAX_LENGHT TLG_DB_DLLNAME_MAX_LENGHT	30	
PDE_DB_SCALE_MAX_LENGHT TLG_DB_SCALE_MAX_LENGHT	30	
PDE_DB_TIMENAME_MAX_LENGHT TLG_DB_TIMENAME_MAX_LENGHT	30	
PDE_DB_FIELDNAME_MAX_LENGHT TLG_DB_FIELDNAME_MAX_LENGHT	64	
PDE_DB_STRUCTNAME_MAX_LENGHT TLG_DB_STRUCTNAME_MAX_LENGHT	30	

日志变量的 VARIANT 记录的字段定义

有关 TLGCSCreateTagMulti、TLGCSReadTagMulti 和 TLGCSModifyTagMulti，

请参见 Include 文件 pde_typ.h

从 WinCC V6.2 开始提供这些函数。

ENUM_TLG_TAG_FIELDINDEX 中的定义	字段索引	关联的类型定义	类型	注释/限制
ENUM_TLG_TAG_ENUMTYPE	0	VT_TLG_TAG_ENUMTYPE	VT_I4	已内部设置（值 >= 8），无法更改
ENUM_TLG_TAG_VERSION	1	VT_TLG_TAG_VERSION	VT_I4	已内部设置（值 >= 1），无法更改
ENUM_TLG_TAG_CREATORID	2	VT_TLG_TAG_CREATORID	VT_I4	
ENUM_TLG_TAG_NAME	3	VT_TLG_TAG_NAME	VT_BSTR	“创建”所必需
ENUM_TLG_TAG_ARCHIVNAME	4	VT_TLG_TAG_ARCHIVNAME	VT_BSTR	“创建”所必需
ENUM_TLG_TAG_TYPE	5	VT_TLG_TAG_TYPE	VT_I4	“创建”所必需
ENUM_TLG_TAG_WRITEBACKNAME	6	VT_TLG_TAG_WRITEBACKNAME	VT_BSTR	
ENUM_TLG_TAG_COMMENT	7	VT_TLG_TAG_COMMENT	VT_BSTR	
ENUM_TLG_TAG_ARCTYPE	8	VT_TLG_TAG_ARCTYPE	VT_I4	
ENUM_TLG_TAG_SUPPLYTYPE	9	VT_TLG_TAG_SUPPLYTYPE	VT_I4	
ENUM_TLG_TAG_LOCKED	10	VT_TLG_TAG_LOCKED	VT_I4	
ENUM_TLG_TAG_SCANTIME	11	VT_TLG_TAG_SCANTIME	VT_BSTR	“创建”所必需
NUM_TLG_TAG_ARCTIME	12	VT_TLG_TAG_ARCTIME	VT_BSTR	“创建”所必需
ENUM_TLG_TAG_ARCFACTOR	13	VT_TLG_TAG_ARCFACTOR	VT_I4	“创建”所必需

ENUM_TLG_TAG_FIELDINDEX 中的定义	字段索引	关联的类型定义	类型	注释/限制
ENUM_TLG_TAG_VARCOUNTFORWARD	14	VT_TLG_TAG_VARCOUNTFORWARD	VT_I4	
ENUM_TLG_TAG_VARCOUNTBACKWARD	15	VT_TLG_TAG_VARCOUNTBACKWARD	VT_I4	
ENUM_TLG_TAG_SAVEBYFAULT	16	VT_TLG_TAG_SAVEBYFAULT	VT_I4	
ENUM_TLG_TAG_ARCTRIGGER	17	VT_TLG_TAG_ARCTRIGGER	VT_I4	“创建”所必需
ENUM_TLG_TAG_STATETEXTHIGH	18	VT_TLG_TAG_STATETEXTHIGH	VT_BSTR	
ENUM_TLG_TAG_STATETEXTLOW	19	VT_TLG_TAG_STATETEXTLOW	VT_BSTR	
ENUM_TLG_TAG_VARPRO	20	VT_TLG_TAG_VARPRO	VT_I4	
ENUM_TLG_TAG_FUNCVALPRO	21	VT_TLG_TAG_FUNCVALPRO	VT_BSTR	
ENUM_TLG_TAG_DLLVARPRO	22	VT_TLG_TAG_DLLVARPRO	VT_BSTR	
ENUM_TLG_TAG_FUNCSTARTEVENT	23	VT_TLG_TAG_FUNCSTARTEVENT	VT_BSTR	
ENUM_TLG_TAG_DLLSTARTEVENT	24	VT_TLG_TAG_DLLSTARTEVENT	VT_BSTR	
ENUM_TLG_TAG_FUNCSTOPEVENT	25	VT_TLG_TAG_FUNCSTOPEVENT	VT_BSTR	

ENUM_TLG_TAG_FIELDINDEX 中的定义	字段索引	关联的类型定义	类型	注释/限制
ENUM_TLG_TAG_DLLSTOPEVENT	26	VT_TLG_TAG_DLLSTOPEVENT	VT_BSTR	
ENUM_TLG_TAG_FUNCDYNAMIC	27	VT_TLG_TAG_FUNCDYNAMIC	VT_BSTR	
ENUM_TLG_TAG_DLLDYNAMIC	28	VT_TLG_TAG_DLLDYNAMIC	VT_BSTR	
ENUM_TLG_TAG_UNITDIRECT	29	VT_TLG_TAG_UNITDIRECT	VT_BSTR	
ENUM_TLG_TAG_NAMEUNITSTRUCT	30	VT_TLG_TAG_NAMEUNITSTRUCT	VT_BSTR	
ENUM_TLG_TAG_UNITINDEX	31	VT_TLG_TAG_UNITINDEX	VT_I4	
ENUM_TLG_TAG_VARUPPER	32	VT_TLG_TAG_VARUPPER	VT_R8	
ENUM_TLG_TAG_VARLOWER	33	VT_TLG_TAG_VARLOWER	VT_R8	
ENUM_TLG_TAG_ARCUPPER	34	VT_TLG_TAG_ARCUPPER	VT_R8	
ENUM_TLG_TAG_ARCLOWER	35	VT_TLG_TAG_ARCLOWER	VT_R8	
ENUM_TLG_TAG_NAMESCALE	36	VT_TLG_TAG_NAMESCALE	VT_BSTR	
ENUM_TLG_TAG_NAMESOURCEARC	37	VT_TLG_TAG_NAMESOURCEARC	VT_BSTR	
ENUM_TLG_TAG_NAMESOURCEVAR	38	VT_TLG_TAG_NAMESOURCEVAR	VT_BSTR	
ENUM_TLG_TAG_NAMERAWVAR	39	VT_TLG_TAG_NAMERAWVAR	VT_BSTR	

ENUM_TLG_TAG_FIELDINDEX 中的定义	字段索引	关联的类型定义	类型	注释/限制
ENUM_TLG_TAG_CONVDLL	40	VT_TLG_TAG_CONVDLL	VT_BSTR	
ENUM_TLG_TAG_PROCVARNAME	41	VT_TLG_TAG_PROCVARNAME	VT_BSTR	“创建”所必需
ENUM_TLG_TAG_TIMEMODIFY	42	VT_TLG_TAG_TIMEMODIFY	VT_BSTR	已内部设置（用于“创建”和“修改”的新时间戳），无法更改
ENUM_TLG_TAG_ARCONCHANGE	43	VT_TLG_TAG_ARCONCHANGE	VT_I4	
ENUM_TLG_TAG_ARCONHYSTERESE	44	VT_TLG_TAG_ARCONHYSTERESE	VT_R8	
ENUM_TLG_TAG_FLAGS	45	VT_TLG_TAG_FLAGS	VT_I4	
ENUM_TLG_TAG_ALIAS	46	VT_TLG_TAG_ALIAS	VT_BSTR	
ENUM_TLG_TAG_VARSTARTEVENT	47	VT_TLG_TAG_VARSTARTEVENT	VT_BSTR	
ENUM_TLG_TAG_VARSTOPEVENT	48	VT_TLG_TAG_VARSTOPEVENT	VT_BSTR	
ENUM_TLG_TAG_PRECISION	49	VT_TLG_TAG_PRECISION	VT_BSTR	
ENUM_TLG_TAG_TAGID	50	VT_TLG_TAG_TAGID	VT_I4	已内部设置，无法更改

日志变量的标记

请参见 Include-Datei pde_typ.h ENUM_TLG_TAG_FLAGVALUES WinCC V6.2 及更高版本

ENUM_TLG_TAG_FLAGVALUE_NOFLAGS	0	默认值
ENUM_TLG_TAG_FLAGVALUE_LONGTERM_DISABLE	1	
ENUM_TLG_TAG_FLAGVALUE_ARCHIVING_ON_SEGMENTCHANGE	2	

TLGCSReadTagMulti 和 TLGCSModifyTagMulti 的选择标记

从 WinCC V6.2 开始提供这些函数。

TLG_TAG_SELECTION_BY_NAME	1	在各种情况下均通过日志名称和变量名称进行选择。
---	---	---
TLG_TAG_DEFAULT_SELECTION	TLG_TAG_SELECTION_BY_NAME	首选选项类型。
TLG_TAG_SELECTION_FLAGS_MASK	0x0000000FL	屏蔽标记中的选项。
TLG_TAG_SELECTION_FLAGS_OFFSET	0x0L	在标记中进行屏蔽后所选内容的值偏移量。

3.8.2 结构

3.8.2.1 TLG_ARCHIV_STR

声明

```
typedef struct {
    TCHAR        szName[
                PDE_DB_ARC_NAME_MAX_LENGHT+1];
    TCHAR        szComment[
                PDE_DB_ARC_COMMENT_MAX_LENGHT+1];
    TCHAR        szServername[
                PDE_DB_ARC_SERVERNAME_MAX_LENGHT +1];
    DWORD        dwTyp;
    DWORD        dwAccessRead;
    DWORD        dwAccessWrite;
    TCHAR        szArchivAction[
                PDE_DB_ARC_ACTIONNAME_MAX_LENGHT+1];
    BOOL         fLocked;
    DWORD        dwRecordTyp;
    DWORD        dwFillMessage;
    DWORD        dwRecordSize;
    DWORD        dwStorage;
    TCHAR        szCircularAction[
                PDE_DB_ARC_ACTIONNAME_MAX_LENGHT+1];
    TCHAR        szCompressTime[
                PDE_DB_TIMENAME_MAX_LENGHT+1];
    DWORD        dwSourceProcess;
    TCHAR        szRawDatVar[
                PDE_DB_VAR_NAME_MAX_LENGHT+1];
    TCHAR        szSendAct[
                PDE_DB_ARC_ACTIONNAME_MAX_LENGHT+1];
    TCHAR        szRecAct[
                PDE_DB_ARC_ACTIONNAME_MAX_LENGHT+1];
    DWORD        dwRecItems;
    }
    TLG_ARCHIV_STR;
```

成员

szName

日志名称的全局参数

szComment

项目注释的全局参数

szServername

为将来升级保留的全局参数

dwTyp

标识日志类型的全局参数。

TLG_ARCHIV_TYP_PROCESS	过程值日志
TLG_ARCHIV_TYP_COMPRESS	压缩日志

dwAccessRead

标识读访问授权级别的全局参数 (0...999)。

dwAccessWrite

标识写访问授权级别的全局参数 (0...999)。

szArchivAction

带有常规日志处理所用操作名称的全局参数。

fLocked

将日志标识为已释放或已锁定的全局参数。

dwRecordTyp

标识日志类型的循环和顺序日志参数。

TLG_RECORD_TYP_CIRCULAR	循环日志
TLG_RECORD_TYP_FOLLOW	顺序日志

dwFillMessage

用于循环、连续和压缩日志的参数，可确认何时发送填充级别消息。使用逻辑 OR 操作最多可组态两个填充级别消息。

例如：100% || 50%-90%)

TLG_FILL_MESSAGE_NO	无填充量消息
TLG_FILL_MESSAGE_50	填充量消息，已达 50%
TLG_FILL_MESSAGE_60	填充量消息，已达 60%
TLG_FILL_MESSAGE_70	填充量消息，已达 70%
TLG_FILL_MESSAGE_80	填充量消息，已达 80%

TLG_FILL_MESSAGE_90	填充量消息，已达 90%
TLG_FILL_MESSAGE_100	填充量消息，已达 100%

dwRecordSize

定义数据记录的周期缓冲区大小的循环日志参数。

dwStorage

标识日志位置的循环日志参数。

TLG_STORAGE_MEMORY	主存储器中的循环日志
TLG_STORAGE_HD	硬盘上的循环日志

szCircularAction

循环日志参数，循环日志数据导出操作的名称。

szCompressTime

压缩日志参数，指定压缩时间段的定时器对象的名称。

dwSourceProcess

标识源日志处理类型的压缩日志参数。

PDE_COMPSRC_CALC	计算
PDE_COMPSRC_CALCCOPY	计算并复制
PDE_COMPSRC_CALCDEL	计算并删除
PDE_COMPSRC_CALCCOPYDEL	计算、复制并删除

szRawDatVar

该参数为将来开发预留，必须预设为 0。

szSendAct

该参数为将来开发预留，必须预设为 0。

szRecAct

该参数为将来开发预留，必须预设为 0。

dwRecItems

该参数为将来开发预留，必须预设为 0。

注释

TLG_ARCHIV_STR 结构的参数取决于日志类型（过程日志或压缩日志）以及记录的类型（循环日志或顺序日志）。

所需文件

pde_def.h

API 函数

TLGReadArchiv (页 2601)	读取日志参数

参见

TLGReadArchiv (页 2601)

3.8.2.2 TLG_ARCHIVDATARAW

声明

```
typedef struct {
    SYSTEMTIME    stTime;
    double        doValue;
    DWORD         dwFlags;
    }
    TLG_ARCHIVDATARAW;
```

成员

stTime

stTime 包含记录时间。

doValue

doValue 包含 stTime 时存在的值。

dwFlags

变量记录会为写入日志的每个值都设置标记，以提供变量状态信息。

该值必须转换为十六进制格式，以便分析标记。其中

- 左侧字 (HighWord) 包含数据管理器的标志
- 左侧字（高字）的质量代码（如果 PDE_RT_QUALITYCODE 已置位）
- 右侧字（低字）为来自变量记录的状态标记

变量记录状态标记（低字）：		
PDE_RT_DAYLIGHT	0x0001	夏令时，状态可通过 GetTimeZomeInformation 确定
PDE_RT_SUBSTITUTION	0x0002	替换值
PDE_RT_TIME_BEVOR_JUMP	0x0004	时间跳跃前的值
PDE_RT_TIME_BEHIND_JUMP	0x0008	时间跳跃后的值
PDE_RT_TIME_OVERLAPPED	0x0010	时间重叠期间的值
PDE_RT_LOAD_SYSTEM	0x0020	首次创建日志归档值后
PDE_RT_RELOAD_SYSTEM	0x0040	激活运行系统后的第一个值
PDE_RT_CMPCOPY	0x0080	压缩值
PDE_RT_TIME_CHANGED	0x0100	时间发生改变
PDE_RT_HAND	0x0200	提供手动变量
PDE_RT_ONCHANGEBACKUPVALUESTART	0x0400	
PDE_RT_ONCHANGEBACKUPVALUESTOP	0x0800	
PDE_RT_QUALITYCODE	0x1000	

数据管理器状态标志 (HighWord)		
DM_VARSTATE_NOT_ESTABLISHED	0x0001	未与伙伴建立连接
DM_VARSTATE_HANDSHAKE_ERROR	0x0002	协议错误
DM_VARSTATE_HARDWARE_ERROR	0x0004	网络模块故障
DM_VARSTATE_MAX_LIMIT	0x0008	违反了组态的上限
DM_VARSTATE_MIN_LIMIT	0x0010	违反了组态的下限
DM_VARSTATE_MAX_RANGE	0x0020	超出格式上限
DM_VARSTATE_MIN_RANGE	0x0040	超出格式下限
DM_VARSTATE_CONVERSION_ERROR	0x0080	显示转换错误和 MAX/MIN_RANGE
DM_VARSTATE_STARTUP_VALUE	0x0100	变量的初始化值
DM_VARSTATE_DEFAULT_VALUE	0x0200	变量的替换值
DM_VARSTATE_ADDRESS_ERROR	0x0400	通道寻址错误

数据管理器状态标志 (HighWord)		
DM_VARSTATE_INVALID_KEY	0x0800	变量未找到或不存在
DM_VARSTATE_ACCESS_FAULT	0x1000	不允许访问变量
DM_VARSTATE_TIMEOUT	0x2000	超时，通道无反馈
DM_VARSTATE_SERVERDOWN	0x4000	与服务器的连接已中断，或服务器停止运行

	WinCC 变量状态	基于 Profibus PA/OPC 的质量代码
DM_VARSTATE_SERVERDOWN (0x4000)	服务器停机	Bad, out of service, 0x1C
DM_VARSTATE_HARDWARE_ERROR (0x0004)	网络模块故障	Bad, device failure, 0x0C
DM_VARSTATE_NOT_ESTABLISHED (0x0001)	未与伙伴建立连接	Bad, not connected, 0x08
DM_VARSTATE_TIMEOUT (0x2000)	超时，通道无反馈	Uncertain, last usable value, 0x44
DM_VARSTATE_HANDSHAKE_ERROR (0x0002)	协议错误	Bad, no communication (no usable value), 0x18
DM_VARSTATE_ADDRESS_ERROR (0x0400)	通道寻址错误	Bad, configuration error, value not accepted, 0x04
DM_VARSTATE_INVALID_KEY (0x0800)	变量未找到或不存在	Bad, configuration error, value not accepted, 0x04
DM_VARSTATE_ACCESS_FAULT (0x1000)	不允许访问变量	Bad, configuration error, value not accepted, 0x04
DM_VARSTATE_MAX_RANGE (0x0020)	超出格式上限	Uncertain, engineering unit range violation, high limit 设置 , 0x56
DM_VARSTATE_MIN_RANGE (0x0040)	超出格式下限	Uncertain, engineering unit range violation, low limit 设置 , 0x55
DM_VARSTATE_CONVERSION_ERROR (0x0080)	显示转换错误	Uncertain, engineering unit range violation, 设置无限制 , 0x54
DM_VARSTATE_MAX_LIMIT (0x0008)	违反了组态的上限	限制位 HIGH, 0x01 与其他变量状态的或运算仿真，例如给出 "good (cascade), ok" 0xC2

	WinCC 变量状态	基于 Profibus PA/OPC 的质量代码
DM_VARSTATE_MIN_LIMIT (0x0010)	违反了组态的下限	限制位 LOW, 0x02 与其他变量状态的或运算仿真，例如给出 "good (cascade), ok" 0xC1
DM_VARSTATE_STARTUP_VALUE (0x0100)	变量的初始化值	Uncertain, initial value, 0x4C
DM_VARSTATE_DEFAULT_VALUE (0x0200)	变量的替换值	Uncertain, substitute-set, 0x48

注释

如果需要一个用于传递 SYSTEMTIME 参数的当前定时器，则必须使用 GetLocalTime，而非 GetSystemTime。通常，这两个函数的时间有很大差异。

如果相应的时间处于夏令时期间，则置位 PDE_RT_DAYLIGHT 标记。可使用 GetTimeZoneInformation 系统函数 (TIME_ZONE_ID_DAYLIGHT) 确定标记的状态。

所需文件

pdertdef.h

pdert.h

dmdefs.h

API 函数

TLGGetArchivDataEx (页 2587)	读出日志的数据

参见

TLGGetArchivDataEx (页 2587)

3.8.2.3 TLG_BACKUP_TABLE_INFO

声明

```
typedef struct {
    TCHAR         szArchivName[ _MAX_PATH + 1 ];
    TCHAR         szFileName[ _MAX_PATH + 1 ];
    TCHAR         szComment[ _MAX_PATH + 1 ];
    DWORD         dwFormatFlags;
    DWORD         dwJobFlags;
    DWORD         dwSize;
    SYSTEMTIME    sysFrom;
    SYSTEMTIME    sysTo;
    DWORD         dwUserData;
    }
    TLG_BACKUP_TABLE_INFO;
```

成员

szArchivName

要导出数据的日志的名称。

szFileName

含路径和扩展名的导出文件的名称

szComment

有关导出的注释文本

dwFormatFlags

格式说明符：

TLG_BAKFMT_CSV:	CSV 格式（逗号分隔值）

dwJobFlags

特定作业标识符，可能的情况如下：

TLG_BACKUP_EXPORT:	仅备份，不删除源数据
TLG_BACKUP_EVACUATE:	导出并删除源数据

dwSize

待导出数据的大小

sysFrom

数据导出的开始时间

sysTo

数据导出的结束时间

dwUserData

应用程序特定的数据

所需文件

pdertdef.h

API 函数

TLG_ENUMBACKUP_ENTRIES (页 2621)	列出备份（回调）

参见

TLG_ENUMBACKUP_ENTRIES (页 2621)

3.8.2.4 TLG_CURVESCALEX

声明

```
typedef struct {
    DWORD         dwDataTyp;
    DWORD         dwBufferSize;
    DWORD         dwRangeTyp;
    DWORD         dwAxisLocation;
    BOOL          fActualize;
    BOOL          fAutoRange;
    BOOL          fGridLinesBig;
    BOOL          fGridLinesFine;
    BOOL          fGridLinesBigVisible;
    BOOL          fGridLinesFineVisible;
    BOOL          fPercent;
    BOOL          fLimitRange;
    BOOL          fSubstitute;
    SYSTEMTIME    stFrom;
    SYSTEMTIME    stTo;
    double        doFrom;
    double        doTo;
    double        doGridBig;
    double        doGridFine;
    double        doShowDigits;
    double        doLimitUpper1;
    double        doLimitUpper2;
    double        doLimitUpper3;
    double        doLimitLower1;
    double        doLimitLower2;
    double        doLimitLower3;
    double        doDisplayRangeFrom;
    double        doDisplayRangeTo;
    double        doLimitRangeFrom;
    double        doLimitRangeTo;
    COLORREF      crColor;
    COLORREF      crColorTimeOverlapped;
    COLORREF      crColorTimeJump;
    COLORREF      crColorLimitUpper1;
    COLORREF      crColorLimitUpper2;
    COLORREF      crColorLimitUpper3;
    COLORREF      crColorLimitLower1;
    COLORREF      crColorLimitLower2;
    COLORREF      crColorLimitLower3;
    TCHAR         szSelectSQL[ TLG_MAX_SQL_SELECT ];
    TCHAR         szText[ TLG_MAX_STD_TEXT_NAME ];
    TCHAR         szFunction[ TLG_MAX_FUNCTION_NAME ];
    TCHAR         szDLL[ TLG_MAX_DLL_NAME ];
    }
```

```
TLG_CURVESCALEX;
```

成员

dwDataTyp

dwDataTyp 可确定趋势所基于的数据类型：

TLG_DATATYP_TIMERANGE	0x00000001	对于时间范围： 数据来自于受时间控制的日志（例如：过程值日志）。stFrom 和 stTo 中指定的值有效。然后 fAutoRange 标记便无效。 其它应用方面： 数据来自于受时间控制的日志（例如：用户日志）。doFrom 和 doTo 中指定的值有效。
TLG_DATATYP_USERARCHIV	0x00000002	与趋势无关： 数据来自于受数据控制的用户日志。doFrom 和 doTo 中指定的值有效。
TLG_DATATYP_BLOCKDATA	0x00000004	由 TLGSetTemplateData 设置数据： 然后便与 stFrom、stTo、doFrom 和 doTo 中的值无关。

dwBufferSize

dwBufferSize 对应于趋势的测量点数量，并确定趋势数据缓冲区的大小。

dwRangeTyp

该参数为将来开发预留，必须预设为 0。

dwAxisLocation

该参数为将来开发预留，必须预设为 0。

fActualize

当 fActualize = TRUE 时，趋势将动态显示。否则为静态。

fAutoRange

如果 fAutoRange = TRUE，则自动缩放将激活。

如果存在受时间控制的日志 (dwDataTyp = TLG_DATATYP_TIMERANGE)，则 fAutoRange 无效，stFrom 和 stTo 中指定的值有效。

fGridLinesBig

该参数为将来开发预留，必须预设为 0。

fGridLinesFine

该参数为将来开发预留，必须预设为 0。

fGridLinesBigVisible

该参数为将来开发预留，必须预设为 0。

fGridLinesFineVisible

该参数为将来开发预留，必须预设为 0。

fPercent

该参数为将来开发预留，必须预设为 0。

fLimitRange

该参数为将来开发预留，必须预设为 0。

fSubstitute

该参数为将来开发预留，必须预设为 0。

stFrom

显示范围的开始时间。只有在 dwDataTy = TLG_TIME_RANGE 时才会评估此参数。

stTo

显示范围的结束时间。只有在 dwDataTyp = TLG_TIME_RANGE 时才会评估此参数。

如果需要一个用于传递 SYSTEMTIME 参数的当前定时器，则必须使用 GetLocalTime，而非 GetSystemTime。通常，这两个函数的时间有很大差异。

doFrom

显示范围的起始值。只有在 dwDataTyp = TLG_DATA_RANGE 时才会评估此参数。

doTo

显示范围的结束值。只有在 dwDataTyp = TLG_DATA_RANGE 时才会评估此参数。

doGridBig

该参数为将来开发预留，必须预设为 0。

doGridFine

该参数为将来开发预留，必须预设为 0。

doShowDigits

小数位数 (Number of decimal places)

doLimitUpper1

该参数为将来开发预留，必须预设为 0。

doLimitUpper2

该参数为将来开发预留，必须预设为 0。

doLimitUpper3

该参数为将来开发预留，必须预设为 0。

doLimitLower1

该参数为将来开发预留，必须预设为 0。

doLimitLower2

该参数为将来开发预留，必须预设为 0。

doLimitLower3

该参数为将来开发预留，必须预设为 0。

doDisplayRangeFrom

指示从哪个值开始进行受数据控制的显示。

doDisplayRangeTo

指示受数据控制的显示到哪个值结束。

doLimitRangeFrom

该参数为将来开发预留，必须预设为 0。

doLimitRangeTo

该参数为将来开发预留，必须预设为 0。

crColor

Windows 特定的 32 位值 crColor 确定趋势的显示颜色。

crColorTimeOverlapped

该参数预留以供将来扩展。

crColorTimeJump

该参数预留以供将来扩展。

crColorLimitUpper1

该参数预留以供将来扩展。

crColorLimitUpper2

该参数预留以供将来扩展。

crColorLimitUpper3

该参数预留以供将来扩展。

crColorLimitLower1

该参数预留以供将来扩展。

crColorLimitLower2

该参数预留以供将来扩展。

crColorLimitLower3

该参数预留以供将来扩展。

szSelectSQL

该参数为将来开发预留，必须预设为 NULL。

szText;

x 轴标签

szFunction

该参数为将来开发预留，必须预设为 NULL。

szDLL

该参数为将来开发预留，必须预设为 NULL。

注释

TLG_CURVESCALEX 由 TLG_TPLITEM_CURVE (页 2536) 结构使用。

所需文件

pdertdef.h

参见

TLG_TPLITEM_CURVE (页 2536)

3.8.2.5 TLG_CURVESCALEY

声明

```
typedef struct {
    DWORD         dwDataTyp;
    DWORD         dwRangeTyp;
    DWORD         dwAxisLocation;
    DWORD         dwCurveForm;
    BOOL          fAutoRange;
    BOOL          fGridLinesBig;
    BOOL          fGridLinesFine;
    BOOL          fGridLinesBigVisible;
    BOOL          fGridLinesFineVisible;
    BOOL          fPercent;
    BOOL          fLimitRange;
    BOOL          fSubstitute;
    SYSTEMTIME    stFrom;
    SYSTEMTIME    stTo;
    double        doFrom;
    double        doTo;
    double        doGridBig;
    double        doGridFine;
    double        doShowDigitsF;
    double        doShowDigitsB;
    double        doLimitUpper1;
    double        doLimitUpper2;
     double        doLimitUpper3;
    double        doLimitLower1;
    double        doLimitLower2;
    double        doLimitLower3;
    double        doDisplayRangeFrom;
    double        doDisplayRangeTo;
    double        doLimitRangeFrom;
    double        doLimitRangeTo;
    COLORREF      crColor;
    COLORREF      crColorTimeOverlapped;
    COLORREF      crColorTimeJump;
    COLORREF      crColorLimitUpper1;
    COLORREF      crColorLimitUpper2;
    COLORREF      crColorLimitUpper3;
    COLORREF      crColorLimitLower1;
    COLORREF      crColorLimitLower2;
    COLORREF      crColorLimitLower3;
    TCHAR         szSelectSQL[ TLG_MAX_SQL_SELECT ];
    TCHAR         szText[ TLG_MAX_STD_TEXT_NAME ];
    }
    TLG_CURVESCALEY;
```

成员

dwDataTyp

数据类型（时间/用户/块数据）在此没有任何意义。

dwRangeTyp

该参数为将来开发预留，必须预设为 0。

dwAxisLocation

该参数为将来开发预留，必须预设为 0。

dwCurveForm

趋势的外观由 dwCurveForm 定义

TLG_CURVEFORM_STEP	作为步骤趋势的趋势视图
TLG_CURVEFORM_POINTS	作为单点的趋势视图
TLG_CURVEFORM_DIRECT	线性连接趋势上的各个点

fAutoRange

fAutoRange = TRUE 自动调整显示范围。

fGridLinesBig

fgridLinesBig = TRUE 激活粗网格线。

fGridLinesFine

fgridLinesFine = TRUE 激活细网格线。

fGridLinesBigVisible

如果 fgridLinesBigVisible = TRUE，则将显示粗网格线。

fGridLinesFineVisible

如果 fgridLinesFineVisible = TRUE，则将显示细网格线。

fPercent

该参数为将来开发预留，必须预设为 0。

fLimitRange

该参数为将来开发预留，必须预设为 0。

fSubstitute

该参数为将来开发预留，必须预设为 0。

stFrom

该参数为将来开发预留。

stTo

该参数为将来开发预留。

doFrom

y 轴上的显示范围下限。

doTo

y 轴上的显示范围上限。

doGridBig

粗网格线之间的间距。

doGridFine

细网格线之间的间距。

doShowDigitsF

值显示的位数（小数点前）

doShowDigitsB

确定小数位数

doLimitUpper1

该参数为将来开发预留，必须预设为 0。

doLimitUpper2

该参数为将来开发预留，必须预设为 0。

doLimitUpper3

该参数为将来开发预留，必须预设为 0。

doLimitLower1

该参数为将来开发预留，必须预设为 0。

doLimitLower2

该参数为将来开发预留，必须预设为 0。

doLimitLower3

该参数为将来开发预留，必须预设为 0。

doDisplayRangeFrom

确定显示范围的起始值。

doDisplayRangeTo

确定显示范围的终止值。

doLimitRangeFrom

该参数为将来开发预留，必须预设为 0。

doLimitRangeTo

该参数为将来开发预留，必须预设为 0。

crColor

Windows 特定的 32 位值 crColor 确定趋势的显示颜色。

crColorTimeOverlapped

该参数为将来开发预留。

crColorTimeJump

该参数为将来开发预留。

crColorLimitUpper1

该参数为将来开发预留。

crColorLimitUpper2

该参数为将来开发预留。

crColorLimitUpper3

该参数为将来开发预留。

crColorLimitLower1

该参数为将来开发预留。

crColorLimitLower2

该参数为将来开发预留。

crColorLimitLower3

该参数为将来开发预留。

szSelectSQL[TLG_MAX_SQL_SELECT];

该参数为以后升级预留，必须设置为默认值 ZERO。

szText[TLG_MAX_STD_TEXT_NAME];

y 轴标签

注释

TLG_CURVESCALEY 由 TLG_TPLITEM_CURVE (页 2536) 结构使用。

所需文件

pdertdef.h

参见

TLG_TPLITEM_CURVE (页 2536)

3.8.2.6 TLG_GETARCHIVDATA

声明

```
typedef struct {
    LPTSTR        lpszArchivName;
    LPTSTR        lpszVarName;
    SYSTEMTIME    stTime;
    double        doValue;
    DWORD         dwFlags;
    }
    TLG_GETARCHIVDATA;
```

成员

lpszArchivName

指向从中读出数据的日志名称的指针。

lpszVarName

指向数值读取操作源变量名称的指针。

stTime

日志值的 x 值

doValue

日志值的 y 值

dwFlags

该参数为将来开发预留，必须预设为 0。

所需文件

pdertdef.h

API 函数

TLG_GETARCHIVDATA_CALLBACK (页 2585)	读出日志的数据（回调）

参见

TLG_GETARCHIVDATA_CALLBACK (页 2585)

3.8.2.7 TLG_IO_BACKUP_SELECT

声明

```
typedef struct {
    SYSTEMTIME    sysFrom;
    SYSTEMTIME    sysTo;
    LPTSTR        lpszSqlString;
    }
    TLG_IO_BACKUP_SELECT;
```

成员

sysFrom

要选择的首个数据记录的系统时间

sysTo

要选择的最后一个数据记录的系统时间

lpszSqlString

该参数为将来开发预留，必须预设为 NULL。

所需文件

pdertdef.h

API 函数

TLGGetBackupSize (页 2624)	确定导出的数据记录的大小
TLGExport (页 2622)	导出日志数据

参见

TLGGetBackupSize (页 2624)

TLGExport (页 2622)

3.8.2.8 TLG_PROT_CURVE_INFOS

声明

```
typedef struct {
    TCHAR         szArchivName[ 128 + 1 ];
    TCHAR         szVariableName[ 128 + 1 ];
    TCHAR         szTextX[ 128 + 1 ];
    TCHAR         szTextY[ 128 + 1 ];
    SYSTEMTIME    stFrom;
    SYSTEMTIME    stTo;
    double        doFrom;
    double        doTo;
    DWORD         dwFlags;
    DWORD         dwCurveForm;
    }
    TLG_PROT_CURVE_INFOS;
```

成员

szArchivName

要记录的数据的源日志名称。

szVariableName

要记录的值所属的日志变量的名称。

szTextX

x 轴标签

szTextY

y 轴标签

stFrom

记录开始的时间。只有在 dwFlags 已将 TLG_PROTFLG_TIME_RANGE 置位时才会对此参数进行评估。

stTo

记录终止的时间。只有在 dwFlags 已将 TLG_PROTFLG_TIME_RANGE 置位时才会评估此参数。

如果需要一个用于传递 SYSTEMTIME 参数的当前定时器，则必须使用 GetLocalTime 函数，而非 GetSystemTime。通常，这两个函数的时间有很大差异。

doFrom

记录的起始时间。只有在 dwFlags 已将 TLG_PROTFLG_DATA_RANGE 置位时才会评估此参数。

doTo

记录终止的时间。只有在 dwFlags 已将 TLG_PROTFLG_DATA_RANGE 置位时才会评估此参数。

dwFlags

dwFlags 可确定趋势所基于的数据类型：

TLG_PROTFLG_TIME_RANGE	数据来自于受时间控制的日志（例如：过程值日志）。 stFrom 和 stTo 中指定的值有效。
TLG_PROTFLG_DATA_RANGE	数据来自于受时间控制的日志（例如：用户日志）。 doFrom 和 doTo 中指定的值有效。

dwCurveForm

趋势的外观由 dwCurveForm 定义

TLG_CURVEFORM_STEP	作为步骤趋势的趋势视图
TLG_CURVEFORM_POINTS	作为单点的趋势视图
TLG_CURVEFORM_DIRECT	线性连接趋势上的各个点

注释

TLG_PROT_CURVE_INFOS 由结构 TLG_TABLESCALE (页 2528) 和 TLG_TPLITEM_INFO (页 2537) 使用。

所需文件

pdertdef.h

API 函数

TLGDrawCurvesInDC (页 2604)	显示趋势

参见

TLGDrawCurvesInDC (页 2604)

TLG_TABLESCALE (页 2528)

TLG_TPLITEM_INFO (页 2537)

3.8.2.9 TLG_SCAL_STR

声明

```
typedef struct {
    double    doScalVarUpper;
    double    doScalVarLower;
    double    doScalArcUpper;
    double    doScalArcLower;
    TCHAR     szStructName[PDE_DB_SCALE_MAX_LENGHT];
    }
    TLG_SCAL_STR;
```

成员

doScalVarLower

变量值下限

doScalVarUpper

变量值上限

doScalArcLower

该参数为将来开发预留，必须预设为 0。

doScalArcUpper

该参数为将来开发预留，必须预设为 0。

szStructName

变量限值的缩放结构名称

注释

TLG_SCAL_STR 结构在 TLG_VAR_STR (页 2539) 结构中使用，供将来升级使用。

所需文件

pde_def.h

参见

TLG_VAR_STR (页 2539)

3.8.2.10 TLG_TABLE_INFO

声明

```
typedef struct {
    DWORD    dwArchivTyp;
    TCHAR    szArchivName[ _MAX_PATH + 1 ];
    DWORD    dwSaveTyp;
    }
    TLG_TABLE_INFO;
```

成员

dwArchivTyp

dwArchivTyp 标识日志类型：

TLG_ARCTYP_USER	用户日志
TLG_ARCTYP_PROCESS	过程值日志
TLG_ARCTYP_COMPRESS	压缩日志

szArchivName

日志名称的提供形式取决于使用 TLG_TABLE_INFO 的函数。

TLGEnumArchivs 和 TLGEnumArchivsSel 对日志的枚举将日志名称作为表格名称提供，即对用户日志采用"UA#ARCHIV#Archivname"，对其它日志采用"PDE#HD#Archivname#Variablenname"，因为此处存在各个变量的表格条目。因此每个变量都会被枚举。

TLGEnumArchivsEx 的枚举提供“纯”日志名称。

dwSaveTyp

dwSaveTyp 标识日志的类型：

TLG_ARCTYP_CIRCULAR	循环日志
TLG_ARCTYP_FOLLOW	顺序日志

所需文件

pdertdef.h

API 函数

TLG_ENUMTABLES (页 2580)	列出日志（回调）

参见

TLG_ENUMTABLES (页 2580)

3.8.2.11 TLG_TABLESCALE

声明

```
typedef struct {
    BOOL          fActualize;
    BOOL          fVisible;
    BOOL          fModify;
    BOOL          fCommon;
    DWORD         dwDataTyp;
    SYSTEMTIME    stFrom;
    SYSTEMTIME    stTo;
    double        doFrom;
    double        doTo;
    double        doShowDigits;
    double        doLimitUpper1;
    double        doLimitUpper2;
    double        doLimitUpper3;
    double        doLimitLower1;
    double        doLimitLower2;
    double        doLimitLower3;
    OLORREF      crColor;
    COLORREF      crColorTimeOverlapped;
    COLORREF      crColorTimeJump;
    COLORREF      crColorLimitUpper1;
    COLORREF      crColorLimitUpper2;
    COLORREF      crColorLimitUpper3;
    COLORREF      crColorLimitLower1;
    COLORREF      crColorLimitLower2;
    COLORREF      crColorLimitLower3;
    TCHAR         szSelectSQL[ TLG_MAX_SQL_SELECT ];
    }
    TLG_TABLESCALE;
```

成员

fActualize

TRUE	趋势视图是动态的。
FALSE	趋势视图是静态的。

fVisible

TRUE	列可见。
FALSE	列不可见。

fModify

TRUE	列中的条目可编辑。
FALSE	列中的条目不可编辑。

fCommon

TRUE	表格中的所有列都有公共的时间列。
FALSE	表格中的列没有公共的时间列。

dwDataTyp

该参数为将来开发预留，必须预设为 0。

stFrom

显示范围的开始时间。

stTo

显示范围的结束时间。

如果需要一个用于传递 SYSTEMTIME 参数的当前定时器，则必须使用 GetLocalTime 函数，而非 GetSystemTime。通常，这两个函数的时间有很大差异。

doFrom

该参数为将来开发预留，必须预设为 0。

doTo

该参数为将来开发预留，必须预设为 0。

doShowDigits

要输出的小数位数。

doLimitUpper1

该参数为将来开发预留，必须预设为 0。

doLimitUpper2

该参数为将来开发预留，必须预设为 0。

dpoLimitUper3

该参数为将来开发预留，必须预设为 0。

doLimitLower1

该参数为将来开发预留，必须预设为 0。

doLimitLower2

该参数为将来开发预留，必须预设为 0。

doLimitLower3

该参数为将来开发预留，必须预设为 0。

crColor

Windows 特定的 32 位值 crColor 定义表格列中使用的颜色。

crColorTimeOverlapped

该参数为将来开发预留。

crColorTimeJump

该参数为将来开发预留。

crColorLimitUpper1

该参数为将来开发预留。

crColorLimitUpper2

该参数为将来开发预留。

crColorLimitUpper3

该参数为将来开发预留。

crColorLimitLower1

该参数为将来开发预留。

crColorLimitLower2

该参数为将来开发预留。

crColorLimitLower3

该参数为将来开发预留。

szSelectSQL

该参数为以后升级预留，必须设置为默认值 ZERO。

注释

TLG_TABLESCALE 由 TLG_TEMPLATEITEM_INFO (页 2531) 结构使用。

所需文件

pdertdef.h

参见

TLG_PROT_CURVE_INFOS (页 2523)

TLG_TPLITEM_INFO (页 2537)

TLG_TEMPLATEITEM_INFO (页 2531)

TLG_TPLITEM_TABLE (页 2538)

3.8.2.12 TLG_TEMPLATEITEM_INFO

声明

```
typedef struct {
    TCHAR               szTemplateItemName[
                        TLG_MAX_TEMPLATEITEM_NAME+1 ];
    TCHAR               szTemplateName[
                        TLG_MAX_TEMPLATE_NAME + 1 ];
    TCHAR               szArchivName[ 128 + 1 ];
    TCHAR               szVariableName[ 128 + 1 ];
    TCHAR               szDMVariableName[ 128 + 1 ];
    DWORD               dwReadAccessLevel;
    DWORD               dwWriteAccessLevel;
    DWORD               dwArchivTyp;
    TCHAR               szTimeNameRange[ 128 + 1 ];
    DWORD               dwTemplateItemTyp;
    DWORD               dwTemplateTyp;
    BOOL                fVisible;
    TLG_TPLITEM_INFO    tplInfo;
    }
    TLG_TEMPLATEITEM_INFO;
```

成员

szTemplateItemName

趋势或列的名称

szTemplateName

趋势或表格窗口模板的名称

szArchivName

与曲线或列相链接的日志变量的记录日志名称。

szVariableName

与趋势或列相链接的日志变量的名称

szDMVariableName

数据管理器变量的名称

dwReadAccessLevel

在趋势或列的枚举中，dwReadAccesLevel 包含读访问的用户授权级别。

dwWriteAccessLevel

在趋势或列的枚举中，dwWriteAccesLevel 包含写访问的用户授权级别。

dwArchivTyp

dwArchivTyp 标识日志类型：

TLG_ARCTYP_USER	用户日志
TLG_ARCTYP_PROCESS	过程值日志
TLG_ARCTYP_COMPRESS	压缩日志

szTimeNameRange

用于根据开始时间确定时间范围的时间对象的名称。

dwTemplateItemTyp

dwTemplateItemTyp 必须对应于 dwTemplateTyp 中指定的值：

TLG_TEMPLATEITEM_ALL	趋势和表格模板
TLG_TEMPLATEITEM_CURVE	趋势模板
TLG_TEMPLATEITEM_TABLE	表格模板

dwTemplateTyp

dwTemplateTyp 标识窗口模板的类型

TLG_TEMPLATE_CURVE	趋势窗口模板
TLG_TEMPLATE_TABLE	表格窗口模板

fVisible

TRUE	列或趋势可见。
FALSE	列或趋势不可见。

tplInfo

具有趋势或列模板属性的 TLG_TPLITEM_INFO (页 2537) 类型的结构。

所需文件

pdertdef.h

API 函数

TLG_TABLESCALE (页 2528)	表格列（属性）（结构）

参见

TLG_TPLITEM_INFO (页 2537)

TLG_TABLESCALE (页 2528)

TLGInsertTemplateItem (页 2605)

3.8.2.13 TLG_TIME_STR

声明

```
typedef struct {
    DWORD        dwBasis;
    DWORD        dwFactor;
    TCHAR        szTimeName[
                PDE_DB_TIMENAME_MAX_LENGHT + 1 ];
    }
    TLG_TIME_STR;
```

成员

dwBasis

时基。周期时间通过时间系数与时基相乘得出。

TLG_TBASE_500MS	500 毫秒
TLG_TBASE_SEC	1 秒
TLG_TBASE_MIN	1 分钟
TLG_TBASE_HOUR	1 小时
TLG_TBASE_DAY	1 天

dwFactor

时间系数。周期时间通过时间系数与时基相乘得出。

szTimeName

时间对象的名称

注释

时间将被视为采集和记录周期，可自由分配。一个时间对象可同时用作采集周期和记录周期。

所需文件

pde_def.h

API 函数

TLGReadTime (页 2618)	确定时间对象参数

参见

TLGReadTime (页 2618)

3.8.2.14 TLG_TIMEDATA

声明

```
typedef struct {
    TCHAR    szTimeName[ _MAX_PATH + 1 ];
    DWORD    dwTimeBase;
    DWORD    dwTimeValue;
    }
    TLG_TIMEDATA;
```

成员

szTimeName

时间对象的名称

dwTimeBase

时基。周期时间通过时间系数与时基相乘得出。

TLG_TBASE_500MS	500 毫秒
TLG_TBASE_SEC	1 秒
TLG_TBASE_MIN	1 分钟
TLG_TBASE_HOUR	1 小时
TLG_TBASE_DAY	1 天

dwTimeValue

时间系数。周期时间通过时间系数与时基相乘得出。

所需文件

pdertdef.h

API 函数

TLG_ENUMTIMES_CALLBACK (页 2616)	列出时间对象（回调）

参见

TLG_ENUMTIMES_CALLBACK (页 2616)

3.8.2.15 TLG_TPLITEM_CURVE

声明

```
typedef struct {
    TLG_CURVESCALEX    csx;
    TLG_CURVESCALEY    csy;
    }
    TLG_TPLITEM_CURVE;
```

成员

csx

TLG_CURVESCALEX (页 2512) 结构包含标定趋势 X 轴的数据。

csy

TLG_CURVESCALEY (页 2517) 结构包含标定趋势 Y 轴的数据。

说明

TLG_TPLITEM_CURVE 由 TLG_TPLITEM_INFO (页 2537) 结构使用。

所需文件

pdertdef.h

参见

TLG_TPLITEM_INFO (页 2537)

TLG_CURVESCALEX (页 2512)

TLG_CURVESCALEY (页 2517)

3.8.2.16 TLG_TPLITEM_INFO

声明

```
typedef union {
    TLG_TPLITEM_CURVE    tplCurve;
    TLG_TPLITEM_TABLE    tplTable;
    }
    TLG_TPLITEM_INFO;
```

成员

tplCurve

趋势的属性从 TLG_TPLITEM_CURVE (页 2536) 结构中读取。

tplTable

列的属性从 TLG_TPLITEM_TABLE (页 2538) 结构中读取。

注释

TLG_TPLITEM_INFO 由 TLG_TEMPLATEITEM_INFO (页 2531) 结构使用。

由于变量既可在趋势中也可在表中显示，结构 TLG_TPLITEM_CURVE 和 TLG_TPLITEM_TABLE 都可分配值。

所需文件

pdertdef.h

参见

TLG_PROT_CURVE_INFOS (页 2523)

TLG_TABLESCALE (页 2528)

TLG_TEMPLATEITEM_INFO (页 2531)

TLG_TPLITEM_CURVE (页 2536)

TLG_TPLITEM_TABLE (页 2538)

3.8.2.17 TLG_TPLITEM_TABLE

声明

```
typedef struct {
    TLG_TABLESCALE    ts;
    }
    TLG_TPLITEM_TABLE;
```

成员

ts

具有表格列属性的 TLG_TABLESCALE (页 2528)。

说明

TLG_TPLITEM_TABLE 由 TLG_TPLITEM_INFO (页 2537) 结构使用。

由于组织原因，TLG_TABLESCALE 不可直接在 TLG_TPLITEM_INFO 结构中使用。

所需文件

pdertdef.h

参见

TLG_TPLITEM_INFO (页 2537)

TLG_TABLESCALE (页 2528)

3.8.2.18 TLG_VAR_STR

声明

```
typedef struct {
    TCHAR             szVarName [PDE_DB_VAR_NAME_MAX_LENGHT + 1];
    TCHAR             szProcName [PDE_DB_VAR_NAME_MAX_LENGHT + 1];
    DWORD             dwVarType;
    DWORD             dwArchivStyle;
    TCHAR             szWriteBackTo [PDE_DB_VAR_WRITEBACK_MAX_LENGHT + 1];
    DWORD             dwSupply;
    BOOL              fLocked;
    TCHAR             szComment [PDE_DB_VAR_COMMENT_MAX_LENGHT + 1];
    TCHAR             szRecordCycle [PDE_DB_VAR_CYCLENAME_MAX_LENGHT + 1];
    TCHAR             szArchivCycle [PDE_DB_VAR_CYCLENAME_MAX_LENGHT + 1];
    DWORD             dwMultiple;
    DWORD             dwValueFlow;
    DWORD             dwValueFollow;
    DWORD             dwSaveByFault;
    DWORD             dwArchivTrigger;
    TCHAR             szTextHighSignal [PDE_DB_VAR_SIGNALTEX_MAX_LENGHT + 1];
    TCHAR             szTextLowSignal [PDE_DB_VAR_SIGNALTEX_MAX_LENGHT + 1];
    DWORD             dwValProcess;
    TCHAR             szFuncValProcess [PDE_DB_VAR_FUNCNAME_MAX_LENGHT + 1];
    TCHAR             szDLLValProcess [PDE_DB_VAR_DLLNAME_MAX_LENGHT + 1];
    TCHAR             szFuncStartEvent [PDE_DB_VAR_FUNCNAME_MAX_LENGHT + 1];
    TCHAR             szDLLStartEvent [PDE_DB_VAR_DLLNAME_MAX_LENGHT + 1];
    TCHAR             szFuncStopEvent [PDE_DB_VAR_FUNCNAME_MAX_LENGHT + 1];
    TCHAR             szDLLStopEvent [PDE_DB_VAR_DLLNAME_MAX_LENGHT + 1];
    TCHAR             szFuncDynamic [PDE_DB_VAR_FUNCNAME_MAX_LENGHT + 1];
    TCHAR             szDLLDynamic [PDE_DB_VAR_DLLNAME_MAX_LENGHT + 1];
    TCHAR             szUnitDirect [PDE_DB_VAR_UNITDIR_MAX_LENGHT + 1];
    TCHAR             szUnitStruct [PDE_DB_VAR_UNITSTR_MAX_LENGHT + 1];
    DWORD             dwRecItems;
    TCHAR             szSourceArchiv [PDE_DB_ARC_NAME_MAX_LENGHT + 1];
    TCHAR             szSourceVarName [PDE_DB_VAR_NAME_MAX_LENGHT + 1];
    TCHAR             szRawdataName [PDE_DB_VAR_NAME_MAX_LENGHT + 1];
    TCHAR             szRawConvDLLName [TLG_DB_DLLNAME_MAX_LENGHT + 1];
    DWORD             dwRawDataIndex;
    DWORD             dwRawDataFormat;
    TLG_SCAL_STR      ScaleStruct;
    TLG_RECORD_STR    RecordStruct;
    } TLG_VAR_STR;
```

注释

TLG_VAR_STR 结构的参数分配取决于记录类型（非周期、周期性选择或周期性连续）和变量类型（二进制或模拟量）。

成员

szVarName

日志变量或变量组名称的全局参数

日志变量名称中不得使用字符 : ? " ' \ * % 和空格，否则会输出错误 TLG_API_NAME_WRONG_CHAR。

szProcName

过程变量名称的全局参数

dwVarType

标识日志变量类型的全局参数。

TLG_VAR_TYP_ANALOG	0x00010001	模拟量变量： dwArchivTrigger 在内部设置为 0L（无关）
TLG_VAR_TYP_BIN	0x00010002	二进制变量： dwValProcess 在内部设置为 0L（无关）
TLG_VAR_TYP_COMPRESS	0x00010004	压缩日志变量： dwSuppy 在内部设置为 TLG_SUPPLY_BY_SYSTEM， dwSaveByFault 在内部设置为 TLG_SAVE_LAST_VALUE
TLG_VAR_TYP_PROCESS	0x00010008	过程控制变量
	无效值	错误消息：TLG_API_VARTYPE_MISMATCH

dwArchivStyle

标识记录类型的全局参数。

TLG_STY_ACYCLIC	0x00800001	非周期记录
TLG_STY_CYCLIC_CON	0x00800002	周期性连续记录
TLG_STY_ CYCLIC_SEL	0x00800004	周期性选择记录
TLG_STY_ON_CHANGE	0x00800008	更改检测
	无效值	错误消息 TLG_API_ERR_ARCHIVSTYLE

szWriteBackTo

日志标签名称将写回的全局参数。

dwSupply

标识变量提供类型的全局参数。

TLG_SUPPLY_BY_HAND	0L - FALSE	手动变量输入
TLG_SUPPLY_BY_SYSTEM	1L - TRUE	由系统提供（过程耦合或内部变量）
	无效值	错误消息 TLG_API_INVALID_PARAM

如果 dwVarType = TLG_VAR_TYP_COMPRESS，则 TLG_SUPPLY_BY_SYSTEM 始终在内部设置。

fLocked

标识系统启动时记录处理方式的全局参数。

TRUE	已禁用
FALSE	启用

szComment

日志变量注释的全局参数

szRecordCycle

指定采集周期的时间对象名称（例如：“1 分钟”）。
参数与模拟量和二进制变量的周期性选择、周期性连续采集相关。

szArchivCycle

指定记录周期的时间对象名称（例如："1 分钟")。记录周期通过 dwMultiple 与 szArchivCycle 相乘得出。
此参数与模拟量和二进制变量的周期性选择、周期性连续记录相关。

dwMultiple

标识相乘的系数。记录周期通过 dwMultiple 与 szArchivCycle 相乘得出。
此参数与模拟量和二进制变量的周期性选择、周期性连续记录相关。

dwValueFlow

标识流值数量
此参数与模拟量和二进制变量的周期性选择、周期性连续记录相关。

dwValueFollow

标识返回值数量。
此参数与模拟量和二进制变量的周期性选择、周期性连续记录相关。

dwSaveByFault

标识发生故障时存储器的行为。
此参数与所有记录类型的模拟量和二进制变量相关。

TLG_SAVE_LAST_VALUE	1L	保存上一个有效值
TLG_SAVE_SUBST_VALUE	2L	保存替换值
	无效值	错误消息 TLG_API_INVALID_PARAM

如果 dwVarType = TLG_VAR_TYP_COMPRESS，则 TLG_SAVE_LAST_VALUE 始终在内部设置。

dwArchivTrigger

标识记录的触发行为
此参数与二进制变量的周期性选择、周期性连续记录相关。

TLG_TRIG_CHANGE	1L	对各个信号更改的记录
TLG_TRIG_CHANGE_01	2L	对 0 -> 1 信号更改的记录
TLG_TRIG_CHANGE_10	3L	对 1 -> 0 信号更改的记录
TLG_TRIG_ALLWAYS	4L	即使未发生信号更改，每个记录周期中也都会记录此值。
	无效值	错误消息 TLG_API_INVALID_PARAM

如果 dwVarType = TLG_VAR_TYP_ANALOG，则 0L 始终在内部设置（无关）。

szTextHighSignal

信号状态 1 的文本。
此参数与二进制变量的周期性选择、周期性连续记录相关。

szTextLowSignal

信号状态 0 的文本。
此参数与二进制变量的周期性选择、周期性连续记录相关。

dwValProcess

标识在记录周期内获取的值中所要保存的日志值的处理方式。
此参数与模拟量变量的周期性选择、周期性连续记录相关。

TLG_VAL_ACTUAL	1L	实际值
TLG_VAL_AVERAGE	2L	平均值
TLG_VAL_SUM	3L	总和
TLG_VAL_MIN	4L	最小值
TLG_VAL_MAX	5L	最大值
TLG_VAL_DLL	6L	函数或 DLL
	无效值	错误消息 TLG_API_INVALID_PARAM

如果 dwVarType = TLG_VAR_TYP_BIN ，则 0L 始终在内部设置（无关）。

szFuncValProcess

对于要保存的日志值，指定其处理方式的操作或 DLL 函数的名称。
此参数与模拟量和二进制变量的周期性选择、周期性连续记录相关，并且只有在 dwValProcess = TLG_VAL_DLL 时才相关。

szDLLValProcess

DLL 的名称
此参数与模拟量和二进制变量的周期性选择、周期性连续记录相关，并且只有在 dwValProcess = TLG_VAL_DLL 时才相关。

szFuncStartEvent

指定起始事件的操作或 DLL 函数的名称。
此参数与模拟量和二进制变量的周期性选择、周期性连续记录相关。

szDLLStartEvent

DLL 的名称。
此参数与模拟量和二进制变量的周期性选择、周期性连续记录相关。

szFuncStopEvent

指定终止事件的操作或 DLL 函数的名称。
此参数与模拟量和二进制变量的周期性选择记录相关。

szDLLStopEvent

DLL 的名称
此参数与模拟量和二进制变量的周期性选择记录相关。

szFuncDynamic

指定动态切换检测和/或记录周期的操作或 DLL 函数的名称。
此参数与模拟量和二进制变量的周期性选择、周期性连续记录相关。

szDLLDynamic

DLL 的名称
此参数与模拟量和二进制变量的周期性选择、周期性连续记录相关。

szUnitDirect

标识日志变量的单位。
此参数与所有记录类型的模拟量变量相关。

szUnitStruct

包含日志变量单位的结构的名称。如果单位在 szUnitDirect 中直接组态，则为 NULL。
此参数与所有记录类型的模拟量变量相关。

dwRecItems

该参数为将来开发预留，必须预设为 0。

szSourceArchiv

源归档的名称。
此参数与周期性连续记录相关。

szSourceVarName

源日志中日志变量的名称
此参数与周期性连续记录相关。

szRawdataName

原始数据变量的名称
此参数与过程控制变量相关。

szRawConvDLLName

标准化 DLL 的名称
此参数与过程控制变量相关。

dwRawDataIndex

原始数据变量的编号对应于与 S7PMC 相连的 PLC 的日志 ID。
此参数与过程控制变量相关。

dwRawDataFormat

原始数据变量的格式
此参数与过程控制变量相关。

ScaleStruct

具有变量限制的 TLG_SCAL_STR (页 2525) 结构。

RecordStruct

该参数为将来开发预留。

所需文件

pde_def.h

API 函数

TLGReadVariable (页 2568)	确定变量参数

参见

TLGReadVariable (页 2568)

TLG_SCAL_STR (页 2525)

3.8.2.19 TLG_VARIABLE_INFO

声明

```
typedef struct {
    DWORD    dwVariableTyp;
    TCHAR    szVariableName[ _MAX_PATH + 1 ];
    }
    TLG_VARIABLE_INFO;
```

成员

dwVariableTyp

dwVariableTyp 用于标识变量的类型。可能值包括：

TLG_VAR_TYP_ANALOG	模拟变量
TLG_VAR_TYP_BIN	二进制变量

TLG_VAR_TYP_COMPRESS	压缩日志变量
TLG_VAR_TYP_PROCESS	过程控制变量

szVariableName

变量的名称

所需文件

pdertdef.h

pde_typ.h

API 函数

TLG_ENUMVARIABLES (页 2567)	列出变量（回调）

参见

TLG_ENUMVARIABLES (页 2567)

3.8.3 常规函数

3.8.3.1 TLGCSConnect

描述

该函数用于建立与 WinCC 项目数据库的连接。

声明

```
BOOL TLGCSConnect (
    HWND           hwndParent,
    LPCMN_ERROR    lpError );
```

参数

hwndParent

父窗口的窗口句柄

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

连接已建立

FALSE

错误

所需文件

pdecscli.h

pdecscli.lib

pdecscli.dll

相关函数

TLGCSConnectEx (页 2548)	建立与项目数据库的连接。
TLGCSDisConnect (页 2549)	建立与项目数据库的连接。

示例

枚举所有采集和记录时间 (页 2630) "TL01.cpp"

读取时间对象的参数 (页 2644) "TL01.cpp"

枚举日志 (页 2633) "TL01.cpp"

读取日志 (页 2640) "TL01.cpp"

参见

TLGCSConnectEx (页 2548)

枚举日志 (页 2633)

TLGOpenProject (页 2557)

读取时间对象的参数 (页 2644)

枚举所有采集和记录时间 (页 2630)

读取日志 (页 2640)

TLGCSDisConnect (页 2549)

3.8.3.2 TLGCSConnectEx

说明

该函数用于建立与 WinCC 项目数据库的连接。

声明

```
BOOL TLGCSConnectEx (
    HWND            hwndParent,
    DWORD           dwMode,
    LPCMN_ERROR     lpError );
```

参数

hwndParent

该参数为以后升级预留，必须设置为默认值 ZERO。

dwMode

该参数为将来开发预留。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

连接已建立

FALSE

错误

所需文件

pdecscli.h

pdecscli.lib

pdecscli.dll

相关函数

TLGCSConnect (页 2546)	建立与项目数据库的连接。

参见

TLGCSConnect (页 2546)

TLGOpenProject (页 2557)

3.8.3.3 TLGCSDisConnect

描述

该函数用于终止与 WinCC 项目数据库建立的连接。需要调用，以便可以再次彻底卸载 DLL。

说明

不能在应用程序（EXE、DLL、OCX 等）的析构函数中调用，因为这样做可能由于特定基于 Microsoft 的机制而导致调用被冻结，进而导致程序崩溃。

声明

```
BOOL TLGCSDisConnect (
    LPCMN_ERROR    lpError );
```

参数

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

连接被终止。

FALSE

错误

所需文件

pdecscli.h

pdecscli.lib

pdecscli.dll

相关函数

TLGCSConnect (页 2546)	建立与项目数据库的连接。

示例

枚举所有采集和记录时间 (页 2630) "TL01.cpp"

读取时间对象的参数 (页 2644) "TL01.cpp"

枚举日志 (页 2633) "TL01.cpp"

读取日志 (页 2640) "TL01.cpp"

参见

TLGCSConnect (页 2546)

枚举所有采集和记录时间 (页 2630)

枚举日志 (页 2633)

读取日志 (页 2640)

读取时间对象的参数 (页 2644)

3.8.3.4 TLGChangeLanguage

说明

可使用此函数切换当前使用的数据语言。

声明

```
BOOL TLGChangeLanguage (
    DWORD          dwLanguage,
    PCMN_ERROR     lpError );
```

参数

dwLanguage

要在将来使用的语言的代码

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

数据语言已切换

FALSE

错误

错误消息

TLG_API_INVALID_PARAM	提供的参数无效/错误
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接
TLG_API_ERR_SUPPLY	发生错误

所需文件

pdertcli.h

pde_glob.h

pdertcli.lib

pdertcli.dll

3.8.3.5 TLGConnect

说明

该函数用于初始化日志系统并建立与变量记录运行系统的连接。

声明

```
BOOL TLGConnect (
    HWND          hwndParent,
    PCMN_ERROR    lpError );
```

参数

hwndParent

该参数为将来开发预留，必须预设为 NULL。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

连接已建立

FALSE

错误

注释

调用时将生成不可见的通信窗口，只能使用 TLGDisconnect 函数删除该窗口。当同时使用 Windows 函数时，这一点可能很重要。

错误消息

TLG_API_NO_CONNECTION	未与激活的项目建立任何连接
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败

所需文件

pdertcli.h

pde_glob.h

pdertcli.lib

pdertcli.dll

相关函数

TLGDisconnect (页 2554)	终止连接

示例

枚举日志 (页 2638) "TL02.cpp"

参见

枚举日志 (页 2638)

TLGDisconnect (页 2554)

编辑曲线模板 - 示例 1 (页 2627)

3.8.3.6 TLGDisconnect

说明

使用此函数可断开与变量记录运行系统的现有连接。

声明

```
BOOL TLGDisconnect (
    PCMN_ERROR    lpError );
```

参数

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

连接被终止

FALSE

错误

注释

说明

不能在应用程序（EXE、DLL、OCX...）的析构函数中调用，因为这可能会由于 Microsoft 特定的机制而导致调用堵塞，进而导致程序堵塞。

错误消息

TLG_API_NO_CONNECTION	未与激活的项目建立任何连接
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败

所需文件

pdertcli.h

pde_glob.h

pdertcli.lib

pdertcli.dll

相关函数

TLGConnect (页 2552)	建立连接

示例

枚举日志 (页 2638) "TL02.cpp"

参见

枚举日志 (页 2638)

TLGConnect (页 2552)

编辑曲线模板 - 示例 1 (页 2627)

枚举所有采集和记录时间 (页 2630)

3.8.4 项目管理函数

3.8.4.1 TLGCloseProject

说明

关闭当前项目。属于此项目的所有数据对象都将关闭。

该函数不再有意义，将始终返回 TRUE。

声明

```
BOOL TLGCloseProject(
    HANDLE         hProject,
    LPCMN_ERROR    lpoes );
```

参数

hProject

使用 TLGOpenProject 打开的项目的句柄。

lpoes

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

项目已关闭

FALSE

错误

所需文件

pdecscli.h

pdecscli.lib

pdecscli.dll

示例

读取时间对象的参数 (页 2644) "TL01.cpp"

枚举日志 (页 2633) "TL01.cpp"

读取日志 (页 2640) "TL01.cpp"

参见

读取时间对象的参数 (页 2644)

枚举日志 (页 2633)

读取日志 (页 2640)

3.8.4.2 TLGOpenProject

说明

建立与 WinCC 项目数据库的连接。

声明

```
BOOL TLGOpenProject (
    HANDLE*         lphProject,
    LPTSTR          lpszProjectName,
    HWND            hwndParent,
    LPCMN_ERROR     lpoes );
```

参数

lphProject

用于保存句柄的内存区域的地址。

lpszProjectName

要打开的项目的名称。

可通过以下 API 函数之一确定要在此处指定的项目路径：

- DMEnumOpenedProjects
- DMGetRuntimeProject
- TLGEnumProject

如果指定的项目路径与当前打开的项目的路径不同，则返回 TLG_API_PROJECT_NAME_NOT_EXIST 错误。如果未打开任何项目，则返回 TLG_API_NO_PROJECT_EXIST 错误。

hwndParent

父窗口的句柄

lpoes

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

项目已打开

FALSE

错误

注释

如果没有先通过 TLGCSConnect 与 WinCC 项目数据库建立连接，则返回连接错误 TLG_API_NO_CONNECTION。

错误消息

TLG_API_INVALID_PARAM	提供的参数无效/错误
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接
TLG_API_ERR_SUPPLY	发生错误
TLG_API_NO_PROJECT_EXIST	不存在项目
TLG_API_PROJECT_NAME_NOT_EXIST	未找到项目名称

所需文件

pdecscli.h

pdecscli.lib

pdecscli.dll

相关函数

TLGCSConnect (页 2546)	建立与项目数据库的连接。
TLGCSConnectEx (页 2548)	建立与项目数据库的连接。
TLGEnumProject (页 2559)	列出项目

示例

读取时间对象的参数 (页 2644) "TL01.cpp"

枚举日志 (页 2633) "TL01.cpp"

读取日志 (页 2640) "TL01.cpp"

枚举日志的变量 (页 2636) "TL02.cpp"

参见

TLGEnumProject (页 2559)

TLGCSConnect (页 2546)

TLGCSConnectEx (页 2548)

读取时间对象的参数 (页 2644)

枚举日志 (页 2633)

读取日志 (页 2640)

枚举日志的变量 (页 2636)

3.8.4.3 TLGEnumProject

说明

此函数用于确定打开的项目的名称。

声明

```
BOOL TLGEnumProject (
    TLG_ENUM_PROJECT_NAME_CALLBACK    lpCallbackFunc,
    PVOID                             lpUser,
    LPCMN_ERROR                       lpoes );
```

参数

lpCallbackFunc

指向为每个打开的项目调用的回调函数的指针。

lpUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpoes

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

项目已列出

FALSE

错误

所需文件

pdecscli.h

pdecscli.lib

pdecscli.dll

相关函数

TLG_ENUM_PROJECT_NAME_CALLBACK (页 2561)	列出项目（回调）

参见

TLG_ENUM_PROJECT_NAME_CALLBACK (页 2561)

TLGOpenProject (页 2557)

3.8.4.4 TLG_ENUM_PROJECT_NAME_CALLBACK

说明

为能够评估系统列出的项目，必须提供 TLG_ENUM_PROJECT_NAME_CALLBACK 类型的回调函数。

声明

```
BOOL ( * TLG_ENUM_PROJECT_NAME_CALLBACK) (
    LPTSTR    lpszName,
    PVOID     lpUser );
```

参数

lpszName

lpszName 指针指向第一个项目的名称。

lpUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

返回值视执行情况而定。

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

pdecscli.h

相关函数

TLGEnumProject (页 2559)	列出项目

参见

TLGEnumProject (页 2559)

3.8.5 用于处理变量的函数

3.8.5.1 TLGEnumVariables

说明

此函数用于确定日志的所有变量名称。

声明

```
BOOL TLGEnumVariables (
    HANDLE                              hProject,
    LPTSTR                              lpszArchivName,
    TLG_ENUM_VARIABLE_NAME_CALLBACK     lpCallbackFunc,
    PVOID                               lpUser,
    LPCMN_ERROR                         lpoes );
```

参数

hProject

日志所在项目的句柄。

lpszArchivName

指向日志名称的指针

lpCallbackFunc

指向为日志中每个变量条目调用的回调函数的指针。

lpUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpoes

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

日志变量已列出

FALSE

错误

所需文件

pdecscli.h

pdecscli.lib

pdecscli.dll

相关函数

TLG_ENUM_VARIABLE_NAME_CALLBACK (页 2564)	列出变量（回调）

参见

TLG_ENUM_VARIABLE_NAME_CALLBACK (页 2564)

TLGReadVariable (页 2568)

3.8.5.2 TLG_ENUM_VARIABLE_NAME_CALLBACK

说明

为能够评估系统列出的变量，必须提供 TLG_ENUM_VARIABLE_NAME_CALLBACK 类型的回调函数。

声明

```
BOOL ( * TLG_ENUM_VARIABLE_NAME_CALLBACK) (
    LPTSTR    lpszName,
    PVOID     lpUser );
```

参数

lpszName

lpszName 指针指向第一个变量的名称。

lpUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

pdecscli.h

相关函数

TLGEnumVariables (页 2562)	列出变量

参见

TLGEnumVariables (页 2562)

3.8.5.3 TLGEnumVariablesEx

说明

此函数用于确定日志的所有变量名称。

声明

```
BOOL TLGEnumVariablesEx (
    LPCTSTR              lpszArchiveName,
    TLG_ENUMVARIABLES    lpfnCallback,
    LPVOID               lpUser,
    PCMN_ERROR           lpError );
```

参数

lpszArchiveName

指向要列出其变量的日志的名称的指针。

lpfnCallback

指向为日志的每个变量调用的回调函数的指针。

lpUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

变量已列出

FALSE

错误

错误消息

TLG_API_ERR_SUPPLY	发生错误
TLG_API_INVALID_PARAM	提供的参数无效/错误
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接

所需文件

pdecscli.h

pdecscli.lib

pdecscli.dll

pdertcli.h

pdertcli.lib

pdertcli.dll

相关函数

TLG_ENUMVARIABLES (页 2567)	列出变量（回调）

示例

枚举日志的变量 (页 2636) "TL02.cpp"

参见

TLG_ENUMVARIABLES (页 2567)

枚举日志的变量 (页 2636)

3.8.5.4 TLG_ENUMVARIABLES

说明

为能够评估系统列出的变量的数据，必须提供 TLG_ENUMVARIABLES 类型的回调函数。

声明

```
BOOL ( * TLG_ENUMVARIABLES) (
    PTLG_VARIABLE_INFO    lpvi,
    LPVOID                lpUser );
```

参数

lpvi

具有变量数据的 TLG_VARIABLE_INFO (页 2545) 类型结构的地址。

lpUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

相关函数

TLGEnumVariablesEx (页 2565)	列出变量

示例

枚举日志的变量 (页 2636) "TL02.cpp"

参见

TLGEnumVariablesEx (页 2565)

TLG_VARIABLE_INFO (页 2545)

枚举日志的变量 (页 2636)

TLGEnumArchivsEx (页 2575)

3.8.5.5 TLGReadVariable

说明

读取变量的参数

声明

```
BOOL TLGReadVariable (
    HANDLE          hProject,
    LPTSTR          lpszArchivName,
    LPTSTR          lpszVariableName,
    PTLG_VAR_STR    lpVariable,
    LPCMN_ERROR     lpoes );
```

参数

hProject

要处理的变量所在项目的句柄。

lpszArchivName

指向日志名称的指针

lpszVariableName

指向日志变量的名称的指针

lpVariable

要从中读出变量数据的 TLG_VAR_STR (页 2539) 结构的地址。

lpoes

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

变量已读取

FALSE

错误

所需文件

pdecscli.h

pdecscli.lib

pdecscli.dll

相关函数

TLGEnumVariables (页 2562)	列出变量
TLGReadVariable6	确定变量参数

示例

读取日志 (页 2640) "TL01.cpp"

参见

TLG_VAR_STR (页 2539)

TLGEnumVariables (页 2562)

读取日志 (页 2640)

3.8.6 用于处理日志的函数

3.8.6.1 TLGEnumArchives

说明

此函数用于确定项目 hproject 中所有日志的名称。所有类型的日志均可使用此函数处理。

声明

```
BOOL TLGEnumArchives (
    HANDLE                      hProject,
    TLG_ENUM_ARCHIV_CALLBACK    lpCallbackFunc,
    PVOID                       lpUser,
    LPCMN_ERROR                 lpoes );
```

参数

hProject

日志所在项目的句柄。

lpCallbackFunc

指向为每个日志调用的回调函数的指针。

lpUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpoes

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

日志已列出

FALSE

错误

所需文件

pdecscli.h

pdecscli.lib

pdecscli.dll

相关函数

TLGReadArchiv (页 2601)	读取日志参数
TLG_ENUM_ARCHIV_CALLBACK (页 2572)	列出日志（回调）

示例

枚举日志 (页 2633) "TL01.cpp"

参见

TLG_ENUM_ARCHIV_CALLBACK (页 2572)

枚举日志 (页 2633)

TLGReadArchiv (页 2601)

3.8.6.2 TLG_ENUM_ARCHIV_CALLBACK

说明

为能够评估系统列出的日志，必须提供 TLG_ENUM_ARCHIV_CALLBACK 类型的回调函数。

声明

```
BOOL ( * TLG_ENUM_ARCHIV_CALLBACK) (
    LPTSTR    lpszName,
    PVOID     lpUser );
```

参数

lpszName

lpszName 指针指向日志的名称。

lpUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

注释

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

pdecscli.h

相关函数

TLGEnumArchives (页 2570)	列出日志

示例

枚举日志 (页 2633) "TL01.cpp"

参见

TLGEnumArchives (页 2570)

枚举日志 (页 2633)

3.8.6.3 TLGEnumArchivs

说明

枚举现有日志。与 TLGEnumArchivsSel 不同，要列出的日志只能通过日志类型进行限制。

通过回调函数在 TLG_TABLE_INFO 类型的结构中提供日志的信息。日志名称在这里传递为表格名称，即“UA#ARCHIV#ArchivName”或“PDE#HD#Archivname#Variablenname”的形式。

声明

```
BOOL TLGEnumArchivs (
    DWORD             dwArchivTyp,
    TLG_ENUMTABLES    lpfnCallback,
    LPVOID            lpUser,
    PCMN_ERROR        lpError );
```

参数

dwArchivTyp

dwArchivTyp 标识日志类型：

TLG_ARCTYP_USER	用户日志
TLG_ARCTYP_PROCESS	过程值日志
TLG_ARCTYP_COMPRESS	压缩日志
NULL	所有日志均已枚举。

lpfnCallback

指向为每个日志调用的回调函数的指针。

lpUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

日志已列出

FALSE

错误

错误消息

TLG_API_ERR_SUPPLY	发生错误
TLG_API_INVALID_PARAM	提供的参数无效/错误
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接
TLG_API_ERR_SUPPLY	发生错误

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

相关函数

TLGEnumArchivsEx (页 2575)	列出日志
TLGEnumArchivsSel (页 2577)	列出日志
TLG_ENUMTABLES (页 2580)	列出日志（回调）

示例

枚举日志 (页 2638) "TL02.cpp"

参见

TLGEnumArchivsEx (页 2575)

TLGEnumArchivsSel (页 2577)

TLG_ENUMTABLES (页 2580)

枚举日志 (页 2638)

3.8.6.4 TLGEnumArchivsEx

说明

枚举现有日志。与 TLGEnumArchivsSel 不同，要列出的日志时只能通过日志类型（用户、过程值、压缩日志）进行限制。

通过回调函数在 TLG_TABLE_INFO 类型的结构中提供日志的信息。与 TLGEnumArchivs 不同，这里不会传递日志的表格名称，而是传递“纯”日志名称。

声明

```
BOOL TLGEnumArchivsEx (
     DWORD             dwArchivTyp,
     TLG_ENUMTABLES    lpfnCallback,
     LPVOID            lpUser,
     PCMN_ERROR        lpError );
```

参数

dwArchivTyp

dwArchivTyp 标识日志类型：

TLG_ARCTYP_U_USER	用户日志
TLG_ARCTYP_PROCESS	过程值日志
TLG_ARCTYP_COMPRESS	压缩日志
NULL	所有日志均已枚举。

lpfnCallback

指向为每个日志调用的回调函数的指针。

lpUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

日志已列出

FALSE

错误

错误消息

TLG_API_ERR_SUPPLY	发生错误
TLG_API_INVALID_PARAM	提供的参数无效/错误
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

相关函数

TLG_ENUMTABLES (页 2580)	列出日志（回调）
TLGEnumArchivs (页 2573)	列出日志
TLGEnumArchivsSel (页 2577)	列出日志

参见

TLGEnumArchivsSel (页 2577)

TLGEnumArchivs (页 2573)

TLG_ENUMTABLES (页 2580)

3.8.6.5 TLGEnumArchivsSel

说明

此函数用于枚举现有的日志类型。与 TLGEnumArchivs 不同，除了日志类型外，要列出的日志还可通过记录类型（循环日志、顺序日志）进行限制。

通过回调函数在 TLG_TABLE_INFO 类型的结构中提供日志的信息。日志名称在这里传递为表格名称，即“UA#ARCHIV#ArchivName”或“PDE#HD#Archivname#Variablenname”的形式。日志的所有变量均已枚举。例如，在 TLGBackup 中需要此形式。

声明

```
BOOL TLGEnumArchivsSel (
        DWORD             dwArchivTyp,
        DWORD             dwSaveTyp,
        TLG_ENUMTABLES    lpfnCallback,
        LPVOID            lpUser,
        PCMN_ERROR        lpError );
```

参数

dwArchivTyp

dwArchivTyp 标识日志类型：

TLG_ARCTYP_USER	用户日志
TLG_ARCTYP_PROCESS	过程值日志
TLG_ARCTYP_COMPRESS	压缩日志
NULL	所有日志均已枚举。

dwSaveTyp

dwSaveTyp 标识日志的类型：

TLG_ARCTYP_ALL	循环和顺序日志
TLG_ARCTYP_CIRCULAR	循环日志
TLG_ARCTYP_FOLLOW	顺序日志

lpfnCallback

指向为每个日志调用的回调函数的指针。

lpUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

日志已列出。

FALSE

错误

错误消息

TLG_API_ERR_SUPPLY	发生错误
TLG_API_INVALID_PARAM	提供的参数无效或错误
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

相关函数

TLG_ENUMTABLES (页 2580)	列出日志（回调）
TLGEnumArchivs (页 2573)	列出日志

参见

TLGEnumArchivsEx (页 2575)

TLGEnumArchivs (页 2573)

TLG_ENUMTABLES (页 2580)

TLG_TABLE_INFO (页 2526)

3.8.6.6 TLG_ENUMTABLES

说明

为能够评估系统列出的日志，必须提供 TLG_ENUMTABLES 类型的回调函数。

声明

```
BOOL TLG_ENUMTABLES (
    LPTSTR                lpTableName,
    PTLG_TABLE_INFO        lpti,
    PVOID lpUser );
```

参数

lpTableName

指向日志名称的指针

lpti

有关数据库表的信息包含在 TLG_TABLE_INFO (页 2526) 结构中。

lpUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

注释

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

相关函数

TLGEnumArchivs (页 2573)	列出日志
TLGEnumArchivsSel (页 2577)	列出日志

示例

枚举归档“TL02.cpp”

参见

TLGEnumArchivs (页 2573)

TLGEnumArchivsSel (页 2577)

TLG_TABLE_INFO (页 2526)

TLGEnumArchivsEx (页 2575)

3.8.6.7 TLGFreeMemory

说明

此函数用于释放 TLGGetArchivDataEx 分配的内存区域。

声明

```
BOOL TLGFreeMemory (
     LPVOID lpMemory );
```

参数

lpMemory

获得在调用 TLGGetArchivDataEx 函数时 ppTLGData 参数中释放的内存的指针。

说明

在调用 TLGGetArchivDataEx() 之前必须将该指针初始化为 NULL，在调用 TLGFreeMemory() 之前必须确定该指针不等于 NULL。

返回值

TRUE

内存已释放

FALSE

错误

说明

在调用 TLGGetArchivDataEx() 之前必须将该指针初始化为 NULL，在调用 TLGFreeMemory() 之前必须确定该指针不等于 NULL。

所需文件

pdertcli.h

pdertcli.lib

pdertcli.dll

相关函数

TLGGetArchivDataEx (页 2587)	读出日志的数据

参见

TLGGetArchivDataEx (页 2587)

3.8.6.8 TLGGetArchivData

说明

此函数用于读出日志中介于两个时间之间的数据。与 TLGGetArchivDataEx 不同，日志变量的值由回调函数提供。

一次调用最多可读取 10,000 条数据记录。如果起始时间和终止时间之间的数据记录多于此值，则只能获取最前面的 10,000 条。

通过将最后一条数据记录的时间加一毫秒并将此值作为新的 TLGGetArchivData 调用的起始时间来获取后面的数据记录。

声明

```
BOOL TLGGetArchivData (
    LPTSTR                          lpszArchivName,
    LPTSTR                          lpszVarName,
    SYSTEMTIME                      stStart,
    SYSTEMTIME                      stStop,
    TLG_GETARCHIVDATA_CALLBACK      lpfnCallback,
    PVOID                           lpUser,
    DWORD                           dwFlags,
    PCMN_ERROR                      lpError );
```

参数

lpszArchivName

指向要从中读取数据的日志的名称的指针。

lpszVarName

指向要读取其值的日志变量的名称的指针。

stStart

读取数据的起始时间。

stStop

读取数据的结束时间

lpfnCallback

指向要为每个读出的测量点调用的回调函数的指针。

lpUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

dwFlags

该参数为将来开发预留，必须预设为 0。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

日志数据已确定

FALSE

错误

错误消息

TLG_API_ERR_SUPPLY	发生错误
TLG_API_INVALID_PARAM	提供的参数无效/错误
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接
TLG_API_NODEFAULTSERVER	未组态标准服务器
TLG_API_NOSERVER	未组态标准服务器，且无可用本地服务器

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

相关函数

TLG_GETARCHIVDATA_CALLBACK (页 2585)	读出日志的数据（回调）
TLGGetArchivDataEx (页 2587)	读出日志的数据

参见

TLG_GETARCHIVDATA_CALLBACK (页 2585)

TLGGetArchivDataEx (页 2587)

3.8.6.9 TLG_GETARCHIVDATA_CALLBACK

说明

为了评估系统列出的日志数据，必须提供 TLG_GETARCHIVDATA_CALLBACK 类型的回调函数。

声明

```
BOOL ( * TLG_GETARCHIVDATA_CALLBACK) (
    PTLG_GETARCHIVDATA    lpGAD,
    PVOID                 lpUser );
```

参数

lpGAD

系统会为每个日志保留 TLG_GETARCHIVDATA 结构长度的临时存储。lpGAD 指针指向第一个元素的开头。

lpUser

指向应用程序特定数据的指针。该指针会在回调函数中重新提供。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

说明

如果可能，仅移动数据。以下回调中的函数调用类型可导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如 GetMessage
- 相同 DLL 中的运行时 API 函数
- 调用其他枚举的枚举

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

相关函数

TLGGetArchivData (页 2583)	读出日志的数据
TLG_GETARCHIVDATA (页 2521)	过程值日志的日志数据（结构），压缩日志的日志数据（结构）

参见

TLGGetArchivData (页 2583)

TLG_GETARCHIVDATA (页 2521)

3.8.6.10 TLGGetArchivDataEx

说明

读出日志中介于两个时间之间的数据。与 TLGGetArchivData 不同，日志变量的值保存在内存中。

说明

在 ppTlgData 中分配的内存区域必须通过 TLGFreeMemory 释放。

声明

```
BOOL TLGGetArchivDataEx (
    LPCTSTR                lpszArchivName,
    LPCTSTR                lpszVarName,
    SYSTEMTIME*            pstStart,
    SYSTEMTIME*            pstStop,
    PTLG_ARCHIVDATARAW*    ppTlgData,
    DWORD*                 pdwNumberOfData,
    DWORD*                 pdwFlags,
    PCMN_ERROR             lpError );
```

参数

lpszArchivName

指向要从中读出数据的日志名称的指针。

lpszVarName

指向要读取其值的日志变量的名称的指针。

stStart

读取数据的起始时间。

stStop

读取数据的结束时间

如果需要一个用于传递 SYSTEMTIME 参数的当前定时器，则使用 GetLocalTime 函数，而非 GetSystemTime。通常，这两个函数的时间有很大差异。

ppTlgData

要存储日志变量地址的指针的地址。数据字段由 TLGGetArchivDataEx 分配并获得 TLG_ARCHIVDATARAW (页 2506) 类型的结构。

说明

分配的内存区域必须再次通过 TLGFreeMemory 释放。

在调用 TLGGetArchivDataEx() 之前必须将该指针初始化为 NULL，在调用 TLGFreeMemory() 之前必须确定该指针不等于 NULL。

pdwNumberOfData

指向数据记录数的指针

- 调用前：要读取的最大数据记录数
- 调用后：已读取的含有（TLG_ARCHIVDATARAW 类型的结构的）日志变量值的数据记录数。

一次调用最多可读取 10,000 条数据记录。如果起始时间和终止时间之间的数据记录多于此值，则只能获取最前面的 10000 条。通过将最后一条数据记录的时间加一毫秒并将此值作为新的 TLGGetArchivDataEx 调用的起始时间来获取后面的数据记录。

dwFlags

该参数为将来开发预留，必须预设为 0。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

日志数据已确定

FALSE

错误

说明

通过 TLGGetArchivDataEx 在 ppTlgData 中分配的内存区域必须再次通过 TLGFreeMemory 释放。

在调用 TLGGetArchivDataEx() 之前必须将该指针初始化为 NULL，在调用 TLGFreeMemory() 之前必须确定该指针不等于 NULL。

错误消息

TLG_API_ERR_SUPPLY	发生错误
TLG_API_INVALID_PARAM	提供的参数无效/错误
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接
TLG_API_NODEFAULTSERVER	未组态标准服务器
TLG_API_NOSERVER	未组态标准服务器，且无可用本地服务器

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

相关函数

TLGGetArchivData (页 2583)	读出日志的数据
TLGFreeMemory (页 2582)	释放内存

参见

TLGFreeMemory (页 2582)

TLGGetArchivData (页 2583)

TLG_ARCHIVDATARAW (页 2506)

3.8.6.11 TLGGetClosestTime

说明

此函数用于确定给定时间的邻近记录时间。

声明

```
BOOL TLGGetClosestTime (
    LPCTSTR         lpszArchivName,
    LPCTSTR         lpszVarName,
    SYSTEMTIME*     pstTime,
    BOOL            bPrevious,
    PCMN_ERROR      lpError );
```

参数

lpszArchivName

指向日志名称的指针

lpszVarName

指向变量名称的指针

pstTime

pstTime 用作输入和输出参数。调用 TLGGetClosestTime 时，pstTime 包含要确定邻近记录时间时的系统时间。函数调用成功后，pstTime 包含最近的记录时间。

bPrevious

bPrevious 标识在传递的系统时间是确定上一个还是下一个记录时间。

TRUE	确定上一个记录时间。
FALSE	确定下一个记录时间。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

pstTime 包含邻近记录时间。

FALSE

确定临近记录时间出错或无法确定临近记录时间。

注释

如果将 bPrevious 指定为 TRUE，并且 pstTime 包含一个现有值，则可能返回该值对应的时间，而不是上一个时间。如果 bPrevious 指定为 FALSE，将始终提供下一个时间。

错误消息

TLG_API_INVALID_PARAM	提供的参数无效/错误
TLG_API_ERR_SUPPLY	发生错误
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接
TLG_API_VARIABLE_NAME_NOT_EXIST	未找到任何具有此名称的变量
TLG_API_TIME_NAME_NOT_EXIST	未找到任何具有此名称的时间对象
TLG_API_NO_TIME_EXIST	不存在时间对象

所需文件

pdertcli.h

pde_glob.h

pdertcli.lib

pdertcli.dll

相关函数

TLGGetClosestTimeEx (页 2592)	确定记录时间

参见

TLGGetClosestTimeEx (页 2592)

3.8.6.12 TLGGetClosestTimeEx

说明

此函数用于确定给定时间的邻近记录时间。

声明

```
BOOL TLGGetClosestTimeEx (
    LPCTSTR         lpszArchivName,
    LPCTSTR         lpszVarName,
    SYSTEMTIME*     pstTime,
    BOOL            bPrevious,
    BOOL            bIgnoreInvalid,
    PCMN_ERROR      lpError );
```

参数

lpszArchivName

指向日志名称的指针

lpszVarName

指向变量名称的指针

pstTime

pstTime 用作输入和输出参数。调用 TLGGetClosestTime 时，pstTime 包含要确定邻近记录时间时的系统时间。函数调用成功后，pstTime 包含最近的记录时间。

bPrevious

bPrevious 标识在传递的系统时间是确定上一个还是下一个记录时间。

TRUE	确定上一个记录时间。
FALSE	确定下一个记录时间。

bIgnoreInvalid

指示是否要在时间确定时忽略具有连接故障标记的值。

TRUE	忽略并跳过连接故障值。
FALSE	所有值都将用于时间确定，以便存在与 TLGGetClosestTime() 中相同的函数。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

pstTime 包含邻近记录时间。

FALSE

确定临近记录时间出错或无法确定临近记录时间。

注释

如果将 bPrevious 指定为 TRUE，并且 pstTime 包含一个现有值，则可能返回该值对应的时间，而不是上一个时间。如果 bPrevious 指定为 FALSE，将始终提供下一个时间。

错误消息

TLG_API_INVALID_PARAM	提供的参数无效/错误
TLG_API_ERR_SUPPLY	发生错误
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接
TLG_API_VARIABLE_NAME_NOT_EXIST	未找到任何具有此名称的变量

TLG_API_TIME_NAME_NOT_EXIST	未找到任何具有此名称的时间对象
TLG_API_NO_TIME_EXIST	不存在时间对象

所需文件

pdertcli.h

pde_glob.h

pdertcli.lib

pdertcli.dll

相关函数

TLGGetClosestTime (页 2590)	确定记录时间

参见

TLGGetClosestTime (页 2590)

3.8.6.13 TLGInsertArchivData

说明

使用此函数可在硬盘上的日志中插入任何数据。

如果针对夏令时期间创建的值执行 TlgInsertArchivData，则必须允许时间偏差。调用这些值时，设置 PDE_RT_DAYLIGHT 标记。

如果没有此标记，读取 TLGGetArchivData 或 TLGGetClosestTime 等调用将返回默认时间。

说明

由于未检查要插入的数据的正确性，因此存在破坏日志的危险。

声明

```
BOOL TLGInsertArchivData (
    LPCTSTR               lpszArchivName,
    LPCTSTR               lpszVarName,
    PTLG_ARCHIVDATARAW    pTlgData,
    DWORD                 dwNumberOfData,
    DWORD                 dwFlags,
    PCMN_ERROR            lpError );
```

参数

lpszArchivName

指向日志名称的指针

lpszVarName

指向变量名称的指针

pTlgData

指向第一个 TLG_ARCHIVDATARAW 类型结构的指针，其包含要插入到日志的变量值。

dwNumberOfData

要插入的变量值数（位于 TLG_ARCHIVDATARAW 结构中）。

dwFlags

TLG_API_FLG_FAST_INSERT	如果设置该标记，则可在最优模式下更快地写入数据。然而，使用该性能优化的先决条件是写入数据记录的时间戳按时间顺序排序，并且始终比日志中已有的数据记录新。 如果在现有数据记录之间插入旧数据记录，则无法使用该标记。如果插入的值比现有值早或未按时间顺序排序，则旧值将被拒且不显示错误消息。
TLG_API_FLG_REPLACE_INSERT	如果该标志置位，则会覆盖日志中的数据。 “TLG_API_FLG_REPLACE_INSERT”标志不能与“TLG_API_FLG_FAST_INSERT”标志结合使用。
TLG_API_FLG_DETAILED_ERRORCODES	如果该标志置位，则会输出详细错误代码。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

变量值已插入日志中

FALSE

错误

错误代码

如果在 lpszArchivName 下指定位于主存储器中的日志，将输出错误消息 TLG_API_ERR_SUPPLY。

使用该函数只能寻址硬盘上的日志。如果某个值发生写入错误，则会在 CMN_ERROR 中返回更具体的错误描述。由于 CMN_ERR.szErrorText 长度受到限制，所以详细错误数限制为最多 57 个。

如果在 lpszArchivName 下指定位于主存储器中的日志，将输出错误消息 TLG_API_ERR_SUPPLY。

dwError1	TLG_API_ERR_SUPPLY
dwError2	0
dwError3	源代码中的行编号
dwError4	错误数（最多 57 个）
dwError5	要写入的数据数 (dwNumberOfData)
szErrorText	TLG-API：发生错误，行 xxx；位置：0=0xXX，1=0xXX，2=0xXX，... (，56=0xXX)

TLG_API_FLG_DETAILED_ERRORCODES 标志在 dwFlags 中置位时，输出以下详细错误代码。

错误代码	描述	值（带标志）	值（不带标志）
OK	无错误	0x00	0x00
错误	一般错误状态	0x01	0x40
记录已存在	数据记录已存在，不会被覆盖。	0x02	0x80

错误代码	描述	值（带标志）	值（不带标志）
不允许	禁止手动输入值。	0x04	0x800
超出限值	值超出限值	0x08	0x1000
无服务器	服务器连接出错	0x10	0x10
未找到变量	未找到记录变量	0x20	0x40
标志无效	禁止结合使用标志 TLG_API_FLG_REPLACE_INSERT 和 TLG_API_FLG_FAST_INSERT	0x40	0x40
未来时间戳	不允许带有未来时间戳的值	0x80	0x40
日志基于 RAM	不允许对主存储器中的日志进行写操作	0x100	0x40

错误消息

TLG_API_INVALID_PARAM	提供的参数无效/错误
TLG_API_ERR_SUPPLY	发生错误
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接
TLG_API_NODEFAULTSERVER	未组态标准服务器
TLG_API_NOSERVER	未组态标准服务器，且无可用本地服务器

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

3.8.6.14 TLGLockArchiv

说明

锁定或释放整个日志。如果日志锁定，则不记录新数据。

声明

```
BOOL TLGLockArchiv (
    HWND          hwnd,
    LPTSTR        lpszArchivName,
    BOOL          fLocked,
    PCMN_ERROR    lpError );
```

参数

hwnd

PDE 运行系统窗口的窗口句柄

lpszArchivName

指向日志名称的指针

fLocked

TRUE：	锁定日志
FALSE：	释放日志

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已锁定/释放日志

FALSE

错误

错误消息

TLG_API_INVALID_PARAM	提供的参数无效/错误
TLG_API_ERR_SUPPLY	发生错误
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接
TLG_API_NO_ARCHIV_EXIST	无可用日志
TLG_API_NODEFAULTSERVER	未组态标准服务器
TLG_API_NOSERVER	未组态标准服务器，且无可用本地服务器

所需文件

pdertcli.h

pde_glob.h

pdertcli.lib

pdertcli.dll

3.8.6.15 TLGLockVariable

说明

锁定或释放变量。系统不再更新和记录锁定的变量。

声明

```
BOOL TLGLockVariable (
    HWND          hwnd,
    LPTSTR        lpszArchivName,
    LPTSTR        lpszVarName,
    BOOL          fLocked,
    PCMN_ERROR    lpError );
```

参数

hwnd

PDE 运行系统窗口的窗口句柄

lpszArchivName

指向日志名称的指针

要选择某服务器，可在日志名称前加前缀“ServerPrefix::”。

lpszVarName

指向变量名称的指针

fLocked

TRUE：	锁定变量
FALSE：	释放变量

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已释放/锁定变量

FALSE

错误

错误消息

TLG_API_INVALID_PARAM	提供的参数无效/错误
TLG_API_ERR_SUPPLY	发生错误
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接
TLG_API_NO_ARCHIV_EXIST	无可用日志
TLG_API_NODEFAULTSERVER	未组态标准服务器
TLG_API_NOSERVER	未组态标准服务器，且无可用本地服务器

所需文件

pdertcli.h

pde_glob.h

pdertcli.lib

pdertcli.dll

3.8.6.16 TLGReadArchiv

说明

读取 hProject 项目中的现有日志的参数。所有类型的日志均可使用此函数处理。

声明

```
BOOL TLGReadArchiv (
    HANDLE              hProject,
    LPTSTR              lpszArchivName,
    PTLG_ARCHIV_STR     lpArchiv,
    LPCMN_ERROR         lpoes );
```

参数

hProject

要处理的日志所在项目的句柄。

lpszArchivName

要处理的日志的名称。

lpArchiv

要用于保存日志参数的 TLG_ARCHIV_STR (页 2503) 结构的地址。

lpoes

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已读取日志内容

FALSE

错误

所需文件

pdecscli.h

pdecscli.lib

pdecscli.dll

相关函数

TLGEnumArchives (页 2570)	列出日志
TLGReadArchiv6	读取日志参数

示例

读取日志 (页 2640)“TL01.cpp”

参见

TLGEnumArchives (页 2570)

读取日志 (页 2640)

TLG_ARCHIV_STR (页 2503)

3.8.7 用于趋势和表格视图的函数

3.8.7.1 TLGCloseWindow

说明

使用该函数可关闭应用程序窗口。

声明

```
BOOL TLGCloseWindow (
    HWND          hwnd,
    PCMN_ERROR    lpError );
```

参数

hwnd

要关闭的应用程序的父窗口的句柄。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

应用程序窗口已关闭

FALSE

错误

错误消息

TLG_API_ERR_SUPPLY	发生错误
TLG_API_INVALID_PARAM	提供的参数无效/错误

所需文件

pdertcli.h

pde_glob.h

pdertcli.lib

pdertcli.dll

3.8.7.2 TLGDrawCurvesInDC

说明

使用此函数可在应用程序窗口中显示趋势。

声明

```
BOOL TLGDrawCurvesInDC (
    HDC                     hDC,
    PRECT                   lprect,
    PTLG_PROT_CURVE_INFOS   lpci,
    DWORD                   dwNumberOfCurves,
    LPCMN_ERROR             lpError );
```

参数

hDC

要在其中输出趋势的父窗口的句柄

lprect

指向具有窗口大小数据的 RECT 类型 Windows 特定结构的指针。

lpci

指向具有趋势输出方式相关信息的第一个 TLG_PROT_CURVE_INFOS (页 2523) 类型结构的指针。

dwNumberOfCurves

lpci 中传送的结构数（要显示的趋势数）。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已显示趋势

FALSE

错误

错误消息

TLG_API_INVALID_PARAM	提供的参数无效/错误
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接
TLG_API_NOT_SUPPORTED	不支持该函数

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

参见

TLG_PROT_CURVE_INFOS (页 2523)

3.8.7.3 TLGInsertTemplateItem

说明

该函数可为现有窗口模板（趋势模板或表格模板）插入新条目。

声明

```
BOOL TLGInsertTemplateItem (
    LPTSTR                  lpszTemplateName,
    PTLG_TEMPLATEITEM_INFO  lpptii,
    LPCMN_ERROR             lpError );
```

参数

lpszTemplateName

指向趋势或表格模板要添加到的窗口模板名称的指针。

lpptii

指向具有要添加趋势或表格模板数据的 TLG_TEMPLATEITEM_INFO (页 2531) 类型结构的指针。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已扩展窗口模板

FALSE

错误

错误消息

TLG_API_ERR_SUPPLY	发生错误
TLG_API_INVALID_PARAM	提供的参数无效/错误
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接
TLG_API_NOT_SUPPORTED	不支持该函数

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

示例

编辑曲线模板 - 示例 1 (页 2627) "TL02.cpp"

参见

TLG_TEMPLATEITEM_INFO (页 2531)

编辑曲线模板 - 示例 1 (页 2627)

3.8.7.4 TLGPressToolbarButton

说明

使用此函数可触发与工具栏按钮相链接的函数。

声明

```
BOOL TLGPressToolbarButton (
    LPTSTR        lpszTemplate,
    DWORD         dwButtonID,
    PCMN_ERROR    lpError );
```

参数

lpszTemplate

模板名称

dwButtonID

要触发的按钮的代码编号：

趋势和表格的常量	
TLG_BASIC_BTN_HELP	打开在线帮助
TLG_BASIC_BTN_DLG	打开参数分配对话框
TLG_BASIC_BTN_FIRST	显示日志中的第一个数据记录。
TLG_BASIC_BTN_PREV	在日志中回滚。
TLG_BASIC_BTN_NEXT	在日志中前滚。
TLG_BASIC_BTN_LAST	显示日志中的最后一个数据记录。

趋势和表格的常量	
TLG_BASIC_BTN_STARTSTOP	激活或禁用趋势或表格的更新视图。
TLG_BASIC_BTN_PREV_ITEM	前景或表格第一列中的上一趋势。
TLG_BASIC_BTN_NEXT_ITEM	前景或表格第一列中的下一趋势。
TLG_BASIC_BTN_ARC_VAR_SELECT	打开日志变量选择对话框。
TLG_BASIC_BTN_ITEM_SELECT	打开趋势或表格列的选择对话框。
TLG_BASIC_BTN_TIME_SELECT	打开显示时间范围的选择对话框。

表格的常量	
TLG_TABLE_BTN_EDIT	激活或禁用编辑功能。

趋势的常量	
TLG_CURVE_BTN_ZOOMIN	激活用于放大某部分的缩放功能。
TLG_CURVE_BTN_ZOOMOUT	禁用缩放功能。
TLG_CURVE_BTN_1_TO_1	切换到常规趋势视图。
TLG_CURVE_BTN_LINEAL	激活或禁用标尺。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

错误消息

TLG_API_INVALID_PARAM	提供的参数无效/错误
IDS_API_ERROR_WINDOW_NOT_FOUND	无法找到按钮（窗口）
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接
TLG_API_NO_TYP_CHANGE_ALLOWED	不允许更改类型

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

3.8.7.5 TLGSetRulerWindowVisible

说明

使用此函数可显示或隐藏读取行的数据窗口。

声明

```
BOOL TLGSetRulerWindowVisible (
    LPTSTR          lpszTemplateName,
    BOOL            bShowRulerWindow,
    LPCMN_ERROR     lpError );
```

参数

lpszTemplateName

指向窗口模板名称的指针

bShowRulerWindow

TRUE：	显示具有数据窗口的读取行。
FALSE：	显示无数据窗口的读取行。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已重新输入读取行数据窗口的可见性参数。

FALSE

错误

注释

在显示具有该对象的相应图像前，该函数在 f(x) 趋势视图和表格视图中无效。

错误消息

TLG_API_ERR_SUPPLY	发生错误
IDS_API_ERROR_WINDOW_NOT_FOUND	无法找到按钮（窗口）
TLG_API_INVALID_PARAM	提供的参数无效/错误

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

3.8.7.6 TLGShowWindow

说明

该函数影响窗口的显示形式。

声明

```
BOOL TLGShowWindow (
    HWND          hwnd,
    DWORD         dwFlags,
    PCMN_ERROR    lpError );
```

参数

hwnd

运行系统窗口的窗口句柄

dwFlags

按照标记在 WinUser.h (SW_SHOW, SW_HIDE, ...) 中的定义，标记在这里的应用与在 ::ShowWindow(HWND hwnd, int nCmdShow) 中的一样。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已显示应用程序窗口

FALSE

错误

错误消息

TLG_API_ERR_SUPPLY	发生错误
TLG_API_INVALID_PARAM	提供的参数无效/错误

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

WinUser.h

3.8.8 用于处理时间系统的函数

3.8.8.1 TLGEnumTime

说明

枚举项目所有检测和记录时间的名称

声明

```
BOOL TLGEnumTime (
    HANDLE                          hProject,
    TLG_ENUM_TIME_NAME_CALLBACK     lpCallbackFunc,
    PVOID                           lpUser,
    LPCMN_ERROR                     lpoes );
```

参数

hProject

要列出的时间对象所在的项目的句柄。

lpCallbackFunc

指向为每个现有时间对象调用的回调函数的指针。

lpUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpoes

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已列出时间对象

FALSE

错误

所需文件

pdecscli.h

pdecscli.lib

pdecscli.dll

相关函数

TLG_ENUM_TIME_NAME_CALLBACK (页 2613)	列出时间对象（回调）

示例

枚举所有采集和记录时间 (页 2630) "TL01.cpp"

参见

TLG_ENUM_TIME_NAME_CALLBACK (页 2613)

枚举所有采集和记录时间 (页 2630)

3.8.8.2 TLG_ENUM_TIME_NAME_CALLBACK

说明

为能够评估系统列出的时间对象，必须提供 TLG_ENUM_TIME_NAME_CALLBACK 类型的回调函数。

声明

```
BOOL ( * TLG_ENUM_TIME_NAME_CALLBACK) (
    LPTSTR    lpszName,
    PVOID     lpUser );
```

参数

lpszName

lpszName 指针指向第一个时间对象的名称。

lpUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

pdecscli.h

相关函数

TLGEnumTime (页 2612)	列出时间对象

示例

枚举所有采集和记录时间 (页 2630) "TL01.cpp"

参见

枚举所有采集和记录时间 (页 2630)

TLGEnumTime (页 2612)

3.8.8.3 TLGEnumTimes

说明

枚举所有检测和记录时间。

声明

```
BOOL TLGEnumTimes(
    TLG_ENUMTIMES_CALLBACK    lpfnCallback,
    LPVOID                    lpUser,
    PCMN_ERROR                lpError );
```

参数

lpfnCallback

指向回调函数的指针

lpUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已列出时间对象

FALSE

错误

错误消息

TLG_API_INVALID_PARAM	提供的参数无效/错误
TLG_API_ERR_SUPPLY	发生错误

TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

相关函数

TLG_ENUMTIMES_CALLBACK (页 2616)	列出时间对象（回调）

参见

TLG_ENUMTIMES_CALLBACK (页 2616)

3.8.8.4 TLG_ENUMTIMES_CALLBACK

说明

为能够评估系统列出的时间对象，必须提供 TLG_ENUMTIMES_CALLBACK 类型的回调函数。

声明

```
BOOL ( * TLG_ENUMTIMES_CALLBACK) (
    PTLG_TIMEDATA    lpTime,
    PVOID            lpUser );
```

参数

lpTime

系统会为每个时间对象保留 TLG_TIMEDATA (页 2535) 结构长度的临时存储。lpTime 指针指向第一个元素的开头。

lpUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

相关函数

TLGEnumTimes (页 2615)	列出时间对象

参见

TLGEnumTimes (页 2615)

TLG_TIMEDATA (页 2535)

3.8.8.5 TLGReadTime

说明

读取已创建时间对象的参数

声明

```
BOOL TLGReadTime (
    HANDLE           hProject,
    LPTSTR           lpszTimeName,
    PTLG_TIME_STR    lpTime,
    LPCMN_ERROR      lpoes );
```

参数

hProject

要处理的时间对象所在项目的句柄。

lpszTimeName

指向时间对象名称的指针

lpTime

具有时间对象数据的 TLG_TIME_STR (页 2533) 结构的地址。

lpoes

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已读取时间对象参数

FALSE

错误

所需文件

pdecscli.h

pdecscli.lib

pdecscli.dll

示例

读取时间对象的参数 (页 2644) "TL01.cpp"

参见

TLG_TIME_STR (页 2533)

读取时间对象的参数 (页 2644)

3.8.9 用于保存和存储的函数

3.8.9.1 TLGEnumBackupEntries

说明

此函数不再受支持，其提供 FALSE 值和错误代码 TLG_API_NOT_SUPPORTED 作为返回值。

声明

```
BOOL TLGEnumBackupEntries (
        LPTSTR                      lpszArchivName,
        TLG_ENUMBACKUP_ENTRIES      lpfnCallback,
        LPVOID                      lpUser,
        PCMN_ERROR                  lpError );
```

参数

lpszArchivName

指向日志名称的指针

lpfnCallback

指向为每个现有备份的回调函数的指针。

lpUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已列出备份

FALSE

错误

错误消息

TLG_API_INVALID_PARAM	提供的参数无效/错误
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接
TLG_API_NOT_SUPPORTED	不支持该函数

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

相关函数

TLG_ENUMBACKUP_ENTRIES (页 2621)	列出备份（回调）

参见

TLG_ENUMBACKUP_ENTRIES (页 2621)

3.8.9.2 TLG_ENUMBACKUP_ENTRIES

说明

不再支持 TLGEnumBackupEntries 函数。为能够评估系统列出的备份，必须提供 TLG_ENUMBACKUP_ENTRIES 类型的回调函数。

声明

```
BOOL ( * TLG_ENUMBACKUP_ENTRIES) (
    PTLG_BACKUP_TABLE_INFO    lpbti,
    PVOID                     lpUser );
```

参数

lpbti

系统会为每个备份保留 TLG_BACKUP_TABLE_INFO (页 2510) 结构长度的临时存储。lpbti 指针指向第一个元素的开头。

lpUser

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

注释

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

相关函数

TLGEnumBackupEntries (页 2619)	列出备份

参见

TLGEnumBackupEntries (页 2619)

TLG_BACKUP_TABLE_INFO (页 2510)

3.8.9.3 TLGExport

说明

使用此函数可导出日志的各部分。可通过 TLG_IO_BACKUP_SELECT 结构选择要导出的数据记录。

声明

```
BOOL TLGExport (
    LPTSTR                  lpszArchivName,
    LPTSTR                  lpszFileName,
    PTLG_IO_BACKUP_SELECT   lpibs,
    DWORD                   dwJobFlags,
    DWORD                   dwFormatFlags,
    PCMN_ERROR              lpError );
```

参数

lpszArchivName

指向日志名称的指针。该名称必须以“日志名称\变量名称”的形式指定。

可以通过 TLGEnumArchivsEx 和 TLGEnumVariablesEx 确定名称的各部分。

lpszFileName

指向数据要导出到的文件名称的指针。

lpibs

选择参数的传送结构 TLG_IO_BACKUP_SELECT (页 2522) 的地址。

dwJobFlags

该参数供以后升级使用，必须为其提供 0L。

dwFormatFlags

格式说明符：

TLG_BAKFMT_CSV:	CSV 格式（逗号分隔值）

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已导出日志数据

FALSE

错误

注释

此函数仅适用于顺序日志、循环日志和压缩日志。

错误消息

TLG_API_ERR_SUPPLY	发生错误
TLG_API_INVALID_PARAM	提供的参数无效/错误
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

参见

TLG_IO_BACKUP_SELECT (页 2522)

3.8.9.4 TLGGetBackupSize

说明

此函数不再受支持，其提供 FALSE 值和错误代码 TLG_API_NOT_SUPPORTED 作为返回值。

声明

```
BOOL TLGGetBackupSize (
    LPTSTR                  lpszArchivName,
    DWORD*                  lpdwSizeOfTable,
    PTLG_IO_BACKUP_SELECT   lpibs,
    DWORD                   dwJobFlags,
    DWORD                   dwFormatFlags,
    PCMN_ERROR              lpError );
```

参数

lpszArchivName

指向日志名称的指针

lpdwSizeOfTable

所选数据大小（以字节为单位）写入到的 DWORD 的地址。

lpibs

选择参数的传送结构 TLG_IO_BACKUP_SELECT (页 2522) 的地址。

dwJobFlags

特定作业标识符，可能的情况如下：

TLG_BACKUP_EXPORT:	仅备份，不删除源数据
TLG_BACKUP_EVACUATE:	导出并删除源数据

dwFormatFlags

格式说明符：

TLG_BAKFMT_CSV:	CSV 格式（逗号分隔值）-

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已确定数据大小

FALSE

错误

错误消息

TLG_API_ERR_SUPPLY	发生错误
TLG_API_INVALID_PARAM	提供的参数无效/错误
TLG_API_NO_INTERFACE	对级别较低的 COM 接口的访问失败
TLG_API_NO_CONNECTION	未与激活的项目建立任何连接
TLG_API_NOT_SUPPORTED	不支持该函数

所需文件

pdertcli.h

pde_glob.h

pdertdef.h

pdertcli.lib

pdertcli.dll

参见

TLG_IO_BACKUP_SELECT (页 2522)

3.8.10 示例

3.8.10.1 编辑曲线模板 - 示例 1

示例

```
// ========================================================================
// ========================================================================
// : Modul with examples to TagLogging-API
// ************************************************************************
// Copyright (C) 1995/96 SIEMENS AG, AUT 913 All rights reserved
// ************************************************************************
#include "stdafx.h" // if MFC classes
//#include "odkapi.h" // if console application
#include "TL02.h"
#include <string.h>
#include <stdlib.h>
#include <malloc.h>
#include <time.h>

// IMPLEMENTATION
//{{ODK_EXAMPLE}Edit curve template (example no. 1) (TLG)}
//{{FUNCTION}TLGConnect (TLG)}
//{{FUNCTION}TLGDisconnect (TLG)}
//{{FUNCTION}TLGInsertTemplateItem (TLG)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: MyTLGInsertTemplateItem(void) ODK TL RT
// ========================================================================
// : Edit curve template
// : Defines a new curve with previous settings
// ========================================================================
void MyTLGInsertTemplateItem (void)
{
    #define COLOR_RED 0x000000FF

//    Farbe           sym. Name       Value (hexadezimal)  Value (dezimal)
//    schwarz         CO_BLACK        0x00000000           0
//    weiß            CO_WHITE        0x00FFFFFF           16777215
//    rot             CO_RED          0x000000FF           255
//    dunkelrot       CO_DKRED        0x00000080           128
//    grün            CO_GREEN        0x0000FF00           65280
//    dunkelgrün      CO_DKGREEN      0x00008000           32768
//    blau            CO_BLUE         0x00FF0000           16711680
//    dunkelblau      CO_DKBLUE       0x00800000           8388608
//    gelb            CO_YELLOW       0x0000FFFF           65535
//    dunkelgelb      CO_DKYELLOW     0x00008080           32896
//    zyan            CO_CYAN         0x00FFFF00           1048320
```

```
//    dunkelzyan      CO_DKCYAN       0x00808000          8421376
//    magenta         CO_MAGENTA      0x00FF00FF          16711935
//    dunkelmagenta   CO_DKMAGENTA    0x00800080          8388736
//    hellgrau        CO_LTGRAY       0x00C0C0C0          12632256
//    dunkelgrau      CO_DKGRAY       0x00808080          8421504

    // scripts
    BOOL ret = FALSE;
    CMN_ERROR Error;
    TCHAR szText[255];
    HWND hwndParent = NULL;
    TLG_TEMPLATEITEM_INFO TemplateItem;
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = TLGConnect(hwndParent, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TLGInsertTemplateItem: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
    else
    {
        memset(&TemplateItem, 0, sizeof(TLG_TEMPLATEITEM_INFO));
        _tcsncpy_s(TemplateItem.szArchivName, _countof(TemplateItem.szArchivName),
_T("PLSMWA"), _TRUNCATE);    // archive name
        _tcsncpy_s(TemplateItem.szTemplateName, _countof(TemplateItem.szTemplateName),
_T("KV_PID1"), _TRUNCATE);   // template for curve window
        _tcsncpy_s(TemplateItem.szVariableName, _countof(TemplateItem.szVariableName),
_T("VPID1_X"), _TRUNCATE);   // DM variable name
       _tcsncpy_s(TemplateItem.szDMVariableName, _countof(TemplateItem.szDMVariableName),
_T("VPID1_X"), _TRUNCATE);
        _tcsncpy_s(TemplateItem.szTemplateItemName,
_countof(TemplateItem.szTemplateItemName), _T("KPID3_X"), _TRUNCATE);   // curve name
        TemplateItem.dwTemplateTyp = TLG_TEMPLATE_CURVE;
        TemplateItem.dwTemplateItemTyp = TLG_TEMPLATEITEM_CURVE;
        TemplateItem.dwReadAccessLevel = 1;
        TemplateItem.dwWriteAccessLevel = 1;
        TemplateItem.dwArchivTyp = TLG_ARCTYP_PROCESS; //_USER, _PROCESS, _COMPRESS
        _tcsncpy_s(TemplateItem.szTimeNameRange, _countof(TemplateItem.szTimeNameRange),
_T("500 ms"), _TRUNCATE);
        TemplateItem.fVisible = TRUE;
        TemplateItem.tplInfo.tplCurve.csx.dwDataTyp = TLG_DATATYP_TIMERANGE;
        TemplateItem.tplInfo.tplCurve.csx.fAutoRange = TRUE;
        TemplateItem.tplInfo.tplCurve.csx.fActualize = TRUE;
        TemplateItem.tplInfo.tplCurve.csx.dwBufferSize = 10;
        TemplateItem.tplInfo.tplCurve.csy.dwDataTyp = TLG_DATATYP_BLOCKDATA;
        TemplateItem.tplInfo.tplCurve.csy.fAutoRange = TRUE;
        TemplateItem.tplInfo.tplCurve.csy.crColor = COLOR_RED;
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = TLGInsertTemplateItem(TemplateItem.szTemplateName, &TemplateItem, &Error);
```

```
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TLGInsertTemplateItem: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
             _sntprintf_s(szText,_countof(szText), _TRUNCATE, _T("InsertItem:
TemplateItemName= %s  "), TemplateItem.szTemplateItemName);
        }
        ODKTrace(szText);
        // Disconnect
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = TLGDisconnect(&Error);
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGDisconnect:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
            ODKTrace(szText);
        }
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

TLGInsertTemplateItem (页 2605)

TLGConnect (页 2552)

TLGDisconnect (页 2554)

3.8.10.2 枚举所有采集和记录时间

示例

```
/ ======================================================================
// ======================================================================
// : Modul with examples to TagLogging-API
// ***********************************************************************
// Copyright (C) 1995/96 SIEMENS AG, AUT 913 All rights reserved
// ***********************************************************************
#include "stdafx.h" // if MFC classes
#include "stdlib.h"
#include "TL01.h"
#include "dm01.h"
// IMPLEMENTATION
// ======================================================================
// 4. Callback
// ======================================================================
//{{ODK_EXAMPLE}Enum all acquisition and archiving times (TLG)}
//{{FUNCTION}TLGCSConnect (TLG)}
//{{FUNCTION}TLGCSDisConnect (TLG)}
//{{FUNCTION}TLGEnumTime (TLG)}
//{{FUNCTION}TLG_ENUM_TIME_NAME_CALLBACK (TLG)}
//{{FUNCTION}(END)}
// ======================================================================
// Function: TLGEnumTime(void) ODK TL CS
// ======================================================================
// : Enum all acquisition and archiving times
// :
// ======================================================================
BOOL MyTLGEnumTimeNameCallback (LPTSTR szTime, PVOID lpUser)
{
    TCHAR szText[255];
    sprintf(szText,"...Enum Time %s", szTime);
    ODKTrace(szText);
    //printf("%s\r\n",szText);
    return TRUE;
}

void MyTLGEnumTime(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    TCHAR szText[255];
    HANDLE hProject;
    CMN_ERROR Error;
    PVOID pUser = NULL;
    sprintf(szText, "TLGEnumTime");
    ODKTrace(szText);
```

```
//printf("%s\r\n",szText);
MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
memset(&Error,0,sizeof(CMN_ERROR));
ret = TLGCSConnect(NULL, &Error);
if (FALSE == ret)
{
    sprintf(szText, "Error in TLGCSConnect: E1= 0x%08lx ; E2= 0x%08lx ; %s",
        Error.dwError1, Error.dwError2, Error.szErrorText);
    ODKTrace(szText);
    //printf("%s\r\n",szText);
}
else
{
    sprintf(szText, " TLGCSConnect");
    ODKTrace(szText);
    //printf("%s\r\n",szText);
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = TLGOpenProject(&hProject, /*PROJ_PATH*/g_szProjectFile, NULL, &Error);
    if (FALSE == ret)
    {
        sprintf(szText, "Error in TLGOpenProject: E1= 0x%08lx ; E2= 0x%08lx ; %s",
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
        //printf("%s\r\n",szText);
    }
    else //TLGOpenProject OK
    {
        sprintf(szText, " TLGOpenProject");
        ODKTrace(szText);
        //printf("%s\r\n",szText);
        // read projected archiving times
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = TLGEnumTime(hProject, MyTLGEnumTimeNameCallback, pUser, &Error);
        if (FALSE == ret)
        {
            sprintf(szText, "Error in TLGEnumTime: E1= 0x%08lx ; E2= 0x%08lx ; %s",
            Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            sprintf(szText, " TLGEnumTime");
        }
        ODKTrace(szText);
        //printf("%s\r\n",szText);
        //TLGCloseProject();
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = TLGCloseProject(hProject, &Error);
        if(FALSE == ret)
        {
            sprintf(szText, "Error in TLGCloseProject: E1= 0x%08lx ; E2= 0x%08lx ; %s",
            Error.dwError1, Error.dwError2, Error.szErrorText);
```

```
            }
            else
            {
                sprintf(szText, " TLGCloseProject");
            }
            ODKTrace(szText);
            //printf("%s\r\n",szText);
        }
        // Disconnect
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = TLGCSDisConnect(&Error);
        if (FALSE == ret)
        {
            sprintf(szText, "Error in TLGCSDisConnect: E1= 0x%08lx ; E2= 0x%08lx ; %s",
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            sprintf(szText, " TLGCSDisConnect");
        }
        ODKTrace(szText);
        //printf("%s\r\n",szText);
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

TLGCSConnect (页 2546)

TLG_ENUM_TIME_NAME_CALLBACK (页 2613)

TLGEnumTime (页 2612)

TLGDisconnect (页 2554)

TLGCSDisConnect (页 2549)

3.8.10.3 枚举日志

示例

```
//{{ODK_EXAMPLE}Enum archives (TLG)}
//{{FUNCTION}TLGCSConnect (TLG)}
//{{FUNCTION}TLGCSDisConnect (TLG)}
//{{FUNCTION}TLGOpenProject (TLG)}
//{{FUNCTION}TLGCloseProject (TLG)}
//{{FUNCTION}TLGEnumArchives (TLG)}
//{{FUNCTION}TLG_ENUM_ARCHIV_CALLBACK (TLG)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: TLGEnumArchives(void) ODK TL CS
// ========================================================================
// : Enum archives
// :
// ========================================================================
BOOL MyTLGEnumArchivCallback(LPTSTR szArchives, PVOID lpUser)
{
    lpUser;
    TCHAR szText[255];
   _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...Enum Archive=%s "),szArchives);
    ODKTrace(szText);
    return TRUE;
}

void MyTLGEnumArchives(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    CMN_ERROR Error;
    HANDLE hProject;
    PVOID pUser = NULL;
    TCHAR szText[255];
    TCHAR szArchivName[255];
    _tcsncpy_s(szArchivName, _countof(szArchivName), _T("PLSMWA"), _TRUNCATE);
    MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
    //TLGConnect
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = TLGCSConnect(NULL, &Error);
    if (FALSE == ret)
    {
          _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGCSConnect: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
               Error.dwError1, Error.dwError2, Error.szErrorText);
          ODKTrace(szText);
        }
        else
```

```
        {
            //TLGOpenProject
            memset(&Error,0,sizeof(CMN_ERROR));
            ret = TLGOpenProject(&hProject, /*PROJ_PATH*/g_szProjectFile, NULL, &Error);
            if (FALSE == ret)
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TLGOpenProject: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
            ODKTrace(szText);
        }
        else
        {
            // TLGEnumArchives
            memset(&Error,0,sizeof(CMN_ERROR));
            ret = TLGEnumArchives(hProject, MyTLGEnumArchivCallback, pUser, &Error);
            if (FALSE == ret)
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TLGEnumArchives: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
            }
            else
            {
               _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   TLGEnumArchives"));
            }
            ODKTrace(szText);
        }
        //CloseProject
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = TLGCloseProject(hProject,&Error);
        if(FALSE == ret)
        {
           _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGCloseProject:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   TLGCloseProject"));
        }
        ODKTrace(szText);
    }
    // Disconnect
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = TLGCSDisConnect(&Error);
    if (FALSE == ret)
    {
       _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGCSDisConnect: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
```

```
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   TLGCSDisConnect"));
    }
    ODKTrace(szText);
}
//{{ODK_EXAMPLE}(END)}
```

参见

TLGCSConnect (页 2546)

TLGOpenProject (页 2557)

TLGCloseProject (页 2555)

TLGEnumArchives (页 2570)

TLG_ENUM_ARCHIV_CALLBACK (页 2572)

TLGCSDisConnect (页 2549)

3.8.10.4 枚举日志的变量

示例

```
//{{ODK_EXAMPLE}Enum variables of a archive (TLG)}
//{{FUNCTION}TLG_ENUM_PROJECT_NAME_CALLBACK (TLG)}
//{{FUNCTION}TLGEnumVariablesEx (TLG)}
//{{FUNCTION}TLG_ENUMVARIABLES (TLG)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: MyTLGEnumVariablesEx(void) ODK TL CS
// ========================================================================
// : Edit archive variables
// :
// ========================================================================
BOOL MyTLGEnumVariablesExCallback(PTLG_VARIABLE_INFO lpvi, LPVOID lpUser)
{
    lpUser;
    TCHAR szText[255];
     _sntprintf_s(szText,_countof(szText), _TRUNCATE, _T("Enum Variables %d %s "), lpvi-
>dwVariableTyp, lpvi->szVariableName);
    ODKTrace(szText);
    return TRUE;
}

void MyTLGEnumVariablesEx(void)
{
    BOOL ret = FALSE;
    TCHAR szText[255];
    CMN_ERROR Error;
    HANDLE hProject = NULL;
    TCHAR szArchivName[255];
    HWND hwndParent = NULL;
    VOID* pUser = NULL;
    _tcsncpy_s(szArchivName, _countof(szArchivName), _T("PLSMWA"), _TRUNCATE);
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = TLGConnect(hwndParent, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGConnect: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TLGConnect"));
        ODKTrace(szText);
```

```
        // Info
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = TLGEnumVariablesEx (szArchivName, MyTLGEnumVariablesExCallback,
                pUser, &Error );
        if (FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TLGEnumVariablesEx: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
           _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TLGEnumVariablesEx: OK"));
        }
        ODKTrace(szText);

        memset(&Error,0,sizeof(CMN_ERROR));
        ret = TLGDisconnect(&Error);
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGDisconnect:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
            ODKTrace(szText);
        }
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

TLGOpenProject (页 2557)

TLGEnumVariablesEx (页 2565)

TLG_ENUMVARIABLES (页 2567)

3.8.10.5 枚举日志

示例

```
//{{ODK_EXAMPLE}Enumerate archives (TLG)}
//{{FUNCTION}TLGConnect (TLG)}
//{{FUNCTION}TLGDisconnect (TLG)}
//{{FUNCTION}TLGEnumArchivs (TLG)}
//{{FUNCTION}TLG_ENUMTABLES (TLG)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: MyTLGEnumArchivs(void) ODK TL RT
// ========================================================================
// : Enumerate archives
// ========================================================================
BOOL MyTLGEnumTablesCallback(LPTSTR lpTableName, PTLG_TABLE_INFO lpti, PVOID lpUser)
{
    lpUser;
    TCHAR szText[255];
     _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TableName=%s"), lpTableName);
    ODKTrace(szText);
     _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T(" ...ArchivTyp=%d"), lpti-
>dwArchivTyp);
    ODKTrace(szText);
     _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T(" ...ArchivName=%s"),lpti-
>szArchivName);
    ODKTrace(szText);
     _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T(" ...dwSaveTyp=%d"), lpti-
>dwSaveTyp);
    ODKTrace(szText);
    return TRUE;
}

void MyTLGEnumArchivs(void)
{
    BOOL ret = FALSE;
    //DWORD StartTime = 0;
    //DWORD StopTime = 0;
    void* lpUser = NULL;
    CMN_ERROR Error;
    TCHAR szText[255];
    HWND hwndParent = NULL;
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = TLGConnect(hwndParent, &Error);
    if(FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGConnect: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
```

```
        ODKTrace(szText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TLGConnect"));
        ODKTrace(szText);

        memset(&Error,0,sizeof(CMN_ERROR));
        ret = TLGEnumArchivs(TLG_ARCTYP_PROCESS, MyTLGEnumTablesCallback, lpUser, &Error);
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGEnumArchivs:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TLGEnumArchivs: OK"));
        }
        ODKTrace(szText);
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = TLGDisconnect(&Error);
        if(FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGDisconnect:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
            ODKTrace(szText);
        }
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

TLGConnect (页 2552)

TLGDisconnect (页 2554)

TLGEnumArchivs (页 2573)

3.8.10.6 读取日志

示例

```
//{{ODK_EXAMPLE}Read archive (TLG)}
//{{FUNCTION}TLGCSConnect (TLG)}
//{{FUNCTION}TLGCSDisConnect (TLG)}
//{{FUNCTION}TLGOpenProject (TLG)}
//{{FUNCTION}TLGCloseProject (TLG)}
//{{FUNCTION}TLGReadArchiv (TLG)}
//{{FUNCTION}TLGReadVariable (TLG)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: TLGReadArchiv(void) ODK TL CS
// ========================================================================
// : Read archive
// :
// ========================================================================
void MyTLGReadArchiv(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    CMN_ERROR Error;
    HANDLE hProject;
    //PVOID pUser = NULL;
    TCHAR szText[255];
    TCHAR szArchivName[255];
    TCHAR szVarName[255];
    TLG_ARCHIV_STR TLGArchivStruct;
    TLG_VAR_STR TLGVarStruct;
    _tcsncpy_s(szArchivName, _countof(szArchivName), _T("PLSMWA"), _TRUNCATE);
    _tcsncpy_s(szVarName, _countof(szVarName), _T("X"), _TRUNCATE);
    MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = TLGCSConnect(NULL, &Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGCSConnect: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
    else
    {
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = TLGOpenProject(&hProject, /*PROJ_PATH*/g_szProjectFile, NULL, &Error);
        if (FALSE == ret)
        {
```

```
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGOpenProject:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
            ODKTrace(szText);
        }
        else
        {
            // read
            memset(&TLGArchivStruct,0,sizeof(TLG_ARCHIV_STR));
            memset(&Error,0,sizeof(CMN_ERROR));
            ret = TLGReadArchiv(hProject, szArchivName, &TLGArchivStruct, &Error);
            if (FALSE == ret)
            {
              _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGReadArchiv:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
                ODKTrace(szText);
            }
            else
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TLGReadArchiv"));
                ODKTrace(szText);
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   ArcName=%s
Comments=%s dwTyp=0x%04x dwRecSize=%d"),
                TLGArchivStruct.szName,
                TLGArchivStruct.szComment,
                TLGArchivStruct.dwTyp,
                TLGArchivStruct.dwRecordSize);
                ODKTrace(szText);
            }
            memset(&TLGVarStruct, 0, sizeof(TLG_VAR_STR));
            memset(&Error,0,sizeof(CMN_ERROR));
           ret = TLGReadVariable(hProject, szArchivName, szVarName, &TLGVarStruct, &Error);
            if (FALSE == ret)
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TLGReadVariable: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
                ODKTrace(szText);
            }
            else
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TLGReadVariable"));
                ODKTrace(szText);
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   szVarName=%s"),
TLGVarStruct.szVarName);
                ODKTrace(szText);
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   szProcName=%s"),
TLGVarStruct.szProcName);
                ODKTrace(szText);
```

```
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   dwVarType=0x%04X"),
TLGVarStruct.dwVarType);
                ODKTrace(szText);
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("
dwArchivStyle=0x%04X"), TLGVarStruct.dwArchivStyle);
                ODKTrace(szText);
               _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   szWriteBackTo=%s"),
TLGVarStruct.szWriteBackTo);
                ODKTrace(szText);
               _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   dwSupply=0x%04X"),
TLGVarStruct.dwSupply);
                ODKTrace(szText);
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   fLocked=%d"),
TLGVarStruct.fLocked);
                ODKTrace(szText);
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   szComment=%s"),
TLGVarStruct.szComment);
                ODKTrace(szText);
               _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   szRecordCycle=%s"),
TLGVarStruct.szRecordCycle);
                ODKTrace(szText);
               _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   szArchivCycle=%s"),
TLGVarStruct.szArchivCycle);
                ODKTrace(szText);
              _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   dwMultiple=0x%04X"),
TLGVarStruct.dwMultiple);
                ODKTrace(szText);
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("
dwValueFlow=0x%04X"), TLGVarStruct.dwValueFlow);
                ODKTrace(szText);
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("
dwValueFollow=0x%04X"), TLGVarStruct.dwValueFollow);
                ODKTrace(szText);
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("
dwSaveByFault=0x%04X"), TLGVarStruct.dwSaveByFault);
                ODKTrace(szText);
            }
        }
        //CloseProject
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = TLGCloseProject(hProject,&Error);
        if(FALSE == ret)
        {
           _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGCloseProject:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   TLGCloseProject"));
        }
```

```
        ODKTrace(szText);
    }
    // Disconnect
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = TLGCSDisConnect(&Error);
    if (FALSE == ret)
    {
       _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGCSDisConnect: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
    }
    else
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   TLGCSDisConnect"));
    }
    ODKTrace(szText);
}
//{{ODK_EXAMPLE}(END)}
```

参见

TLGOpenProject (页 2557)

TLGCloseProject (页 2555)

TLGCSConnect (页 2546)

TLGReadVariable (页 2568)

TLGReadArchiv (页 2601)

TLGCSDisConnect (页 2549)

3.8.10.7 读取时间对象的参数

示例

```
//{{ODK_EXAMPLE}Read parameters of time object (TLG)}
//{{FUNCTION}TLGCSConnect (TLG)}
//{{FUNCTION}TLGCSDisConnect (TLG)}
//{{FUNCTION}TLGReadTime (TLG)}
//{{FUNCTION}TLGOpenProject (TLG)}
//{{FUNCTION}TLGCloseProject (TLG)}
//{{FUNCTION}(END)}
// ========================================================================
// Function: TLGReadTime(void) ODK TL CS
// ========================================================================
// : Reads the parameters of an already existing time object.
// :
// ========================================================================
void MyTLGReadTime(void)
{
    // #define PROJ_PATH "C:\\siemens\\odk\\samples\\projects\\demo\\odk.mcp"
    BOOL ret = FALSE;
    TCHAR szText[255];
    HANDLE hProject;
    CMN_ERROR Error;
    TLG_TIME_STR TLGTimeStruct;
    TCHAR szTime[255];
    _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("TLGReadTime"));
    ODKTrace(szText);
    MyDMEnumOpenedProjects(); // open the DM and set the g_szProjectFile
    memset(&Error,0,sizeof(CMN_ERROR));
    ret = TLGCSConnect(NULL, &Error);
    if (FALSE == ret)
    {
        _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGCSConnect: E1=
0x%08lx ; E2= 0x%08lx ; %s"),
            Error.dwError1, Error.dwError2, Error.szErrorText);
        ODKTrace(szText);
    }
    else
    {
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = TLGOpenProject(&hProject, /*PROJ_PATH*/g_szProjectFile, NULL, &Error);
        if (FALSE == ret)
        {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGOpenProject:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
            ODKTrace(szText);
        }
```

```
        else
        {
            // read properties of selected time
            memset(&Error,0,sizeof(CMN_ERROR));
            memset(&TLGTimeStruct, 0, sizeof(TLG_TIME_STR));
            _tcsncpy_s(szTime, _countof(szTime), _T("500 ms"), _TRUNCATE);
            ret = TLGReadTime(hProject, szTime, &TLGTimeStruct, &Error);
            if (FALSE == ret)
            {
               _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGReadTime:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
                ODKTrace(szText);
            }
            else
            {
                 _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...szTimeName = %s
"),TLGTimeStruct.szTimeName);
                ODKTrace(szText);
                 _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...dwBasis = %10d
Millisec"), TLGTimeStruct.dwBasis);
                ODKTrace(szText);
               _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("...dwFactor = %10d "),
TLGTimeStruct.dwFactor);
                ODKTrace(szText);
            }
            // save changes permanent
            memset(&Error, 0, sizeof(CMN_ERROR));
            ret = TLGSaveProject(hProject , &Error);
            if (FALSE == ret)
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TLGSaveProject: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
            }
            else
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   TLGSaveProject"));
            }
            ODKTrace(szText);
            //TLGCloseProject();
            memset(&Error, 0, sizeof(CMN_ERROR));
            ret = TLGCloseProject(hProject, &Error);
            if(FALSE == ret)
            {
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in
TLGCloseProject: E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
            }
            else
            {
```

```
                _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   TLGCloseProject"));
            }
            ODKTrace(szText);
        }
        // Disconnect
        memset(&Error,0,sizeof(CMN_ERROR));
        ret = TLGCSDisConnect(&Error);
        if (FALSE == ret)
        {
           _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("Error in TLGCSDisConnect:
E1= 0x%08lx ; E2= 0x%08lx ; %s"),
                Error.dwError1, Error.dwError2, Error.szErrorText);
        }
        else
            {
            _sntprintf_s(szText, _countof(szText), _TRUNCATE, _T("   TLGCSDisConnect"));
        }
        ODKTrace(szText);
    }
}
//{{ODK_EXAMPLE}(END)}
```

参见

TLGOpenProject (页 2557)

TLGCloseProject (页 2555)

TLGCSConnect (页 2546)

TLGReadTime (页 2618)

TLGCSDisConnect (页 2549)

3.9 配方函数

3.9.1 基本知识

3.9.1.1 函数概览

概述

uaSetLocalEvents (页 2664)	设置本地事件。
uaArchiveClose (页 2677)	关闭与当前用户归档的连接
uaArchiveDelete (页 2678)	删除用户归档中的数据记录
uaArchiveExport (页 2680)	导出用户归档
uaArchiveGetCount (页 2692)	读取数据记录的数量
uaArchiveGetFieldLength (页 2702)	读取当前字段长度
uaArchiveGetFieldName (页 2703)	读取当前字段名称
uaArchiveGetFields (页 2705)	确定字段数
uaArchiveGetFieldType (页 2706)	读取当前字段类型
uaArchiveGetFieldValueDate (页 2707)	从当前数据字段中读取日期和时间
uaArchiveGetFieldValueDouble (页 2708)	从当前数据字段中读取双精度值
uaArchiveGetFieldValueLong (页 2710)	从当前数据字段中读取长值
uaArchiveGetFieldValueString (页 2711)	从当前数据字段中读取字符串
uaArchiveGetFilter (页 2694)	读取过滤器设置
uaArchiveGetID (页 2681)	读取当前数据字段的 ID
uaArchiveGetName (页 2682)	读取归档名称
uaArchiveGetSort (页 2684)	读取当前数据字段的排序
uaArchiveImport (页 2685)	导入用户归档
uaArchiveInsert (页 2695)	将数据记录插入到用户归档中
uaArchiveMoveFirst (页 2696)	移动到第一条数据记录
uaArchiveMoveLast (页 2697)	移动到最后一条数据记录
uaArchiveMoveNext (页 2698)	移动到下一条数据记录
uaArchiveMovePrevious (页 2700)	移动到上一条数据记录
uaArchiveOpen (页 2687)	创建与当前用户归档的连接

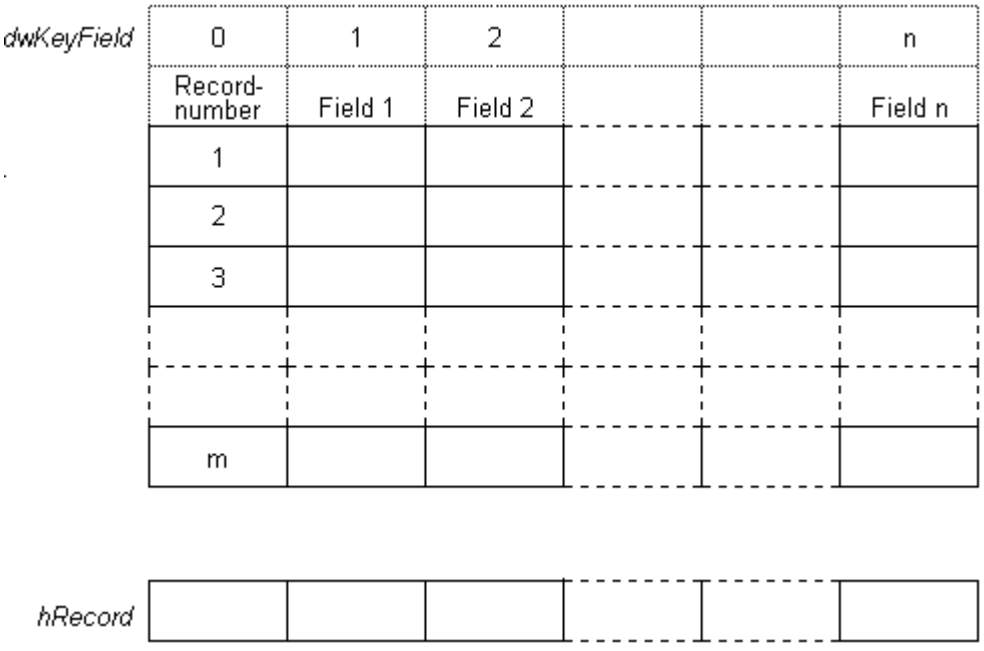
uaArchiveReadTagValuesByName (页 2726)	根据名称读取变量值
uaArchiveReadTagValues (页 2725)	读取变量值
uaArchiveRequery (页 2720)	重新加载到当前用户归档
uaArchiveSetFieldValueDate (页 2713)	向当前数据字段写入日期和时间
uaArchiveSetFieldValueDouble (页 2714)	向当前数据字段写入双精度值
uaArchiveSetFieldValueLong (页 2715)	向当前数据字段写入长值
uaArchiveSetFieldValueString (页 2717)	向当前数据字段写入字符串
uaArchiveSetFilter (页 2721)	设置过滤器
uaArchiveSetSort (页 2722)	设置排序条件
uaArchiveUpdate (页 2688)	将数据记录更新到用户归档中
uaArchiveWriteTagValuesByName (页 2730)	根据名称向变量写入数据记录
uaArchiveWriteTagValues (页 2728)	向变量写入数据记录
uaConnect (页 2665)	建立与用户归档的连接（运行系统）。
uaDisconnect (页 2666)	关闭与用户归档的连接
uaGetArchive (页 2689)	读取归档组态
uaGetField (页 2718)	读取字段组态
uaGetLastError (页 2660)	查询用户归档中的上一个错误
uaGetNumArchives (页 2690)	查找已组态用户归档的数量
uaGetNumFields (页 2701)	确定字段数
uaIsActive (页 2662)	查询运行系统中是否有归档打开
uaOpenArchives (页 2691)	查询运行系统中打开了多少个用户归档
uaOpenViews (页 2724)	查询运行系统中打开了多少个用户归档视图
uaQueryArchiveByName (页 2670)	根据名称设置与用户归档的连接（运行系统）
uaQueryArchive (页 2668)	设置与用户归档的连接（运行系统）
UaQueryConfiguration (页 2673)	查询与组态的连接
uaReleaseArchive (页 2674)	关闭与用户归档的连接（运行系统）
uaReleaseConfiguration (页 2675)	关闭与组态的连接

参见

uaUsers (页 2663)

3.9.1.2 配方的结构

简介



用户归档是数据库中的各个表，除了表的第一列外，用户可以通过任何方式对其进行构造。数据类型限制在 char、double、int 和 DateTime。

GetField 和 SetField 函数对内部 hRecord 数据记录起作用，可以使用 Insert 和 Update 将该数据记录写入归档。

如果直接通过数据库 (ODBC/SQL) 修改用户归档，则不会执行在线同步。

用户归档的组态

第一步是组态用户归档。 可以使用用户归档编辑器、用户归档脚本函数或用户归档 API 函数执行此组态。 在这种情况下，脚本函数与 API 函数相同。

使用用户归档 API 函数进行组态

UAQueryConfiguration 函数为组态函数提供了一个句柄 (UAHCONFIG)。 此句柄可用于调用组态函数 UASetArchive、UAAddArchive、UASetField、UAAddField 等。
“UAReleaseConfiguration”函数完成用户归档的组态。

在多客户机项目中不能使用这些函数。

建立与用户归档的连接

接下来，必须调用标准函数 UAConnect 来建立到用户归档组件的连接。 UAConnect 将创建 UAHCONNECT 句柄，它可以使归档（而不是视图）打开和关闭。

运行时模式下的信息函数

UAIsActive 函数指示是否在运行时激活并打开归档。 UAUsers 返回登录用户的数目，UAOpenArchives 返回打开归档的数目，UAOpenViews 返回打开视图的数目。

运行时函数的打开

运行时模式需要组态的用户归档。 UAQueryArchive 和 UAQueryArchiveByName 为运行时函数提供了句柄。 一旦使用 UAArchiveOpen 函数打开归档，便可以使用用户归档运行时函数。

运行时模式的函数

UAArchiveMoveNext、UAArchiveMovePrevious、UAArchiveMoveFirst 和 UAArchiveMoveLast 函数用于移动游标。 hArchive 句柄用于确保用户归档中数据记录分配的唯一性。 此分配还允许间接寻址，例如满足画面掩码的要求。

UAArchiveUpdate 函数可将临时数据记录保存在归档中，此操作将覆盖游标当前所在位置的数据记录。 此数据记录必须预先使用 UAArchiveMoveNext、UAArchiveMovePrevious、UAArchiveMoveFirst 或 UAArchiveMoveLast 函数进行读取。

关闭与用户归档的连接

UAArchiveClose 函数用于关闭用户归档。 UAReleaseArchive 函数关闭与当前归档的连接，然后 UADisconnect 函数终止与用户归档组件的连接。

3.9.1.3 API 函数的调用序列的相关性

运行时函数的调用序列的相关性

通过“用户归档”函数 UAConnect 可创建 UAHCONNECT 句柄。 打开和关闭归档（而非视图）时需要此句柄。 因此，为了获取 UAHCONNECT 句柄，必须首先调用 UAConnect 函数。然后，为打开和关闭归档（而非视图），可使用该句柄调用 API 函数，如下所列。 要完成组态，必须调用 UADisconnect 函数。

通过 UAQueryArchive 和 UAQueryArchiveByName 函数可创建 UAHARCHIVE 句柄。 能够打开运行时模式下的归档的“用户归档 API”函数 UAArchiveOpen 需要使用该句柄。

运行时归档函数的句柄

UAConnect 句柄 UAHCONNECT
--->
适用对象 UAIsActive
UAUsers
UAOpenArchives
UAOpenViews
UAQueryArchive ---> 句柄 UAHARCHIVE
UAQueryArchiveByName ---> 句柄 UAHARCHIVE
适用对象：
UAArchiveOpen
--->
适用对象 UAArchiveDelete
UAArchiveExport
UAArchiveGetCount
UAArchiveGetFieldLength
UAArchiveGetFields
UAArchiveGetFieldType
UAArchiveGetFieldValueDate
UAArchiveGetFieldValueDouble
UAArchiveGetFieldValueFloat
UAArchiveGetFieldValueLong
UAArchiveGetFieldValueString
UAArchiveGetFieldName
UAArchiveGetFilter
UAArchiveGetID
UAArchiveGetName
UAArchiveGetSort
UAArchiveImport
UAArchiveInsert
UAArchiveMoveFirst
UAArchiveMoveLast

UAArchiveMoveNext
UAArchiveMovePrevious
UAArchiveReadTagValues
UAArchiveReadTagValuesByNam
e
UAArchiveRequery
UAArchiveSetFieldValueDate
UAArchiveSetFieldValueDouble
UAArchiveSetFieldValueFloat
UAArchiveSetFieldValueLong
UAArchiveSetFieldValueString
UAArchiveSetFilter
UAArchiveSetSort
UAArchiveUpdate
UAArchiveWriteTagValues
UAArchiveWriteTagValuesByNam
e
<--- UAArchiveClose

<---
UAReleaseArchive

<--- UADisconnect

3.9.1.4 错误消息

概述

可使用 API 函数 UAGetLastError 查询以下错误消息。

UA_ERROR_SUCCESS	0	无错误，一切正常。
UA_ERROR_GENERIC	1	组态函数出错。
UA_ERROR_CONNECT_FAILED	100	未建立运行系统连接（非运行系统）。
UA_ERROR_OPEN_FAILED	101	无法打开归档。
UA_ERROR_CLOSE_FAILED	102	无法关闭归档，或者句柄无效。

UA_ERROR_REQUERY_FAILED	103	无法重新加载到带有诸如新过滤器/新排序的归档中。
UA_ERROR_MOVE_FAILED	104	无法移动到第一个、最后一个、下一个或上一个数据记录。
UA_ERROR_INSERT_FAILED	105	数据记录无法插入到指定位置（请检查位置）。
UA_ERROR_UPDATE_FAILED	106	无法更新指定数据记录（已锁定或该位置不存在数据记录）。
UA_ERROR_DELETE_FAILED	107	无法删除数据记录。
UA_ERROR_IMPORT_FAILED	108	归档导入失败（请检查格式、大小）。
UA_ERROR_EXPORT_FAILED	109	未能导出用户归档（请检查空间）。
UA_ERROR_READ_FAILED	110	无法读取变量。
UA_ERROR_WRITE_FAILED	111	无法写入变量
UA_ERROR_GET_FAILED	112	无法读取归档/字段描述和/或数据记录值。
UA_ERROR_SET_FAILED	113	无法写入归档/字段描述和/或数据记录值。
UA_ERROR_INVALID_NAME	200	名称无效。
UA_ERROR_INVALID_TYPE	201	指定的类型不正确
UA_ERROR_INVALID_NUMRECS	202	无法接收数据记录数量
UA_ERROR_INVALID_COMMTYPE	203	所选连接（无/原始/直接）不正确
UA_ERROR_INVALID_LENGTH	204	未接收指定（字符串）长度
UA_ERROR_INVALID_PRECISION	205	不会接收小数点后的指定位数。
UA_ERROR_NULL_POINTER	1000	已传送 NULL 指针
UA_ERROR_INVALID_POINTER	1001	指定的指针无效
UA_ERROR_INVALID_HANDLE	1002	指定的句柄无效
UA_ERROR_INVALID_INDEX	1003	指定的索引无效
UA_ERROR_SERVER_UNKNOWN	1004	指定的使用中的服务器无效。

3.9.1.5 常数

组态常量：

归档类型

UA_ARCHIVETYPE_UNKNOWN	0	未知归档类型
UA_ARCHIVETYPE_UNLIMITED	1	循环归档
UA_ARCHIVETYPE_LIMITED	2	带有固定数量的数据记录的归档

归档标记

UA_ARCHIVEFLAG_ACCESS	0x00000001	还创建另一个显示最后一次访问的时间的列
UA_ARCHIVEFLAG_USER	0x00000002	还创建另一个显示用户名的列

通信类型

UA_COMMTYPE_NONE	1	无变量连接
UA_COMMTYPE_RAW	2	通过原始数据变量进行的通信
UA_COMMTYPE_DIRECT	3	通过 DM 变量进行的通信

字段类型

UA_FIELDTYPE_INTEGER	1	INT 字段类型
UA_FIELDTYPE_DOUBLE	2	双精度字段类型
UA_FIELDTYPE_FLOAT	3	浮点型字段类型
UA_FIELDTYPE_STRING	6	空终止字符串/字节数组字段类型
UA_FIELDTYPE_DATETIME	8	日期和时间的字段类型

字段标记

UA_FIELDFLAG_UNIQUE	0x00000001	字段必须包含唯一值
UA_FIELDFLAG_NOTNULL	0x00000002	字段必须包含值
UA_FIELDFLAG_INDEX	0x00000010	索引应支持字段

最大长度

UA_MAXLEN_NAME	20	归档/字段名称的最大长度
UA_MAXLEN_ALIAS	50	次要补充名称/注释的最大长度
UA_MAXLEN_PLCID	8	通信设备名称的最大长度
UA_MAXLEN_DMVARNAME	128	数据管理器变量名称的最大长度
UA_MAXLEN_VALUE	255	字符串变量的最大长度

参数常量：

动作

UA_ACTION_GENERIC	0	组态
UA_ACTION_INSERT	1	插入
UA_ACTION_UPDATE	2	同步/覆盖
UA_ACTION_DELETE	3	删除

移动值

UA_MOVE_FIRST	0	移动到第一条数据记录
UA_MOVE_NEXT	1	移动到下一条数据记录
UA_MOVE_PREVIOUS	-1	移动到上一条数据记录
UA_MOVE_LAST	0x80000000	移动到最后一条数据记录

导入/导出文件类型

UA_FILETYPE_DEFAULT	0	CSV 为默认文件类型
UA_FILETYPE_CSV	1	（逗号分隔值）文件类型

3.9.2 结构

3.9.2.1 uaCONFIGARCHIVE

声明

```
typedef struct {
    LONG    lArchiveId;
    LONG    lPosition;
    CHAR    szName[UA_MAXLEN_NAME+1];
    CHAR    szAlias[UA_MAXLEN_ALIAS+1];
    LONG    lType;
    LONG    lNumRecs;
    LONG    lCommType;
    CHAR    szPLCID[UA_MAXLEN_PLCID+1];
    CHAR    szDMVarName[UA_MAXLEN_DMVARNAME+1];
    CHAR    szIDVar[UA_MAXLEN_DMVARNAME+1];
    CHAR    szJobVar[UA_MAXLEN_DMVARNAME+1];
    CHAR    szFieldVar[UA_MAXLEN_DMVARNAME+1];
    CHAR    szValueVar[UA_MAXLEN_DMVARNAME+1];
    DWORD   dwReadRight;
    DWORD   dwWriteRight;
    DWORD   dwFlags;
    } UACONFIGARCHIVE;
```

成员

lArchiveId

用户归档的唯一 ID

lPosition

用户归档的位置

szName

归档名称最多 20 个字符

szAlias

别名最多 50 个字符

lType

UA_ARCHIVETYPE_UNLIMITED	“无限制”归档类型
UA_ARCHIVETYPE_LIMITED	“有限”归档类型

lNumRecs

最大数据记录数

lCommType

UA_COMMTYPE_NONE	无通信
UA_COMMTYPE_RAW	与原始数据通信
UA_COMMTYPE_DIRECT	与 WinCC 变量通信

szPLCID

原始数据变量的 PLCID

szDMVarName

原始数据变量名称

szIDVar

控制变量的 ID

szJobVar

控制变量“Job”

szFieldVar

控制变量“Field”

szValueVar

控制变量“Value”

dwReadRight

读访问权限

dwWriteRight

写访问权限

dwFlags

UA_ARCHIVEFLAG_ACCESS	“上次访问”标记
UA_ARCHIVEFLAG_USER	“上个用户”标记

3.9.2.2 uaCONFIGFIELD

构造“uaCONFIGFIELDA”

```
typedef struct {
    LONG    lArchiveId
    LONG    lFieldId;
    LONG    lPosition;
    CHAR    szName[UA_MAXLEN_NAME+1];
    CHAR    szAlias[UA_MAXLEN_ALIAS+1];
    LONG    lType;
    LONG    lLength;
    LONG    lPrecision;
    CHAR    szMinValue[UA_MAXLEN_VALUE+1];
    CHAR    szMaxValue[UA_MAXLEN_VALUE+1];
    CHAR    szStartValue[UA_MAXLEN_VALUE+1];
    CHAR    szDMVarName[UA_MAXLEN_DMVARNAME+1];
    DWORD   dwReadRight;
    DWORD   dwWriteRight;
    DWORD   dwFlags;
    } UACONFIGARCHIVE;
```

成员

lArchiveId

用户归档的唯一 ID

lFieldID

数据字段的唯一 ID

lPosition

用户归档的位置

szName

归档名称最多 20 个字符

szAlias

别名最多 50 个字符

lType

UA_ARCHIVETYPE_UNLIMITED	“无限制”归档类型
UA_ARCHIVETYPE_LIMITED	“有限”归档类型

lLength

数据字段类型为 STRING 时的最大字符数；否则，不使用

lPrecision

预留为内部使用

szMinValue

最小值。不适用于 STRING 或 DATE 类型的数据字段。

szMaxValue

最大值。不适用于 STRING 或 DATE 类型的数据字段。

szStartValue

起始值

szDMVarName

WinCC 变量的名称。用于支持通过 WinCC 变量进行通信的配方。

dwReadRight

读访问权限

dwWriteRight

写访问权限

dwFlags

UA_ARCHIVEFLAG_ACCESS	“上次访问”标记
UA_ARCHIVEFLAG_USER	“上个用户”标记

3.9.3 常规函数

3.9.3.1 uaGetLastError

说明

许多函数返回布尔值。TRUE 表明处理函数时未出错。如果返回 FALSE 值，可使用 uaGetLastError 函数读取上一次使用的函数出现的错误。

uaGetLastError 始终显示上一次出现的错误。如果准确知道包含错误的函数，则必须评估每次调用函数后的返回值，并在出现错误时调用 uaGetLastError。

带有返回值的函数的示例

```
if ( uaArchiveGetFieldValueLong ( hArchive, Index, &IntValue ) == TRUE )
    printf( "Field Value = %u\n", IntValue );
else
    printf("Error calling uaArchiveGetFieldValueLong: %d / %08lx\n", uaGetLastError(),
uaGetLastHResult());
```

无返回值的函数的示例

```
uaArchiveGetFilter(hArchive, pszFilter, cMaxLen);
    INT nUAError = uaGetLastError (  );
    if ( UA_ERROR_SUCCESS != nUAError)
    {
        printf( "Filter = [%s]\n", pszFilter );
    }
    else
    {
        rintf("Error calling uaArchiveGetFilter: %d, hr=0x%08lX\n", nUAError,
uaGetLastHResult());
    }
INT uaGetLastError()
```

返回值

执行最后一个函数的错误状态。错误常量和预定义条目在 CCUACAPI.H 中确定。

UA_ERROR_SUCCESS	无错误，一切正常。
UA_ERROR_GENERIC	组态函数出错。
UA_ERROR_CONNECT_FAILED	未建立运行系统连接。
UA_ERROR_OPEN_FAILED	无法打开归档。
UA_ERROR_CLOSE_FAILED	无法关闭归档，或者句柄无效。
UA_ERROR_REQUERY_FAILED	无法建立带有诸如新过滤器/新排序的归档复位关系。
UA_ERROR_MOVE_FAILED	无法定位新的“第一个”、“最后一个”、“下一个”或“上一个”。
UA_ERROR_INSERT_FAILED	记录无法插入给定位置。请检查位置。
UA_ERROR_UPDATE_FAILED	无法更新已说明的记录。记录已锁定，或者该位置无记录。
UA_ERROR_DELETE_FAILED	无法删除（所有）记录。
UA_ERROR_IMPORT_FAILED	未能导入归档。请检查归档的格式和大小。
UA_ERROR_EXPORT_FAILED	未能导出用户归档。请检查存储单元的大小。
UA_ERROR_READ_FAILED	未能读取变量。
UA_ERROR_WRITE_FAILED	未执行变量写入操作。
UA_ERROR_GET_FAILED	未能读取归档/字段描述和/或记录值。
UA_ERROR_SET_FAILED	未能写入归档/字段描述和/或记录值。
UA_ERROR_INVALID_NAME	名称无效。
UA_ERROR_INVALID_TYPE	类型无效。
UA_ERROR_INVALID_NUMRECS	记录编号无效。
UA_ERROR_INVALID_COMMTYPE	与所选“无/原始/直接”的链接错误。
UA_ERROR_INVALID_LENGTH	指定的（字符串）长度无效。
UA_ERROR_INVALID_PRECISION	指定的小数点数量无效。
UA_ERROR_NULL_POINTER	指定了零指针。
UA_ERROR_INVALID_POINTER	指定的指针无效。
UA_ERROR_INVALID_HANDLE	指定的句柄无效。
UA_ERROR_INVALID_INDEX	指定的索引无效。
UA_ERROR_SERVER_UNKNOWN	指定使用的服务器未知。

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

3.9.3.2 uaIsActive

说明

确认活动运行系统中无正在使用的配方。

声明

```
BOOL uaIsActive (
    UAHCONNECT hConnect )
```

参数

hConnect

运行系统中的配方句柄。此句柄通过 uaConnect 生成。

返回值

TRUE

活动运行系统中正在使用配方。

FALSE

活动运行系统中未使用的配方。

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaConnect (页 2665)	生成句柄

参见

uaConnect (页 2665)

3.9.3.3 uaUsers

说明

提供所有通过 uaConnect 连接到配方的用户的数量。在这里，需考虑用户通过同样包含在 WinCC 中的脚本等所进行的调用。

声明

```
LONG uaUsers (
    UAHCONNECT hConnect )
```

参数

hConnect

运行系统中的配方句柄。此句柄通过 uaConnect 生成。

返回值

活动连接的数目

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaConnect (页 2665)	生成句柄

参见

uaConnect (页 2665)

3.9.3.4 uaSetLocalEvents

说明

设置本地事件。

声明

```
void uaSetLocalEvents (
    UAHCONNECT hConnect
    BOOL bLocalEvents )
```

参数

hConnect

运行系统中的配方句柄。此句柄通过 uaConnect 生成。

bLocalEvents

本地事件

返回值

无返回值

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaConnect (页 2665)	生成句柄

参见

uaConnect (页 2665)

3.9.4 用于生成连接的函数

3.9.4.1 uaConnect

说明

在运行系统中生成与配方的连接。

声明

```
BOOL uaConnect (
    UAHCONNECT* phConnect )
```

参数

phConnect

指向配方句柄的指针。

返回值

TRUE

与配方的连接已生成

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

注释

uaDisconnect、uaIsActive、uaUsers、uaOpenViews、uaOpenArchives、uaQueryArchive 和 uaQueryArchiveByName 函数需要使用提供的句柄。

相关函数

uaDisconnect (页 2666)	终止连接

参见

uaIsActive (页 2662)

uaUsers (页 2663)

uaSetLocalEvents (页 2664)

uaDisconnect (页 2666)

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

uaOpenArchives (页 2691)

uaOpenViews (页 2724)

函数概览 (页 2647)

3.9.4.2 uaDisconnect

说明

在运行系统中与配方建立连接时，该连接将会终止。

声明

```
BOOL uaDisconnect (
    UAHCONNECT hConnect )
```

参数

hConnect

运行系统中的配方句柄。此句柄通过 uaConnect 生成。

返回值

TRUE

连接终止。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaConnect (页 2665)	生成句柄

参见

uaConnect (页 2665)

函数概览 (页 2647)

3.9.4.3 uaQueryArchive

说明

与配方建立连接。uaQueryArchive 生成句柄 phArchive。

声明

```
BOOL uaQueryArchive (
    UAHCONNECT        hConnect,
    LONG            lArchive,
    UAHARCHIVE*        phArchive )
```

参数

hConnect

运行系统中的配方句柄。此句柄通过 uaConnect 生成。

lArchive

待连接归档的 ID。

phArchive

指向已连接配方句柄的指针。

返回值

TRUE

已建立连接

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

注释

如果只有一个归档索引可用，则可使用组态函数 uaGetArchive 确定该归档的 ID。

相关函数

uaConnect (页 2665)	生成句柄

参见

uaConnect (页 2665)

uaReleaseArchive (页 2674)

uaArchiveClose (页 2677)

uaArchiveDelete (页 2678)

uaArchiveExport (页 2680)

uaArchiveGetID (页 2681)

uaArchiveGetName (页 2682)

uaArchiveGetSort (页 2684)

uaArchiveImport (页 2685)

uaArchiveOpen (页 2687)

uaArchiveUpdate (页 2688)

uaArchiveGetCount (页 2692)

uaArchiveGetFilter (页 2694)

uaArchiveInsert (页 2695)

uaArchiveMoveFirst (页 2696)

uaArchiveMoveLast (页 2697)

uaArchiveMoveNext (页 2698)

uaArchiveMovePrevious (页 2700)

uaArchiveGetFieldLength (页 2702)

uaArchiveGetFields (页 2705)

uaArchiveGetFieldType (页 2706)

uaArchiveGetFieldName (页 2703)
uaArchiveGetFieldValueDate (页 2707)
uaArchiveGetFieldValueDouble (页 2708)
uaArchiveGetFieldValueLong (页 2710)
uaArchiveSetFieldValueDate (页 2713)
uaArchiveSetFieldValueDouble (页 2714)
uaArchiveSetFieldValueLong (页 2715)
uaArchiveSetFieldValueString (页 2717)
uaArchiveRequery (页 2720)
uaArchiveSetFilter (页 2721)
uaArchiveSetSort (页 2722)
uaArchiveReadTagValues (页 2725)
uaArchiveReadTagValuesByName (页 2726)
uaArchiveWriteTagValues (页 2728)
uaArchiveWriteTagValuesByName (页 2730)
函数概览 (页 2647)

3.9.4.4 uaQueryArchiveByName

说明

通过名称与配方建立连接。uaQueryArchiveByName 生成句柄 phArchive。

声明

```
BOOL uaQueryArchiveByName (
    UAHCONNECT        hConnect,
    LPCSTR            pszName,
    UAHARCHIVE*        phArchive )
```

参数

hConnect

运行系统中的配方句柄。此句柄通过 uaConnect 生成。

pszName

配方名称。

phArchive

指向已连接配方句柄的指针。

返回值

TRUE

已建立连接

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

注释

如果只有一个归档索引可用，则可使用组态函数 uaGetArchive 确定该归档的名称。

相关函数

uaConnect (页 2665)	生成句柄

参见

uaConnect (页 2665)

uaReleaseArchive (页 2674)

uaArchiveClose (页 2677)

uaArchiveReadTagValues (页 2725)

uaArchiveReadTagValuesByName (页 2726)

uaArchiveWriteTagValues (页 2728)

uaArchiveWriteTagValuesByName (页 2730)

函数概览 (页 2647)

3.9.4.5 UaQueryConfiguration

说明

生成与配方（用于组态）的连接。

声明

```
BOOL uaQueryConfiguration (
    UAHCONFIG*        phConfig )
```

参数

phConfig

指向接收器句柄的指针。

返回值

TRUE

已建立连接

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

参见

uaReleaseConfiguration (页 2675)

uaGetArchive (页 2689)

uaGetNumArchives (页 2690)

uaGetNumFields (页 2701)

uaGetField (页 2718)

函数概览 (页 2647)

3.9.4.6 uaReleaseArchive

说明

中断与当前连接的配方的连接。

声明

```
BOOL uaReleaseArchive (
    UAHARCHIVE        hArchive )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

连接终止。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

注释

要成功终止连接，必须将 hArchive 句柄设置为 NULL。如果所使用的句柄不再有效，系统就会生成错误消息 UA_ERROR_INVALID_HANDLE。这将避免不必要的内存过载。

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.4.7 uaReleaseConfiguration

说明

在组态与配方的连接之后将其终止。

声明

```
BOOL uaReleaseConfiguration (
    UAHCONFIG        hConfig,
    BOOL            bSave )
```

参数

hConfig

配方的句柄。此句柄通过 uaQueryConfiguration 生成。

bSave

确认在终止连接之前保存对组态的更改。

TRUE	保存更改
FALSE	取消更改

说明

如果活动 WinCC Runtime 中未使用配方，那么应该只使用 bSave = TRUE！可使用 uaIsActive 函数检查运行系统是否为激活状态。

返回值

TRUE

连接终止。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

UaQueryConfiguration (页 2673)	生成句柄

参见

UaQueryConfiguration (页 2673)

函数概览 (页 2647)

3.9.5 用于配方处理的函数

3.9.5.1 uaArchiveClose

说明

关闭打开的配方。

声明

```
BOOL uaArchiveClose (
    UAHARCHIVE        hArchive )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

配方已关闭。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄
uaArchiveOpen (页 2687)	打开配方

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

uaArchiveOpen (页 2687)

函数概览 (页 2647)

3.9.5.2 uaArchiveDelete

说明

从配方中删除数据。已组态配方保持选中状态。

声明

```
BOOL uaArchiveDelete (
    UAHARCHIVE        hArchive,
    LPCSTR            pszWhere )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

pszWhere

含有待删除数据条目的 SQL 选择的字符串。此字符串对应于 SQL 指令 DELETE FROM <archive> WHERE pszWhere。

说明

如果 pszWhere 为空，将删除整个用户归档。

返回值

TRUE

数据已删除。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.5.3 uaArchiveExport

说明

将数据从配方导出到 CSV 文件。

声明

```
BOOL uaArchiveExport (
    UAHARCHIVE        hArchive,
    LPCSTR            pszDestination,
    LONG              lType,
    LONG              lOptions )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

pszDestination

目标文件名。在客户端上调用该函数时，指定的路径表示服务器。

lType

目标文件数据格式。两种格式可用：

UA_FILETYPE_DEFAULT = 0	预设 CSV 文件格式
UA_FILETYPE_CSV = 1	CSV 文件格式

lOptions

该参数为将来开发预留，必须预设为 0。

返回值

TRUE

数据已导出。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄
uaArchiveImport (页 2685)	导入数据

参见

uaArchiveImport (页 2685)

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.5.4 uaArchiveGetID

说明

读取配方 ID

返回值：

用户归档的“ID”

声明

```
LONG uaArchiveGetID (
    UAHARCHIVE        hArchive )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

配方 ID

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.5.5 uaArchiveGetName

说明

读取配方名称。

声明

```
VOID uaArchiveGetName (
    UAHARCHIVE        hArchive,
    LPSTR             pszName,
    LONG             cMaxLen )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

pszName

指向配方名称缓冲区的指针。

cMaxLen

最大长度

返回值

无返回值

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

示例

```
char tank [40];
uaArchiveGetName( hArchive, tank, 39 );
```

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.5.6 uaArchiveGetSort

说明

读取配方排序参数。

声明

```
VOID uaArchiveGetSort (
    UAHARCHIVE        hArchive,
    LPSTR             pszSort,
    LONG             cMaxLen )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

pszSort

SQL 语句形式的排序参数。

cMaxLen

最大长度

返回值

无返回值

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.5.7 uaArchiveImport

说明

将数据从 CSV 文件导入配方。配方的结构必须与导入的 CSV 归档结构相同。

声明

```
BOOL uaArchiveImport (
    UAHARCHIVE        hArchive,
    LPCSTR            pszSource,
    LONG            lType,
    LONG            lOptions )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

pszSource

使用待导入数据的文件名。

lType

源文件数据格式。两种格式可用：

UA_FILETYPE_DEFAULT = 0	预设 CSV 文件格式
UA_FILETYPE_CSV = 1	CSV 文件格式

lOptions

该参数为将来开发预留，必须预设为 0。

返回值

TRUE

数据已导入。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄
uaArchiveExport (页 2680)	导出文件。

参见

uaArchiveExport (页 2680)

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.5.8 uaArchiveOpen

说明

打开配方。在配方中执行读写操作均需调用 uaArchiveOpen。例如调用函数 uaArchiveMoveNext、uaArchiveDelete 或 uaArchiveUpdate。

声明

```
BOOL uaArchiveOpen (
    UAHARCHIVE        hArchive )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

配方已打开。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄
uaArchiveClose (页 2677)	关闭配方

参见

uaArchiveClose (页 2677)

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.5.9 uaArchiveUpdate

说明

更新打开的配方。将在数据库中应用所有已更改的配方数据。配方结构保持不变。

声明

```
BOOL uaArchiveUpdate (
U    AHARCHIVE hArchive )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

配方数据已更新。

FALSE

如果违反一致性，则显示错误消息 Update_failed = 106。例如，如果某一字段具有“字段必须包含值”(Field must contain a value) 属性，但字段中却没有值，就会出现这种违反一致性的情况。

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.5.10 uaGetArchive

说明

读取配方组态。

声明

```
BOOL uaGetArchive (
    UAHCONFIG            hConfig,
    Long                 lArchive,
    UACONFIGARCHIVE*     pArchive )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryConfiguration 生成。

lArchive

配方索引。值：0... uaGetNumArchives() -1

pArchive

指向组态数据缓冲区的指针。

返回值

TRUE

成功访问。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

UaQueryConfiguration (页 2673)	生成句柄

参见

UaQueryConfiguration (页 2673)

函数概览 (页 2647)

uaCONFIGARCHIVE (页 2656)

3.9.5.11 uaGetNumArchives

说明

读取当前已组态的配方的数目。

声明

```
LONG uaGetNumArchives (
    UAHCONFIG        hConfig )
```

参数

hConfig

配方的句柄。此句柄通过 uaQueryConfiguration 生成。

返回值

已组态的配方数。如果出现错误，则输出 -1。

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

UaQueryConfiguration (页 2673)	生成句柄

参见

UaQueryConfiguration (页 2673)

函数概览 (页 2647)

3.9.5.12 uaOpenArchives

说明

传输已打开的运行系统中的配方数。

声明

```
LONG uaOpenArchives (
    UAHCONNECT hConnect )
```

参数

hArchive

配方的句柄。此句柄通过 uaConnect 生成。

返回值

已打开的配方数。

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaConnect (页 2665)	生成句柄

参见

uaConnect (页 2665)

函数概览 (页 2647)

3.9.6 用于配方元素处理的函数

3.9.6.1 uaArchiveGetCount

说明

读取数据记录的数量。

返回值：

数据记录的数量。如果为“0”，则说明归档为空或出现错误。需要使用“uaGetLastError”进行查询。

声明

```
LONG uaArchiveGetCount(
    UAHARCHIVE        hArchive,
    LONG*            plCount )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

plCount

指向用于储存记录数量的变量的指针。

返回值

数据记录的数量。如果为 0，则说明归档为空或出现错误。建议使用 uaGetLastError 进行查询。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.6.2 uaArchiveGetFilter

说明

读取当前数据记录的选择标准。

声明

```
VOID uaArchiveGetFilter (
    UAHARCHIVE        hArchive,
    LPSTR             pszFilter,
    LONG              cMaxLen )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

pszFilter

SQL 语句形式的选择参数。

cMaxLen

最大长度

返回值

无返回值

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.6.3 uaArchiveInsert

说明

将本地数据记录缓冲区插入到配方中。为了在新数据文件中说明有意义的记录，调用 uaArchiveInsert 前必须使用 uaArchiveSetFieldValue... 函数写入本地数据记录缓冲区的字段。

如果归档没有 ID 或已写成“0”，则必须使用 uaArchiveSetFieldValueLong 函数写入“ID”字段。

声明

```
BOOL uaArchiveInsert (
    UAHARCHIVE        hArchive )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

数据记录已插入。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.6.4 uaArchiveMoveFirst

说明

跳转到第一条数据记录。

声明

```
BOOL uaArchiveMoveFirst (
    UAHARCHIVE        hArchive )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

成功跳转。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.6.5 uaArchiveMoveLast

说明

跳转到最后一条数据记录。

声明

```
BOOL uaArchiveMoveLast (
    UAHARCHIVE        hArchive )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

成功跳转。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.6.6 uaArchiveMoveNext

说明

跳转到下一条数据记录。

声明

```
BOOL uaArchiveMoveNext (
    UAHARCHIVE        hArchive )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

成功跳转。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.6.7 uaArchiveMovePrevious

说明

跳转到前一条数据记录。

声明

```
BOOL uaArchiveMovePrevious (
    UAHARCHIVE        hArchive )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

成功跳转。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.6.8 uaGetNumFields

说明

提供已组态域的数目。不包括"ID"、“上个用户”和“上次访问”域。在组态调用中，使用"0 ... uaGetNumFields() -1"说明索引。

声明

```
LONG uaGetNumFields (
    UAHCONFIG        honfig,
    long             lArchive )
```

参数

hConfig

配方的句柄。此句柄通过 uaQueryConfiguration 生成。

lArchive

配方索引。值：0... uaGetNumArchives() -1

返回值

已组态字段的数量。如果出现错误，则输出 -1。

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

UaQueryConfiguration (页 2673)	生成句柄

参见

UaQueryConfiguration (页 2673)

函数概览 (页 2647)

3.9.7 用于字段处理的函数

3.9.7.1 uaArchiveGetFieldLength

说明

读取当前数据记录中域的长度。

声明

```
LONG uaArchiveGetFieldLength(
    UAHARCHIVE      hArchive,
    LONG            lField )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

lField

字段编号，其中 iField = 1 寻址第一个已组态字段。lField = 0 寻址"ID"字段。

返回值

当前域的长度。

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.7.2 uaArchiveGetFieldName

说明

读取当前数据记录中域的名称。

声明

```
VOID uaArchiveGetFieldName (
    UAHARCHIVE    hArchive,
    LONG          lField,
    LPCSTR        pszName,
    LONG          cMaxLen )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

lField

字段编号，其中 iField = 1 寻址第一个已组态字段。lField = 0 寻址"ID"字段。

pszName

域名称

cMaxLen

最大长度

返回值

TRUE

名称已读取。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.7.3 uaArchiveGetFields

说明

读取已组态数据字段数。包括"ID"、“上个用户”和“上次访问”字段。在运行系统调用中，使用 1 ... N 表示已组态字段的索引。"ID"字段索引为"0"。“上个用户”和“上次访问”字段附加在已组态字段末尾。

声明

```
LONG uaArchiveGetFields (
    UAHARCHIVE        hArchive )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

已组态字段的数量。

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.7.4 uaArchiveGetFieldType

说明

读取当前数据记录中域的类型。

声明

```
LONG uaArchiveGetFieldType (
    UAHARCHIVE      hArchive,
    LONG            lField )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

lField

字段编号，其中 iField = 1 寻址第一个已组态字段。lField = 0 寻址"ID"字段。

返回值

当前域的类型。域类型的符号定义是：

UA_FIELDTYPE_INTEGER	
UA_FIELDTYPE_FLOAT	
UA_FIELDTYPE_DOUBLE	
UA_FIELDTYPE_STRING	
UA_FIELDTYPE_DATETIME	

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.7.5 uaArchiveGetFieldValueDate

说明

读取当前数据记录中域的日期和时间。

声明

```
BOOL uaArchiveGetFieldValueDate (
    UAHARCHIVE        hArchive,
    LONG            lField,
    LPSYSTEMTIME    pstDateTime )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

lField

字段编号，其中 iField = 1 寻址第一个已组态字段。lField = 0 寻址"ID"字段。

pstDateTime

指向 SYSTEMTIME 类型变量的指针。

返回值

TRUE

更改日期和时间。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.7.6 uaArchiveGetFieldValueDouble

说明

从当前数据记录中的字段读取 Double 数据类型的值。

声明

```
BOOL uaArchiveGetFieldValueDouble (
    UAHARCHIVE        hArchive,
    LONG            lField,
    double*            pdValue )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

lField

字段编号，其中 iField = 1 寻址第一个已组态字段。lField = 0 寻址“ID”字段。

pdValue

指向字段值的变量的指针。

返回值

TRUE

值已读取。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.7.7 uaArchiveGetFieldValueLong

说明

从当前数据记录中的字段读取 Long 数据类型的值。

声明

```
BOOL uaArchiveGetFieldValueLong (
    UAHARCHIVE        hArchive,
    LONG             lField,
    LONG*             pdValue )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

lField

字段编号，其中 iField = 1 寻址第一个已组态字段。lField = 0 寻址"ID"字段。

pdValue

指向字段值的变量的指针。

返回值

TRUE

值已读取。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.7.8 uaArchiveGetFieldValueString

说明

从当前数据记录中的字段读取 String 数据类型的值。

声明

```
BOOL uaArchiveGetFieldValueString (
    UAHARCHIVE        hArchive,
    LONG             lField,
    LPSTR             pszString,
    LONG             cMaxLen )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

lField

字段编号，其中 iField = 1 寻址第一个已组态字段。lField = 0 寻址"ID"字段。

pszString

指向字段值的变量的指针。

pdValue

字符串的最大长度。

返回值

TRUE

值已读取。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.7.9 uaArchiveSetFieldValueDate

说明

将日期和时间写入当前数据记录的域中。

声明

```
BOOL uaArchiveSetFieldValueDate (
    UAHARCHIVE        hArchive,
    LONG            lField,
    LPSYSTEMTIME    pstDateTime )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

lField

字段编号，其中 iField = 1 寻址第一个已组态字段。lField = 0 寻址"ID"字段。

pstDateTime

日期和时间。

返回值

TRUE

值已写入。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.7.10 uaArchiveSetFieldValueDouble

说明

从当前数据记录中的字段写入 Double 数据类型的值。

声明

```
BOOL uaArchiveSetFieldValueDouble (
    UAHARCHIVE        hArchive,
    LONG             lField,
    double             dValue )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

lField

字段编号，其中 iField = 1 寻址第一个已组态字段。lField = 0 寻址"ID"字段。

dValue

要写入的值。

返回值

TRUE

值已写入。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.7.11 uaArchiveSetFieldValueLong

说明

从当前数据记录中的字段写入 Long 数据类型的值。

声明

```
BOOL uaArchiveSetFieldValueLong (
    UAHARCHIVE hArchive,
    LONG lField,
    LONG dValue )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

lField

字段编号，其中 iField = 1 寻址第一个已组态字段。lField = 0 寻址"ID"字段。

dValue

要写入的值。

返回值

TRUE

值已写入。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.7.12 uaArchiveSetFieldValueString

说明

从当前数据记录中的字段写入 String 数据类型的值。

声明

```
BOOL uaArchiveSetFieldValueString (
    UAHARCHIVE        hArchive,
    LONG            lField,
    LPCSTR            pszString )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

lField

字段编号，其中 iField = 1 寻址第一个已组态字段。lField = 0 寻址"ID"字段。

pszString

要写入的字符串。

返回值

TRUE

值已写入。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.7.13 uaGetField

说明

读取域组态。

声明

```
BOOL uaGetField (
    UAHCONFIG        hConfig,
    Long            lArchive,
    Long            lField,
    UACONFIGFIELD*    pField )
```

参数

hConfig

配方的句柄。此句柄通过 uaQueryConfiguration 生成。

lArchive

配方索引。值：0... uaGetNumArchives() -1

lField

字段编号，其中 iField = 1 寻址第一个已组态字段。lField = 0 寻址“ID”字段。

pField

指向组态数据缓冲区的指针。

返回值

TRUE

组态已读取。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

UaQueryConfiguration (页 2673)	生成句柄

参见

UaQueryConfiguration (页 2673)

函数概览 (页 2647)

uaCONFIGFIELD (页 2658)

3.9.8 过滤和排序函数。

3.9.8.1 uaArchiveRequery

说明

调用 uaArchiveSetFilter 和 uaArchiveSetSort 后，必须使用 uaArchiveRequery 重新加载配方。

说明

可使用 uaArchiveSetSort 和 uaArchiveSetFilter 函数，无需使用 uaArchiveOpen 打开配方。这种情况下不能使用 uaArchiveRequery 进行调用。

声明

```
BOOL uaArchiveRequery(
    UAHARCHIVE        hArchive )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

返回值

TRUE

配方已重新加载。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

注释

如果已将运行系统中的值插入配方视图，还可使用 uaArchiveRequery 进行调用。

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.8.2 uaArchiveSetFilter

说明

设置配方选择参数。还可使用该函数，无需使用 uaArchiveOpen 打开配方。

如果已使用 uaArchiveOpen 打开用户归档，过滤后必须使用 uaArchiveRequery 重新加载配方。

声明

```
VOID uaArchiveSetFilter (
    UAHARCHIVE       hArchive,
    LPSTR            pszFilter )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

pszFilter

SQL 参数形式的选择参数。

返回值

TRUE

选择参数已更改。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.8.3 uaArchiveSetSort

说明

设置配方排序。还可使用该函数，无需使用 uaArchiveOpen 打开配方。

如果已使用 uaArchiveOpen 打开配方，则排序后必须使用 uaArchiveRequery 重新加载配方。

声明

```
BOOL uaArchiveSetSort (
    UAHARCHIVE        hArchive,
    LPSTR            pszSort )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

pszSort

SQL 语句形式的排序参数。

返回值

TRUE

排序成功。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

函数概览 (页 2647)

3.9.9 用于配方视图处理的函数

3.9.9.1 uaOpenViews

说明

给出打开的运行系统视图中的配方数。

声明

```
LONG uaOpenViews (
    UAHCONNECT        hConnect )
```

参数

hArchive

配方的句柄。此句柄通过 uaConnect 生成。

返回值

当前打开的视图数。

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaConnect (页 2665)	生成句柄

参见

uaConnect (页 2665)

函数概览 (页 2647)

3.9.10 用于变量处理的函数

3.9.10.1 uaArchiveReadTagValues

说明

从域变量中读取当前值。

声明

```
BOOL uaArchiveReadTagValues (
    UAHARCHIVE    hArchive,
    LONG*         pnFields,
    LONG         cFields,
    LONG         lOptions )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

pnFields

该参数为将来开发预留，必须预设为 0。

cFields

该参数为将来开发预留，必须预设为 0。

lOptions

该参数为将来开发预留，必须预设为 0。

返回值

TRUE

数据已读取。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaArchiveWriteTagValues (页 2728)	写入值
uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

uaArchiveWriteTagValues (页 2728)

函数概览 (页 2647)

3.9.10.2 uaArchiveReadTagValuesByName

说明

从变量读取当前值。

声明

```
BOOL uaArchiveReadTagValuesByName (
    UAHARCHIVE        hArchive,
    LPCSTR            pszFields,
    LONG            lOptions )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

pszFields

该参数为将来开发预留，必须预设为 NULL。

lOptions

该参数为将来开发预留，必须预设为 0。

返回值

TRUE

数据已读取。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaArchiveWriteTagValuesByName (页 2730)	读取值
uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

uaArchiveWriteTagValuesByName (页 2730)

函数概览 (页 2647)

3.9.10.3 uaArchiveWriteTagValues

说明

将当前数据记录的值写入变量中。

声明

```
BOOL uaArchiveWriteTagValues (
    UAHARCHIVE        hArchive,
    LONG*             pnFields,
    LONG              cFields,
    LONG              lOptions )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

pnFields

该参数为将来开发预留，必须预设为 0。

cFields

该参数为将来开发预留，必须预设为 0。

lOptions

该参数为将来开发预留，必须预设为 0。

返回值

TRUE

数据已写入。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaArchiveReadTagValues (页 2725)	读取值
uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

uaArchiveReadTagValues (页 2725)

函数概览 (页 2647)

3.9.10.4 uaArchiveWriteTagValuesByName

说明

将当前数据记录的值写入变量中。

声明

```
BOOL uaArchiveWriteTagValuesByName (
    UAHARCHIVE        hArchive,
    LPCSTR            pszFields,
    LONG              lOptions )
```

参数

hArchive

配方的句柄。此句柄通过 uaQueryArchive 或 uaQueryArchiveByName 生成。

pszFields

该参数为将来开发预留，必须预设为 NULL。

lOptions

该参数为将来开发预留，必须预设为 0。

返回值

TRUE

数据已写入。

FALSE

错误

所需文件

ccuacapi.h

ccuacapi.lib

ccuacapi.dll

相关函数

uaArchiveReadTagValuesByName (页 2726)	读取值
uaQueryArchive (页 2668)	生成句柄
uaQueryArchiveByName (页 2670)	生成句柄

参见

uaQueryArchive (页 2668)

uaQueryArchiveByName (页 2670)

uaArchiveReadTagValuesByName (页 2726)

函数概览 (页 2647)

3.10 报警函数

3.10.1 基本知识

3.10.1.1 函数概览

概述

MSG_SERVICE_NOTIFY_PROC_PLUS (页 2798)	发送和接收服务（回调）
MSRTActivateMProtPlus (页 2805)	暂停或继续协议服务
MSRTCheckWinFilterPlus (页 2852)	检查报警过滤条件
MSRTCreateMsgInstancePlus (页 2810)	创建报警
MSRTCreateMsgInstanceWithCommentPlus (页 2808)	创建报警
MSRTDialogMsgLockPlus (页 2814)	设置报警禁用参数
MSRTEnumArchivDataPlus (页 2870)	列出日志报警
MSRTEnumBackupListPlus (页 2867)	列出导出列表条目

MSRTEnumGroupMsgPlus (页 2845)	列出报警组单个报警
MSRTExportPlus (页 2869)	从顺序日志导出报警
MSRTGetClassInfoPlus (页 2819)	查询报警类别信息
MSRTGetCommentInstancePlus (页 2860)	查询注释文本
MSRTGetFilterDataPlus (页 2850)	查询报警过滤条件
MSRTGetInfotextPlus (页 2864)	查询信息文本
MSRTGetLastMsgWithCommentPlus (页 2821)	查询最后一个日志报警
MSRTGetMsgActualPlus (页 2825)	确定报警数
MSRTGetMsgCSDataPlus (页 2827)	查询报警组态数据
MSRTGetMsgPriorityPlus (页 2836)	查询报警优先级
MSRTGetMsgQuitPlus (页 2838)	查询需要确认的报警
MSRTGetSelectedMsgPlus (页 2839)	确定报警编号
MSRTLoopInAlarmPlus (页 2842)	报警循环
MSRTMsgWinCommandPlus (页 2857)	在报警窗口中执行控制函数
MSRTPrintMProtPlus (页 2806)	打印未决报警的报警序列报表
MSRTQuitGroupPlus (页 2849)	确认报警组
MSRTQuitHornPlus (页 2797)	确认听觉信号设备
MSRTResetMsgPlus (页 2843)	确认报警
MSRTSetCommentInstancePlus (页 2861)	指定注释文本
MSRTSetInfotextPlus (页 2865)	指定信息文本
MSRTSetMsgFilterPlus (页 2854)	设置报警过滤器
MSRTSetMsgWinFilterPlus (页 2856)	为报警窗口设置报警过滤器
MSRTStartMsgServicePlus (页 2791)	启动收发服务（回调）
MSRTStopMsgServicePlus (页 2795)	停止收发服务

参见

MSRTEnumMsgRTDataPlus (页 2818)

MSRTLockGroupPlus (页 2847)

3.10.1.2 结构概览

概述

MSG_BACKUP_STRUCT_PLUS (页 2753)	导出列表
MSG_CLASS_STRUCT_PLUS (页 2755)	报警类别
MSG_COMMENT_INSTANCE_STRUCT_PLUS (页 2757)	注释
MSG_CSDATA_STRUCT_PLUS (页 2759)	单个报警
MSG_FILTER_STRUCT_PLUS (页 2765)	报警过滤器
MSG_INFOTEXT_STRUCT_PLUS (页 2772)	信息文本
MSG_RTCREATE_STRUCT_PLUS (页 2774)	在运行系统中创建报警
MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (页 2780)	单个报警
MSG_RTDATA_INSTANCE_STRUCT_PLUS (页 2776)	单个报警
MSG_RTGROUPENUM_STRUCT_PLUS (页 2784)	报警组的单个报警
MSG_RTGROUPSET_STRUCT_PLUS (页 2786)	报警组数据
MSG_RTLOCK_STRUCT_PLUS (页 2787)	报警锁定
MSG_TEXTVAL256_STRUCT_PLUS (页 2788)	关联文本值

3.10.1.3 错误消息

概述

CMN_ERROR 错误结构中的 API 函数可返回以下错误消息：

报警记录 CS

MSG_ERR_CS_CONNECT	0x00001001	未连接到 Alarm Logging CS 或项目。
MSG_ERR_CS_PROJECT	0x00001002	项目不正确或无效
MSG_ERR_CS_PROJECTNAME	0x00001003	项目名称不正确或无效
MSG_ERR_CS_PARAM	0x00001004	参数不正确；dwError2 包含不正确参数的编号。
MSG_ERR_CS_NODATA	0x00001005	未找到数据
MSG_ERR_CS_STRUCTFIELD	0x00001006	结构字段出错；dwError2 包含值不正确的结构字段的编号。
MSG_ERR_CS_DBCONNECT	0x00001011	DBConnect 不可用
MSG_ERR_CS_DBCDELETE	0x00001012	删除时出错（无数据？）
MSG_ERR_CS_DBOPEN	0x00001013	无法打开含有单个报警的表
MSG_ERR_CS_SOURCES	0x00001014	内存不足？
MSG_ERR_CS_CREATOR_ID	0x00001015	CreatorID 为 WinCC-CreatorID！
MSG_ERR_CS_NOTSUPPORTED	0x00001016	不支持该函数
MSG_ERR_UNKNOWN_ERROR	0x00001020	发生未指明错误（WinCC V6.0 及更高版本）
MSG_ERR_API_TERMINATED	0x00001021	处理被过早取消（WinCC V6.0 及更高版本）
MSG_ERR_NO_MEMORY	0x00001022	主内存不足（WinCC V6.0 及更高版本）
MSG_ERR_DB_FAIL	0x00001023	数据库不一致（WinCC V6.0 及更高版本）

MSG_ERR_I_WRITE	0x00001024	无法将数据输入到数据库中。dwError2 和 dwError3 包含详细信息。（WinCC V6.0 及更高版本）
MSG_ERR_CS_EXIST	0x00001025	元素已存在（WinCC V6.0 及更高版本）
MSG_ERR_CS_NO_EXIST	0x00001026	元素不存在（WinCC V6.0 及更高版本）
MSG_ERR_HANDLE	0x00000090	句柄无效

报警记录 RT

MSG_ERR_RT_NOTCONNECTED	0x00000001	未与 Alarm Logging RT 建立连接
MSG_ERR_RT_NOINIT	0x00000002	Alarm Logging RT 未初始化
MSG_ERR_RT_SERVICEMAX	0x00000003	达到最大服务数
MSG_ERR_MSG_NOEXIST	0x00000010	报警不存在
MSG_ERR_MSG_NOTFOUND	0x00000011	未找到报警
MSG_ERR_MSG_NOQUIT	0x00000012	报警无需确认
MSG_ERR_MSG_ALREADYQUIT	0x00000013	报警已确认
MSG_ERR_MSG_NOCLASS	0x00000014	类别信息不可用
MSG_ERR_MSG_STATE	0x00000015	消息状态未知
MSG_ERR_MSG_LOCKED	0x00000016	报警已锁定
MSG_ERR_MSG_DATETIME	0x00000017	报警日期或报警时间无效

MSG_ERR_WIN_NAMENOTFOUND	0x00000020	未找到模板名称
MSG_ERR_WIN_CREATE	0x00000021	创建报警窗口时出错
MSG_ERR_WIN_DATA	0x00000022	报警窗口数据无效
MSG_ERR_VAR_BITEXCEEDS_	0x00000030	变量位无效
MSG_ERR_VAR_NOCHANGE_	0x00000031	无变量更改
MSG_ERR_VAR_PROCESS_USED_	0x00000032	事件变量已分配
MSG_ERR_VAR_QUIT_USED_	0x00000033	确认变量已分配
MSG_ERR_VAR_STATE_USED_	0x00000034	状态变量已分配
MSG_ERR_VAR_VARIANTCONV_	0x00000035	变量转换期间出错
MSG_ERR_VAR_USED_TYPDIFF_	0x00000036	事件变量已分配，但类型不同（确认、创建...）
MSG_ERR_HANDLE	0x00000090	句柄无效
MSG_ERR_FILTER	0x00000091	过滤条件无效
MSG_ERR_API_PARAM	0x00000095	参数无效
MSG_ERR_API_SERVICE	0x00000099	dwServiceID 无效
MSG_ERR_API_EXCEPTION	0x0000009a	代码出现异常情况
MSG_ERR_API_DATETIME	0x0000009b	日期/时间戳无效
MSG_ERR_API_SERVICEMAX	0x0000009c	达到最大服务数

MSG_ERR_API_NODATA	0x0000009d	无数据
MSG_ERR_PROT_ID	0x00000111	协议标识符无效
MSG_ERR_PROT_BLOCKS	0x00000110	协议中无块
MSG_ERR_PROT_ACTIVE	0x00000112	协议服务已激活
MSG_ERR_NODEFAULTSERVER	0x00000401	未组态默认服务器
MSG_ERR_NOLOCALSERVER	0x00000402	无可用本地服务器
MSG_ERR_NOSERVER	0x00000403	未组态默认服务器，且无可用本地服务器
MSG_ERR_NOMC	0x00000404	无多客户端项目（此处未使用）
MSG_ERR_NOMCDEFAULTSERVER	0x00000405	无多客户端项目并指定 “@default”（此处未使用）

3.10.1.4 常数

常规规范

MSG_MAX_CLASS	（值： 18)	报警类别的数目
MSG_MAX_CLASSTYPE	（值： 16)	报警类别中的报警类型数
MSG_MAX_PVALUE	（值： 10)	过程值数
MSG_MAX_TB	（值： 10)	用户文本数
MSG_MAX_HITLIST	（值： 7)	统计列表元素个数
MSG_MAX_TB_CONTENT	（值： 255)	每用户文本字符数
MSG_MAX_TEXTLEN	（值： 64)	所有文本字符数
MSG_MAX_TEXTVALUE	（值： 32)	关联文本值中的字符数
MSG_MAX_TEXTVALUE256	（值： 256)	关联文本值中的字符数
MSG_MAX_STATE	（值： 4)	状态文本数
MSG_MAX_LOCKITEMS	（值： 16)	锁定报警数

MSG_MAX_LEVEL	（值： 6)	组报警节点数
MSG_MAX_MSG_COUNT	（值： 150000) V6.2 及更高版本： （值： 50000)	项目中的最大单个报警数
MSG_MAX_GROUPITEMS	（值： 1000)	含枚举的组的报警数
MSG_MAX_USERNAME	（值： 16)	用户组名称的最大长度
MSG_MAX_APPLNAME	（值： 32)	应用程序名称的最大长度
MSG_MAX_INSTANCE	（值： 512)	实例名称
MSG_MAX_ARCHIV_QUEUE	（值： 1500)	最大可缓冲日志条目数

报警状态

MSG_STATE_COME	（值： 0x00000001）	传入报警
MSG_STATE_GO	（值： 0x00000002）	离去报警
MSG_STATE_QUIT	（值： 0x00000003）	报警已确认
MSG_STATE_LOCK	（值： 0x00000004）	报警已锁定
MSG_STATE_UNLOCK	（值： 0x00000005）	报警已释放
MSG_STATE_HIDE	（值： 0x0000000A）	报警已隐藏
MSG_STATE_UNHIDE	（值： 0x0000000B）	报警已显示
MSG_STATE_HIDE_MANUAL	（值： 0x0000000C）	报警已手动隐藏
MSG_STATE_UNHIDE_MANUAL	（值： 0x0000000D）	报警已手动显示
MSG_STATE_QUIT_SYSTEM	（值： 0x00000010）	系统确认的报警
MSG_STATE_QUIT_EMERGENCY	（值： 0x00000011）	紧急确认
MSG_STATE_QUIT_HORN	（值： 0x00000012）	报警器确认
MSG_STATE_COMEGO	（值： 0x00000013）	到达与离去报警，只显示在报警列表中
MSG_STATE_RESET	（值： 0x00020000）	报警复位位
MSG_STATE_SUMTIME	（值： 0x00040000）	夏令时激活位
MSG_STATE_INSTANCE	（值： 0x00080000）	实例报警位（第 n 个报警）

MSG_STATE_TIMEINVALID	（值： 0x00100000）	无效日期/时间戳位 或运算 （MSG_STATE_COME、 MSG_STATE_GO....）
MSG_STATE_HIDDEN	（值： 0x02000000）	当前隐藏报警位，可用于 V6.2 及更高版本

数据库表中的报警标志

MSG_FLAG_SUMTIME	（值： 0x00000001）	夏令时激活
MSG_FLAG_COMMENT	（值： 0x00000002）	报警具有注释
MSG_FLAG_ARCHIV	（值： 0x00000004）	日志
MSG_FLAG_PROTOCOL	（值： 0x00000008）	报告
MSG_FLAG_TEXTVALUES	（值： 0x00000010）	报警具有相关联的文本值
MSG_FLAG_TIMEINVALID	（值： 0x00000020）	无效日期/时间戳位
MSG_FLAG_INSTANCE	（值： 0x00000040）	实例报警标记
MSG_FLAG_HIDDEN	（值： 0x00000080）	隐藏报警，可用于 V6.2 及更高版本
MSG_FLAG_TEXT10_EXT	（值： 0x00000100）	扩展通道诊断，可用于 V7.0 及更高版本
MSG_FLAG_INSTANCE10	（值： 0x08000000）	过程值为 10 的实例报警，可用于 V5.0 SP2 及更高版本
MSG_FLAG_TEXTREF1	（值： 0x10000000）	文本参考 1，可用于 V5.0 SP2 及更高版本
MSG_FLAG_TEXTREF2	（值： 0x20000000）	文本参考 2，可用于 V5.0 SP2 及更高版本

在线表中的报警标志

只在 m_global.h 中定义，始于 WinCC V6.0 SP4。

MSG_RT_FLAG_COME	（值： 0x00000001）	传入报警
MSG_RT_FLAG_GO	（值： 0x00000002）	离去报警
MSG_RT_FLAG_QUIT_COME	（值： 0x00000004）	到达确认
MSG_RT_FLAG_QUIT_GO	（值： 0x00000008）	离去确认

MSG_RT_FLAG_NOQUIT	（值： 0x00000010）	无确认
MSG_RT_FLAG_QUIT_NOGO	（值： 0x00000020）	TM II 确认
MSG_RT_FLAG_FLASH	（值： 0x00000040）	闪烁
MSG_RT_FLAG_FIRSTFLASH	（值： 0x00000080）	仅允许第一个报警闪烁
MSG_RT_FLAG_NOLIST	（值： 0x00000100）	消息列表中未收到报警
MSG_RT_FLAG_UNIQUE_USER	（值： 0x00000200）	带唯一用户名的报警
MSG_RT_FLAG_ONL_COM_COM	（值： 0x00000400）	只在报警列表中显示到达报警的注释
MSG_RT_FLAG_HIDDEN	（值： 0x00000800）	报警已隐藏
MSG_RT_FLAG_TEXT10_EXT	（值： 0x00001000）	扩展通道诊断，可用于 V7.0 及更高版本
MSG_RT_FLAG_DELETE	（值： 0x00010000）	可以删除报警
MSG_RT_FLAG_SUMTIME	（值： 0x00020000）	夏令时标记
MSG_RT_FLAG_SQUIT	（值： 0x00040000）	报警具有组确认
MSG_RT_FLAG_QUITNR	（值： 0x00080000）	报警编号 - 确认
MSG_RT_FLAG_MSGLIST	（值： 0x00100000）	报警列表中的报警
MSG_RT_FLAG_COMMENT	（值： 0x00200000）	报警具有注释
MSG_RT_FLAG_ARCHIV	（值： 0x00400000）	记录报警
MSG_RT_FLAG_PROTOCOL	（值： 0x00800000）	记录报警
MSG_RT_FLAG_INSTANCE	（值： 0x01000000）	实例报警的标记
MSG_RT_FLAG_TIMEINVALID	（值： 0x02000000）	无效日期/时间戳的标记
MSG_RT_FLAG_LOCKGROUP	（值： 0x04000000）	组锁定的标记
MSG_RT_FLAG_INSTANCE10	（值： 0x08000000）	过程值为 10 的新实例报警的位。
MSG_RT_FLAG_TEXTREF1	（值： 0x10000000）	文本参考 1 的位
MSG_RT_FLAG_TEXTREF2	（值： 0x20000000）	文本参考 2 的位
MSG_RT_FLAG_QUITCOUNTER	（值： 0x40000000）	通过计数器确认的位
MSG_RT_FLAG_QUITGROUP	（值： 0x80000000）	通过计数器确认的位

单个报警的标记

MSG_PARAMS_SQUIT	（值： 0x00000001）	报警受单个确认的限制。
MSG_PARAMS_HORN	（值： 0x00000002）	报警激活报警器。
MSG_PARAMS_ARCHIV	（值： 0x00000004）	将记录报警。
MSG_PARAMS_PROTOCOL	（值： 0x00000008）	将报告报警。
MSG_PARAMS_VARFLANK	（值： 0x00000010）	报警于变量下降沿时生成。
MSG_PARAMS_LOCKED	（值： 0x00000020）	启动期间锁定报警。
MSG_PARAMS_INFOTEXT	（值： 0x00000040）	报警具有信息文本。
MSG_PARAMS_LOOPINALARM	（值： 0x00000080）	报警具有报警循环。
MSG_PARAMS_APFUNC	（值： 0x00000100）	报警激活全局 AP 函数。
MSG_PARAMS_NORMDLL	（值： 0x00000200）	报警具有标准化 DLL。
MSG_PARAMS_TEXT10_EXT	（值： 0x00000400）	扩展通道诊断，可用于 V7.0 及更高版本。

状态机

MSG_QUIT_QUIT_COME	（值： 0x00000004）	到达确认
MSG_QUIT_QUIT_GO	（值： 0x00000008）	离去确认
MSG_QUIT_NOQUIT	（值： 0x00000010）	无确认
MSG_QUIT_QUIT_NOGO	（值： 0x00000020）	TM II 确认（无离去报警）
MSG_QUIT_FLASH	（值： 0x00000040）	闪烁
MSG_QUIT_FIRSTFLASH	（值： 0x00000080）	仅允许第一个报警闪烁
MSG_QUIT_NOLIST	（值： 0x00000100）	消息列表中未收到报警
MSG_QUIT_UNIQUE_USER	（值： 0x00000200）	带唯一用户名的报警
MSG_QUIT_ONL_COM_COME	（值： 0x00000400）	只在报警列表中显示到达报警的注释

组报警

MSG_COLLECT_ALL	（值： 0x00000001）	所有报警
MSG_COLLECT_QUIT	（值： 0x00000002）	所有未确认报警

MSG_COLLECT_CLASS	（值： 0x00000003）	一个报警类别的所有报警
MSG_COLLECT_TYPE	（值： 0x00000004）	一个报警类型的所有报警
MSG_COLLECT_USER	（值： 0x00000005）	用户定义的组报警

单个报警

MSG_SINGLE_OPERATION	（值： 0x00BEDBED）	操作员输入报警

报警类别

MSG_CLASS_SYSTEM_QUIT	（值： 0x00000011）	带确认的系统报警类别
MSG_CLASS_SYSTEM	（值： 0x00000012）	不带确认的系统报警类别

报警块

HIWORD 包含报警块的主类型。 LOWORD 包含报警块的子类型。

MSG_BLOCK_SYSTEM	（值： 0x00010000）	系统块
MSG_BLOCK_DATE	（值： 0x00010001）	日期
MSG_BLOCK_TIME	（值： 0x00010002）	时间
MSG_BLOCK_TIMEDIFF	（值： 0x00010003）	报警持续时间
MSG_BLOCK_SUMWIN	（值： 0x00010004）	夏令时/标准时间
MSG_BLOCK_STATE	（值： 0x00010005）	报警状态
MSG_BLOCK_QUIT	（值： 0x00010006）	确认状态
MSG_BLOCK_NUMBER	（值： 0x00010007）	报警编号
MSG_BLOCK_CLASS	（值： 0x00010008）	报警类别
MSG_BLOCK_TYP	（值： 0x00010009）	报警类型
MSG_BLOCK_AGCPU	（值： 0x0001000A）	控制器/CPU 编号
MSG_BLOCK_DMVAR	（值： 0x0001000B）	报警变量
MSG_BLOCK_LIMIT	（值： 0x0001000C）	超出限值
MSG_BLOCK_ARCHIV	（值： 0x0001000D）	日志标识符
MSG_BLOCK_PROTOCOL	（值： 0x0001000E）	报告标识符

MSG_BLOCK_COMMENT	（值： 0x0001000F）	注释
MSG_BLOCK_INFO	（值： 0x00010010）	信息文本
MSG_BLOCK_LOOPINALARM	（值： 0x00010011）	报警循环/AP 函数
MSG_BLOCK_COMPUTERNAME	（值： 0x00010012）	计算机名称
MSG_BLOCK_USERNAME	（值： 0x00010013）	用户名
MSG_BLOCK_PRIORITY	（值： 0x00010014）	优先级
MSG_BLOCK_TEXT	（值： 0x00020000）	可使用常数 MSG_BLOCK_TEXT1 到 MSG_BLOCK_TEXT16（值 0x00020001 到 0x00020010）激活最多 16 个用户文本块。
MSG_BLOCK_VALUE	（值： 0x00030000）	可使用常数 MSG_BLOCK_VALUE1 到 MSG_BLOCK_VALUE16（值 0x00030001 到 0x00030010）激活最多 16 个过程值文本块。
MSG_MAX_SYSBL		系统报警块数 = MSG_BLOCK_LOOPINALARM - MSG_BLOCK_SYSTEM
MSG_MAX_SYSBL_V6		V6.0 及更高版本中的系统块数 = MSG_BLOCK_PRIORITY - MSG_BLOCK_SYSTEM
MSG_MAX_BLOCKS		报警块总数 = (MSG_MAX_TB + MSG_MAX_PVALUE + MSG_MAX_SYSBL)
MSG_MAX_BLOCKS_V6		V6.0 及更高版本中的报警块总数 = (MSG_MAX_TB + MSG_MAX_PVALUE + MSG_MAX_SYSBL_V6)

数据格式

DATE_FORMAT_YEAR_2	（值： 0x00000001）	两位年份规范
DATE_FORMAT_YEAR_4	（值： 0x00000002）	四位年份规范
DATE_FORMAT_DMY	（值： 0x00000004）	顺序： 日、月、年
DATE_FORMAT_MDY	（值： 0x00000008）	顺序： 月、日、年
DATE_FORMAT_YMD	（值： 0x00000010）	顺序： 年、月、日

时间格式

TIME_FORMAT_HOUR	（值： 0x00000001）	小时
TIME_FORMAT_MIN	（值： 0x00000002）	分钟
TIME_FORMAT_SEC	（值： 0x00000004）	秒钟
TIME_FORMAT_MSEC	（值： 0x00000008）	毫秒
TIME_FORMAT_24HOUR	（值： 0x00000010）	24 小时格式
TIME_FORMAT_DAY	（值： 0x00000020）	小时显示

报警编号 - 格式

HiWord 包含格式规则（带前导零）；LoWord 包含位数。

MSGNR_FORMAT_NOLEADING	（值： 0x00000000）	
MSGNR_FORMAT_LEADINGZERO	（值： 0x00010000）	
MSG_BLOCK_FORMAT_RIGHT	（值： 0x00020000）	右对齐输出
MSG_BLOCK_FORMAT_CENTER	（值： 0x00040000）	居中输出
MSG_BLOCK_FORMAT_FLASH	（值： 0x80000000）	报警块闪烁

状态栏的元素

MSG_SB_HELP	（值： 0x00000001）	帮助行
MSG_SB_DATE	（值： 0x00000002）	日期
MSG_SB_TIME	（值： 0x00000004）	时间
MSG_SB_COUNTALL	（值： 0x00000008）	所有报警
MSG_SB_COUNTSEL	（值： 0x00000010）	当前报警
MSG_SB_COUNTQUIT	（值： 0x00000020）	已确认报警
MSG_SB_SELECT	（值： 0x00000040）	选择激活
MSG_SB_LOCK	（值： 0x00000080）	锁定激活
MSG_SB_HIDE	（值： 0x00000100）	隐藏报警激活
MSG_SB_WARNING	（值： 0x00000200）	隐藏信息
MSG_MAX_SBITEMS	（值： 10)	状态栏中的最大元素数

符号栏的元素

MSG_TB_MSGWIN	（值： 0x00000001）	显示“报警窗口”(Alarm window) 按钮。
MSG_TB_ARC_S	（值： 0x00000002）	显示“循环日志”(Circular log) 按钮。
MSG_TB_ARC_L	（值： 0x00000004）	显示“顺序日志”(Sequential log) 按钮。
MSG_TB_QH	（值： 0x00000008）	显示“报警器确认”(Horn acknowledgment) 按钮。
MSG_TB_QM	（值： 0x00000010）	显示“单个确认”(Single acknowledgment) 按钮。
MSG_TB_QS	（值： 0x00000020）	显示“组确认”(Group acknowledgment) 按钮。
MSG_TB_SCROLL	（值： 0x00000040）	显示“滚动函数开/关”(Scroll function On/Off) 按钮。
MSG_TB_SELECT	（值： 0x00000080）	显示“选择对话框”(Selection dialog) 按钮。
MSG_TB_LOCK	（值： 0x00000100）	显示“锁定对话框”(Lock dialog) 按钮。

MSG_TB_PROTOCOL	（值： 0x00000200）	显示“报表对话框”(Report dialog) 按钮。
MSG_TB_CITYRUF	（值： 0x00000400）	显示“市话”(City call) 按钮。
MSG_TB_RESET	（值： 0x00000800）	显示“复位按钮”(Reset button) 按钮。
MSG_TB_MSGFIRST	（值： 0x00001000）	显示“列表的开始”(Start of list) 按钮。
MSG_TB_MSGLAST	（值： 0x00002000）	显示“列表的结束”(End of list) 按钮。
MSG_TB_MSGNEXT	（值： 0x00004000）	显示“下一个报警”(Next alarm) 按钮。
MSG_TB_MSGPREV	（值： 0x00008000）	显示“上一个报警”(Previous alarm) 按钮。
MSG_TB_INFO	（值： 0x00010000）	显示“信息文本”(Info texts) 按钮。
MSG_TB_COMMENT	（值： 0x00020000）	显示“输入注释”(Enter comment) 按钮。
MSG_TB_LOOPINALARM	（值： 0x00040000）	显示“报警循环”(Loop-in-alarm) 按钮。
MSG_TB_LANGUAGESET	（值： 0x00080000）	显示“语言切换”(Language switch) 按钮。
MSG_TB_PRINT	（值： 0x00100000）	显示“打印”(Print) 按钮。
MSG_TB_LOCKWIN	（值： 0x00200000）	显示“锁定窗口”(Lock window) 按钮。
MSG_TB_UNLOCKMSG	（值： 0x00400000）	显示“释放报警”(Release alarm) 按钮。
MSG_TB_LOCK_UNLOCK_MSG	（值： 0x00400000）	显示“锁定或释放报警”(Lock or release alarm) 按钮。
MSG_TB_SORTDLG	（值： 0x00800000）	显示“排序对话框”(Sorting dialog) 按钮。
MSG_TB_TIMEBASE	（值： 0x01000000）	显示“时基对话框”(Dialog for timebase) 按钮。
MSG_TB_HITLIST	（值： 0x02000000）	显示“统计列表”(Hit list) 按钮。

MSG_TB_HIDELIST	（值： 0x04000000）	显示“隐藏消息的列表”(List of hidden messages) 按钮。
MSG_TB_UNHIDEMSG	（值： 0x08000000）	显示“显示报警”(Display alarm) 按钮。
MSG_TB_HIDE_UNHIDE_MSG	（值： 0x080000000）	显示“隐藏和显示报警”(Hide and display alarm) 按钮。
MSG_TB_HIDEDLG	（值： 0x10000000）	显示“隐藏和显示报警对话框”(Hide and display alarm dialog) 按钮。
MSG_MAX_TBITEMS	（值： 24)	状态栏中的最大操作员控件数。
MSG_MAX_TBITEMS_V6	（值： 25)	最大数目，可用于 V6 及更高版本
MSG_MAX_TBITEMS_V6_1	（值： 26)	最大数目，可用于 V6.1 及更高版本
MSG_MAX_TBITEMS_V6_2	（值： 30)	最大数目，可用于 V6.2 及更高版本
MSG_ALIGN_TOP	（值： 1)	上对齐
MSG_ALIGN_BOTTOM	（值： 2)	下对齐
MSG_ALIGN_LEFT	（值： 3)	左对齐
MSG_ALIGN_RIGHT	（值： 4)	右对齐

窗口选项

MSG_MWWIN_SWITCHENABLE	（值： 0x00000001）	过程值切换
MSG_MWWIN_PROCESS	（值： 0x00000002）	过程报警窗口
MSG_MWWIN_ARCHIV_CIRCLE	（值： 0x00000004）	循环日志
MSG_MWWIN_ARCHIV_LONG	（值： 0x00000008）	顺序日志
MSG_MWWIN_GRIDLINEHORZ	（值： 0x00000010）	水平网格线

MSG_MWWIN_GRIDLINEVERT	（值： 0x00000020）	垂直网格线
MSG_MWWIN_LINETITLE	（值： 0x00000040）	行标签
MSG_MWWIN_LINEHEIGHT	（值： 0x00000080）	行高可调
MSG_MWWIN_LINEHEIGHTFONT	（值： 0x00000100）	调整字体以适应行高
MSG_MWWIN_COLUMNTITLE	（值： 0x00000200）	列标签
MSG_MWWIN_COLUMNTITLE2	（值： 0x00000400）	2 个标题行
MSG_MWWIN_COLUMNTITLE3	（值： 0x00000800）	3 个标题行
MSG_MWWIN_COLUMNWIDTH	（值： 0x00001000）	列宽可调
MSG_MWWIN_COLUMNMOVE	（值： 0x00002000）	列可移动
MSG_MWWIN_CELLSELECT	（值： 0x00004000）	仅选择单元格（默认设置：行）
MSG_MWWIN_LINESELECT	（值： 0x00008000）	行选择
MSG_MWWIN_AUTOSCROLLOFF	（值： 0x00010000）	自动滚动关闭（默认设置：打开）
MSG_MWWIN_TITLECUT	（值： 0x00020000）	必要时用句号截断标题
MSG_MWWIN_CELLCUT	（值： 0x00040000）	必要时用句号截断单元格内容
MSG_MWWIN_TITLEBAR	（值： 0x00080000）	显示窗口标题行（加边框）
MSG_MWWIN_SIZEABLE	（值： 0x00100000）	窗口大小可调
MSG_MWWIN_TOOLTIP	（值： 0x00200000）	专为报警文本显示的工具提示
MSG_MWWIN_LOOP_IN_ALARM_DC	（值： 0x00400000）	单击“报警循环”(Loop-in-alarm)
MSG_MWWIN_SELECTFOCUSRECT	（值： 0x00800000）	通过框选（而不是反选）进行选择
MSG_MWWIN_OPMSG_HIDE	（值： 0x01000000）	操作员输入报警（隐藏时）

MSG_MWWIN_OPMSGUNHIDE	（值： 0x02000000）	操作输入报警（显示时）
MSG_MWWIN_INSTANCE_COLWIDTH	（值： 0x08000000）	实例特定列宽
MSG_MWWIN_INSTANCEBLOCKINFO	（值： 0x10000000）	实例特定报警块设置
MSG_MWWIN_OPMSG_LOCK	（值： 0x20000000）	操作员输入报警（锁定时）
MSG_MWWIN_OPMSG_UNLOCK	（值： 0x40000000）	操作员输入报警（释放时）
MSG_MWWIN_OPMSG_ACK	（值： 0x80000000）	操作员输入报警（确认时）

过滤条件

MSG_FILTER_DATE_FROM	（值： 0x00000001）	开始日期
MSG_FILTER_DATE_TO	（值： 0x00000002）	结束日期
MSG_FILTER_TIME_FROM	（值： 0x00000004）	开始时间
MSG_FILTER_TIME_TO	（值： 0x00000008）	结束时间
MSG_FILTER_NR_FROM	（值： 0x00000010）	报警编号初值
MSG_FILTER_NR_TO	（值： 0x00000020）	报警编号终值
MSG_FILTER_CLASS	（值： 0x00000040）	报警类别
MSG_FILTER_STATE	（值： 0x00000080）	报警状态
MSG_FILTER_AG_FROM	（值： 0x00000100）	AG 编号初值
MSG_FILTER_AG_TO	（值： 0x00000200）	AG 编号终值
MSG_FILTER_AGSUB_FROM	（值： 0x00000400）	AG 子编号初值
MSG_FILTER_AGSUB_TO	（值： 0x00000800）	AG 子编号终值
MSG_FILTER_ARC	（值： 0x00001000）	日志/报告标识符
MSG_FILTER_TEXT	（值： 0x00002000）	报警文本
MSG_FILTER_PVALUE	（值： 0x00004000）	过程值
MSG_FILTER_COUNTER_FROM	（值： 0x00008000）	内部报警计数器初值
MSG_FILTER_COUNTER_TO	（值： 0x00010000）	内部报警计数器终值
MSG_FILTER_PROCESSMSG	（值： 0x00020000）	过程报警

MSG_FILTER_SYSMSG	（值： 0x00040000）	系统事件
MSG_FILTER_BEDMSG	（值： 0x00080000）	操作员输入报警
MSG_FILTER_QUICKSELECT	（值： 0x00100000）	快速选择（当日等）
MSG_FILTER_TEXT_EQUAL	（值： 0x00200000）	精确报警文本
MSG_FILTER_PRIORITY_FROM	（值： 0x00400000）	优先级初值
MSG_FILTER_PRIORITY_TO	（值： 0x00800000）	优先级终值
MSG_FILTER_VISIBLEONLY	（值： 0x01000000）	显示已显示报警
MSG_FILTER_HIDDENONLY	（值： 0x02000000）	显示隐藏报警
MSG_FILTER_NODYNAMIC	（值： 0x04000000）	不选择动态文本
MSG_FILTER_DATE	值：(MSG_FILTER_DATE_FROM | MSG_FILTER_DATE_TO)	
MSG_FILTER_TIME	值：(MSG_FILTER_TIME_FROM | MSG_FILTER_TIME_TO)	
MSG_FILTER_NR	值：(MSG_FILTER_NR_FROM | MSG_FILTER_NR_TO)	

快速选择

快速选择的结束时间，请参考本地计算机的当前系统时间。 起始时间则可通过以下方式进行回推：n *（月、天、小时）。

MSG_FILTER_QUICK_MONTH	（值： 0x00000001）	最近 n 个月
MSG_FILTER_QUICK_DAYS	（值： 0x00000002）	最近 n 天
MSG_FILTER_QUICK_HOUR	（值： 0x00000003）	最近 n 小时

日志类型

MSG_ARPRO_LONGARCH	（值： 0x00000001）	顺序日志
MSG_ARPRO_SHORTARCH	（值： 0x00000002）	循环日志

MSG_ARPRO_ARCHPROT	（值： 0x00000003）	日志报表
MSG_ARPRO_MFPROT	（值： 0x00000004）	报警序列报告

日志的标记

MSG_ARPRO_PARAM_HSPONLY	（值： 0x00000001）	主内存中的短期日志
MSG_ARPRO_PARAM_CIRCLE	（值： 0x00000002）	循环日志
MSG_ARPRO_PARAM_ENDLES	（值： 0x00000004）	无限日志
MSG_ARPRO_PARAM_RELOAD	（值： 0x00000008）	掉电后重新加载
MSG_ARPRO_PARAM_LINEPRINT	（值： 0x00000010）	按行打印（仅报警序列报告）

用于锁定报警的 ID

MSG_LOCK_UNKNOWN	（值：0）	锁定未知
MSG_LOCK_CLASS	（值：1）	报警类别的锁定
MSG_LOCK_GROUP1	（值：2）	报警组等级 1 的锁定
MSG_LOCK_GROUP2	（值：3）	报警组等级 2 的锁定

日志 ID

MSG_PROT_MSG	（值：1）	报警序列报告
MSG_PROT_ARCHIV_CIRCLE	（值：2）	循环日志的日志报告
MSG_PROT_ARCHIV_LONG	（值：3）	顺序日志的日志报告

用于导出报警日志的 ID

MSG_DB_FMT_CSV	（值：1）	CSV 文件格式
MSG_DB_FMT_DBASE3	（值：2）	DBASE3 文件格式
MSG_DB_FMT_WATCOM	（值：3）	Watcom 格式

MSG_DB_FLAGS_OVERWRITE	（值：1）	覆盖现有数据记录
MSG_DB_FLAGS_ADD	（值：2）	仅添加
MSG_DB_FLAGS_DELETE	（值：4）	删除源数据记录
MSG_DB_FLAGS_NOEXPORT	（值：8）	不导出数据记录 --> 只删除（不再可用）

一般组态的标志

可在逻辑上合并标志。

MSG_GENERIC_APPEND	（值： 0x00000001）	新建
MSG_GENERIC_OVERWRITE	（值： 0x00000002）	覆盖
MSG_GENERIC_DELETE	（值： 0x00000004）	删除

变量模式

MSG_CS_VAR_NOTSET	（值： 0x0000）	未设置变量
MSG_CS_VARNAME_SET	（值： 0x0001）	已设置变量名称
MSG_CS_VARID_SET	（值： 0x0002）	已设置变量 ID
MSG_CS_VARALL_SET	（值： 0x0003）	已设置变量 ID 和名称

日志的枚举

MSG_ARCHIV_ENUM_DESC	（值： 0x00000001）	按降序排序（默认设置： 升序）

3.10.2 结构

3.10.2.1 MSG_BACKUP_STRUCT_PLUS

声明

```
typedef struct {
    SYSTEMTIME    stFrom;
    SYSTEMTIME    stTo;
    WCHAR         szArchivFile[MAX_PATH+1];
    WCHAR         szArchivTextFile[MAX_PATH+1];
    WCHAR         szCommentFile[MAX_PATH+1];
    WCHAR         szInstanceFile[MAX_PATH+1];
    WCHAR         szComment[MAX_PATH+1];
    DWORD         dwFlags;
    DWORD         dwFormat;
    }
    MSG_BACKUP_STRUCTPlus;
```

成员

stFrom

开始日期

stTo

结束日期

szArchivFile

日志导出文件的文件名和路径

szArchivTextFile

日志文本导出文件的文件名和路径

szCommentFile

注释导出文件的文件名和路径

szInstanceFile

实例导出文件的文件名和路径

szComment

用户自定义注释

dwFlags

MSG_DB_FLAGS_OVERWRITE	0x0001	覆盖现有数据记录
MSG_DB_FLAGS_ADD	0x0002	仅添加
MSG_DB_FLAGS_DELETE	0x0004	删除源数据记录

dwFormat

可以使用"m_global.h"文件中的以下常量格式化导出文件：

MSG_DB_FMT_CSV	CSV 文件格式
MSG_DB_FMT_DBASE3	DBASE3 文件格式
MSG_DB_FMT_WATCOM	Sybase 格式

注释

如果需要一个用于传递 SYSTEMTIME 参数的当前定时器，则必须使用 GetLocalTime 而非 GetSystemTime。通常，这两个函数的时间有很大差异。

所需文件

CCMSRTCLIPlus.h

API 函数

MSRTEnumBackupListPlus (页 2867)	列出导出文件的条目
MSRTGetBackupSize	确定导出文件的大小
MSRTRestore	导入报警

参见

MSRTEnumBackupListPlus (页 2867)

MSRTExportPlus (页 2869)

结构概览 (页 2733)

3.10.2.2 MSG_CLASS_STRUCT_PLUS

声明

```
typedef struct {
    WCHAR               szName[MSG_MAX_TEXTLEN+1];
    DWORD               dwName;
    DWORD               dwClassID;
    COLORREF            crStateCome[2];
    COLORREF            crStateGo[2];
    COLORREF            crStateQuit[2];
    WORD                wQuitType;
    WORD                wHornQuit;
    DWORD               dwHornVar;
    WCHAR               szState[MSG_MAX_STATE][MSG_MAX_TEXTLEN+1];
    DWORD               dwState[MSG_MAX_STATE];
    WCHAR               szStateQuit[MSG_MAX_TEXTLEN+1];
    DWORD               dwStateQuit;
    DWORD               dwCreatorID;
    DWORD               dwClassTyp;
    }
    MSG_CLASS_STRUCT_PLUS;
```

成员

szName

要指定的类别名称取决于所选的数据语言！

dwName

如果类别名称包含在项目文本中，则可在此处指定 ID。

dwClassID

报警类别 ID

crStateCome[0]	文本颜色，前景色
crStateCome[1]	背景色

crStateCome

“到达”状态的颜色

crStateGo[0]	文本颜色，前景色
crStateGo[1]	背景色

crStateGo

“离去”状态的颜色

crStateQuit[0]	文本颜色，前景色
crStateQuit[1]	背景色

crStateQuit

“已确认”状态的颜色

wQuitType

通过“m_global.h”文件中的以下常量定义确认原则：

MSG_QUIT_QUIT_COME	到达确认
MSG_QUIT_QUIT_GO	离去确认
MSG_QUIT_NOQUIT	无确认
MSG_QUIT_QUIT_NOGO	确认（无离去报警）
MSG_QUIT_FLASH	闪烁
MSG_QUIT_FIRSTFLASH	仅允许第一个报警闪烁
MSG_QUIT_NOLIST	不接收报警列表中的报警

如果状态机的常量不互相矛盾，则可通过“或”运算逐位对其进行逻辑组合。

wHornQuit

确认中央报警器

dwHornVar

中央报警器的变量

szState

“到达”、“离去”和“到达和离去”状态的文本。

dwState

TextID

szStateQuit

“已确认”状态的文本。

dwStateQuit

TextID

dwCreatorID

CreatorID

dwClassTyp

该参数为将来开发预留，必须预设为 0。

所需文件

CCMSRTCLIPlus.h

API 函数

MSRTGetClassInfoPlus (页 2819)	查询报警类别信息

参见

MSRTGetClassInfoPlus (页 2819)

结构概览 (页 2733)

3.10.2.3 MSG_COMMENT_INSTANCE_STRUCT_PLUS

声明

```
typedef struct {
    SYSTEMTIME    stTime;
    DWORD         dwMsgNrLow;
    DWORD         dwMsgNrHigh;
    WCHAR         szText[MSG_MAX_TB_CONTENT+1];
    WCHAR         szUser[MSG_MAX_USERNAME+1];
    WCHAR         szComputerName[MAX_COMPUTERNAME_LENGTH+1];
    WCHAR         szApplicationName[MSG_MAX_APPLNAME+1];
    WCHAR         szInstance[MSG_MAX_INSTANCE+1];
    }
    MSG_COMMENT_INSTANCE_STRUCT_PLUS;
```

成员

stTime

日期/时间

dwMsgNrLow

64 位报警编号的最低有效 DWORD。

dwMsgNrHigh

64 位报警编号的最高有效 DWORD，一般设为 0L。

szText

注释

szUser

用户名

szComputerName

计算机名称

szApplicationName

应用程序名称

szInstance

实例名称

注释

报警特定的注释分配给归档中的报警。该分配取决于报警编号以及报警的日期/时间。

如果传送 SYSTEMTIME 参数需要当前时间，则必须使用 GetLocalTime 函数而不是 GetSystemTime。通常，这两个函数的时间有很大差异。

所需文件

CCMSRTCLIPlus.h

API 函数

MSRTGetCommentInstancePlus (页 2860)	查询注释文本
MSRTSetCommentInstancePlus (页 2861)	指定注释文本

参见

MSRTSetCommentInstancePlus (页 2861)

MSRTGetCommentInstancePlus (页 2860)

结构概览 (页 2733)

3.10.2.4 MSG_CSDATA_STRUCT_PLUS

声明

```
typedef struct {
    DWORD     dwMsgNrLow;
    DWORD     dwMsgNrHigh;
    DWORD     dwStatus;
    WORD      wClass;
    DWORD     dwTextID[MSG_MAX_TB];
    DWORD     dwQuitVar;
    WORD      wQuitBit;
    DWORD     dwStateVar;
    WORD      wStateBit;
    DWORD     dwMsgVar;
    WORD      wMsgBit;
    WORD      wAGNr;
    WORD      wAGSubNr;
    DWORD     dwGroupID;
    DWORD     dwPriority;
    DWORD     dwHidingMask;
    }
    MSG_CSDATA_STRUCT_PLUS;
```

成员

dwMsgNrLow

64 位报警编号的最低有效 DWORD。

dwMsgNrHigh

64 位报警编号的最高有效 DWORD（一般设为 0L）。

dwStatus

通过文件"m_global.h"中的以下常量定义各报警参数：

MSG_PARAMS_SQUIT	单个确认
MSG_PARAMS_HORN	激活报警器
MSG_PARAMS_ARCHIV	将归档
MSG_PARAMS_PROTOCOL	将记录
MSG_PARAMS_VARFLANK	将在下降沿创建
MSG_PARAMS_INFOTEXT	具有信息文本。
MSG_PARAMS_LOOPINALARM	具有报警回路
MSG_PARAMS_NORMDLL	具有标准化 DLL

如果这些常量互不矛盾，则可对其进行“或”运算。

wClass

报警类别

dwTextID

对文本块的引用

dwQuitVar

确认变量。使用 DM_VARKEY (页 1869) 的 dwID。

wQuitBit

确认变量的位

dwStateVar

与变量相同的报警状态。使用 DM_VARKEY (页 1869) 的 dwID

wStateBit

wStateBit 用于指定 dwStateVar 的哪些位定义报警状态。

如果 wStateBit = 1 并且 dwStateVar 的数据类型为"8 Bit unsigned"，则 dwStateVar 的位 1 指定到达/离去状态，位 5 指定是否需要确认。

如果 wStateBit = 2 并且 dwStateVar 为字变量，则 dwStateVar 的位 2 指定到达/离去状态，位 10 指定是否需要确认。

dwMsgVar

报警变量。使用 DM_VARKEY (页 1869) 的 dwID。

wMsgBit

报警变量位

wAGNr

PLC 编号

wAGSubNr

PLC 子编号

dwGroupID

报警组 ID（内部）

dwPriority

报警优先级

dwHidingMask

显示抑制掩码

注释

通过此结构，应用程序可使用 API 函数来确定单个消息的组态数据。

所需文件

CCMSRTCLIPlus.h

API 函数

MSRTGetMsgCSDataPlus (页 2827)	查询报警组态数据

参见

MSRTGetMsgCSDataPlus (页 2827)

DM_VARKEY (页 1869)

结构概览 (页 2733)

3.10.2.5 MSG_CSDATA_EX_STRUCT_PLUS

声明

```
typedef struct {
    DWORD     dwMsgNrLow;
    DWORD     dwMsgNrHigh;
    DWORD     dwStatus;
    WORD      dwClass;
    DWORD     dwTextID[MSG_MAX_TB];
    DWORD     lpstrQuitVar;
    WORD      dwQuitBit;
    DWORD     lpstrStateVar;
    WORD      dwStateBit;
    DWORD     lpstrMsgVar;
    WORD      dwMsgBit;
    WORD      dwAGNr;
    WORD      dwAGSubNr;
    DWORD     dwGroupID;
    DWORD     dwPriority;
    DWORD     dwHidingMask;
    }
    MSG_CSDATA_STRUCT_PLUS;
```

成员

dwMsgNrLow

64 位报警编号的最低有效 DWORD。

dwMsgNrHigh

64 位报警编号的最高有效 DWORD（一般设为 0L）。

dwStatus

通过文件“m_global.h”中的以下常量定义各报警参数：

MSG_PARAMS_SQUIT	单个确认
MSG_PARAMS_HORN	激活报警器
MSG_PARAMS_ARCHIV	将归档
MSG_PARAMS_PROTOCOL	将记录
MSG_PARAMS_VARFLANK	将在下降沿创建
MSG_PARAMS_INFOTEXT	具有信息文本。

MSG_PARAMS_LOOPINALARM	具有报警回路
MSG_PARAMS_NORMDLL	具有标准化 DLL

如果这些常量互不矛盾，则可对其进行“或”运算。

wClass

报警类别

dwTextID

对文本块的引用

lpstrQuitVar

确认变量（如果不是本地变量，则带有服务器前缀）。

dwQuitBit

确认变量的位

lpstrStateVar

以变量形式表示的报警状态（如果不是本地变量，则带有服务器前缀）。

dwStateBit

dwStateBit 用于指定 lpstrStateVar 中定义报警状态的位。

如果 dwStateBit = 1 并且 lpstrStateVar 的数据类型为“8 Bit unsigned”，则 lpstrStateVar 的位 1 指定到达/离去状态，位 5 指定是否需要确认。

如果 dwStateBit = 2 并且 lpstrStateVar 为字变量，则 lpstrStateVar 的位 2 指定到达/离去状态，位 10 指定是否需要确认。

lpstrMsgVar

报警变量（如果不是本地变量，则带有服务器前缀）。

dwMsgBit

报警变量位

dwAGNr

PLC 编号

dwAGSubNr

PLC 子编号

dwGroupID

报警组 ID（内部）

dwPriority

报警优先级

dwHidingMask

显示抑制掩码

注释

通过此结构，应用程序可使用 API 函数来确定单个消息的组态数据。

所需文件

CCMSRTCLIPlus.h

API 函数

MSRTGetMsgCSDataExPlus (页 2830)	查询指定报警服务器中报警的组态数据
MSG_CSDATA_EX_CALLBACK_PROC_PLUS (页 2833)	用于评估组态数据的回调函数

3.10.2.6 MSG_FILTER_STRUCT_PLUS

声明

```
typedef struct {
    WCHAR          szFilterName[MSG_MAX_TEXTLEN+1];
    DWORD          dwFilter;
    SYSTEMTIME     st[2];
    DWORD          dwMsgNrLow[2];
    DWORD          dwMsgNrHigh[2];
    VARIANT        vMsgClasses;
    DWORD          dwMsgState;
    WORD           wAGNr[2];
    WORD           wAGSubNr[2];
    DWORD          dwArchivMode;
    WCHAR          szTB[MSG_MAX_TB][MSG_MAX_TB_CONTENT+1]
    DWORD          dwTB;
    double         dPValue[MSG_MAX_PVALUE][2];
    DWORD          dwPValue[2];
    ULONGLONG      ullMsgCounter[2]; /* in Script it has the union type
MSGULONGLONG, see CCMsrtCliPlus.h */
    DWORD          dwQuickSelect;
    DWORD          dwPriority[2];
    WCHAR         szInstance[MSG_MAX_INSTANCE+1];
    }
    MSG_FILTER_STRUCT_PLUS;
```

成员

szFilterName

过滤器名称

dwFilter

通过文件“m_global.h”中的以下常量定义过滤条件：

MSG_FILTER_DATE_FROM	开始日期
MSG_FILTER_DATE_TO	结束日期
MSG_FILTER_TIME_FROM	开始时间
MSG_FILTER_TIME_TO	结束时间
MSG_FILTER_NR_FROM	报警编号初值
MSG_FILTER_NR_TO	报警编号终值

MSG_FILTER_CLASS	报警类别
MSG_FILTER_STATE	报警状态
MSG_FILTER_AG_FROM	AG 编号初值
MSG_FILTER_AG_TO	AG 编号终值
MSG_FILTER_AGSUB_FROM	AG 子编号初值
MSG_FILTER_AGSUB_TO	AG 子编号终值
MSG_FILTER_TEXT	报警文本
MSG_FILTER_PVALUE	过程值
MSG_FILTER_COUNTER_FROM	内部报警计数器初值
MSG_FILTER_COUNTER_TO	内部报警计数器终值
MSG_FILTER_PROCESSMSG	过程报警（保留）
MSG_FILTER_SYSMSG	系统报警（保留）
MSG_FILTER_BEDMSG	操作员输入报警（保留）
MSG_FILTER_VISIBLEONLY	显示可见报警（不包括在过滤条件中）
MSG_FILTER_HIDDENONLY	显示隐藏报警（不包括在过滤条件中）
MSG_FILTER_NODYNAMIC	不选择动态文本（不包括在过滤条件中）
MSG_FILTER_DATE	日期跨度
MSG_FILTER_TIME	时间跨度
MSG_FILTER_NR	报警编号跨度

各项过滤条件均可进行“或”运算。

st

st[0] 起点日期和时间，st[1] 终点日期和时间

如果使用过滤条件 MSG_FILTER_DATE、MSG_FILTER_DATE_FROM、MSG_FILTER_DATE_TO、MSG_FILTER_TIME、MSG_FILTER_TIME_FROM 或 MSG_FILTER_TIME_To，则必须分配 st。

如果需要一个用于传递 SYSTEMTIME 参数的当前定时器，则必须使用 GetLocalTime 而非 GetSystemTime。通常，这两个函数的时间有很大差异。

dwMsgNrLow

dwMsgNrLow 64 位报警编号的最低有效 DWORD。此处的 dwMsgNr[0] 为起始编号（起始），dwMsgNr[1] 为结束编号（结束）。

dwMsgNrHigh

dwMsgHigh 64 位报警编号的最高有效 DWORD（一般设为 0L）。

如果您使用 MSG_FILTER_NR、MSG_FILTER_NR_FROM 或 MSG_FILTER_NR_TO 过滤条件，则必须分配 dwMsgNrLow 和 dwMsgNrHigh。

vMsgClasses

以一维变量数组形式表示的已用报警类别列表。该列表不支持报警类型 (MsgType)。由用户确定工程组态系统中报警类别的 ID。

dwMsgState

按位编码的报警状态。

如果使用过滤条件 MSG_FILTER_STATE，则分配此字段。

使用 MSRTCheckWinFilter 检查以下状态值：

MSG_STATE_COME	到达报警
MSG_STATE_GO	离去报警
MSG_STATE_QUIT	报警已确认
MSG_STATE_QUIT_SYSTEM	系统确认的报警
MSG_STATE_QUIT_EMERGENCY	紧急确认

必须自行过滤其它状态值（如 MSG_STATE_HIDE 或 MSG_STATE_UNHIDE），因为过滤条件中不考虑这些值。要过滤这些状态值，使用 MSG_FILTER_HIDDENONLY 等。

wAGNr

wAGNr [0] 第一个 PLC 编号，wAGNr [1] 最后一个 PLC 编号。

如果使用过滤条件 MSG_FILTER_AG_FROM 或 MSG_FILTER_AG_TO，则必须分配 wAGNr。

wAGSubNr

wAGSubNr [0] 第一个 PLC 编号，wAGSubNr [1] 最后一个 PLC 编号。

如果使用过滤条件 MSG_FILTER_AGSUB_FROM 或 MSG_FILTER_AGSUB_TO，则必须分配 wAGSubNr。

dwArchivMode

日志或报表的 ID。必须为该参数分配 0。

szTB

文本块的文本

如果使用过滤条件 MSG_FILTER_TEXT，则必须分配 szTB。

dwTB

活动文本块，按位编码

如果使用过滤条件 MSG_FILTER_TEXT，则必须分配 dwTB。

dPValue

过程值范围

如果使用过滤条件 MSG_FILTER_PVALUE，则必须分配 dPValue。

dwPValue

过程值，按位编码

如果使用过滤条件 MSG_FILTER_PVALUE，则必须分配 dwPValue。

ullMgCounter

内部 64 位报警计数器

如果使用过滤条件 MSG_FILTER_COUNTER_...，则必须分配 dwllMsgCounter。在脚本中使用时，ullMsgCounter 的类型不为 ULONGLONG，而是 union MSGULONGLONG（请参见 CCMsRtCliPlus.h）。

dwQuickSelect

该参数为将来开发预留，必须预设为 0。

dwPriority

报警优先级

szInstance

实例名称。如果未提供 szInstance（空字符串），则创建正常报警。

所需文件

CCMsRTCliPlus.h

API 函数

MSRTCheckWinFilterPlus (页 2852)	检查报警过滤条件
MSRTGetFilterDataPlus (页 2850)	查询报警过滤条件
MSRTSetMsgFilterPlus (页 2854)	设置报警过滤器
MSRTSetMsgWinFilterPlus (页 2856)	为报警窗口设置报警过滤器
MSRTGetLastMsgWithCommentPlus (页 2821)	查询最后一个已归档报警
MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSRTGetLastMsgWithCommentPlus (页 2821)

MSRTSetMsgWinFilterPlus (页 2856)

MSRTStartMsgServicePlus (页 2791)

MSRTSetMsgFilterPlus (页 2854)

MSRTCheckWinFilterPlus (页 2852)

MSRTGetFilterDataPlus (页 2850)

结构概览 (页 2733)

MSGULONGLONG (页 2790)

3.10.2.7 MSG_HELPTEXTS_STRUCT_PLUS

声明

```
typedef struct {
    DWORD     dwMsgNrLow;
    DWORD     dwMsgNrHigh;
    DWORD     dwLCID;
    DWORD     dwFlags;
    DWORD     dwResponseTime;
    LPWSTR    lpszHelp_Description;
    DWORD     dwHelp_DescriptionCount;
    LPWSTR    lpszHelp_Causes;
    DWORD     dwHelp_CausesCount;
    LPWSTR    lpszHelp_Action;
    DWORD     dwHelp_ActionCount;
    LPWSTR    lpszHelp_Consequence;
    DWORD     dwHelp_ConsequenceCount;
    LPWSTR    lpszHelp_Reserved;
    DWORD     dwHelp_ReservedCount;
    }
    MSG_HELPTEXTS_STRUCT_PLUS;
```

成员

dwMsgNrLow

64 位报警编号的最低有效 DWORD。必须将编号指定为输入参数。

dwMsgNrHigh

64 位报警编号的最高有效 DWORD（一般设为 0L）。必须将编号指定为输入参数。

dwLCID

报警帮助文本所用语言的语言 ID。必须将 ID 指定为输入参数。

dwFlags

如果已分配自行指定的缓冲区，则函数不会更改或考虑 USERALLOC_xxx 标记。

如果未指定缓冲区，并且缓冲区是由函数分配的，则函数会设置 FUNCALLOC_xxx 标记。

如果指定的缓冲区过小，且函数已截短帮助文本，则会设置 TRUNCATE_xxx 标记。

调用函数并处理帮助文本后，用户必须再次释放已分配的缓冲区。另请参见"MSRTAPI.H"中的定义

USERALLOC_HLPDESCRIPTION	0x00000001	用户已为"Description"分配缓冲区。
FUNCALLOC_HLPDESCRIPTION	0x00000002	函数已为"Description"分配缓冲区。
USERALLOC_HLPCAUSES	0x00000004	用户已为"Causes"分配缓冲区。
FUNCALLOC_HLPCAUSES	0x00000008	函数已为"Causes"分配缓冲区。
USERALLOC_HLPACTION	0x00000010	用户已为"Action"分配缓冲区。
FUNCALLOC_HLPACTION	0x00000020	函数已为"Action"分配缓冲区。
USERALLOC_HLPCONSEQUENCE	0x00000040	用户已为"Consequence"分配缓冲区。
FUNCALLOC_HLPCONSEQUENCE	0x00000080	函数已为"Consequence"分配缓冲区。
USERALLOC_HLPRESERVED	0x00000100	保留供以后使用。
FUNCALLOC_HLPRESERVED	0x00000200	保留供以后使用。
TRUNCATE_HLPDESCRIPTION	0x00010000	"Description"的缓冲区过小。文本已被截短。

TRUNCATE_HLPCAUSES	0x0002 0000	“Causes”的缓冲区过小。文本已被截短。
TRUNCATE_HLPACTION	0x0004 0000	“Action”的缓冲区过小。文本已被截短。
TRUNCATE_HLPCONSEQUENCE	0x0008 0000	“Consequence”的缓冲区过小。文本已被截短。

如果这些常量互不矛盾，则可对其进行“或”运算。

dwResponseTime

必须遵守的报警响应时间（以秒表示）会在此返回。

lpszHelp_Description

指向用于存储“Description”帮助文本的文本缓冲区的指针。

如果设置“NULL”，函数会将指针设为在内部为“Description”分配的缓冲区。
“dwHelp_DescriptionCount”会设为内部缓冲区大小。

dwHelp_DescriptionCount

以字符数表示的缓冲区大小。
如果设置“0L”，函数会设置缓冲区大小，并在“lpszHelp_Description”中存储一个指针。

lpszHelp_Causes

指向用于存储“Causes”帮助文本的文本缓冲区的指针。

如果设置“NULL”，函数会将指针设为在内部为“Description”分配的缓冲区。
“dwHelp_CausesCount”会设为内部缓冲区大小。

dwHelp_CausesCount

以字符数表示的缓冲区大小。
如果设置“0L”，函数会设置缓冲区大小，并在“lpszHelp_Causes”中存储一个指针。

lpszHelp_Action

指向用于存储“Action”帮助文本的文本缓冲区的指针。

如果设置“NULL”，函数会将指针设为在内部为“Description”分配的缓冲区。
“dwHelp_ActionCount”会设为内部缓冲区大小。

dwHelp_Action

以字符数表示的缓冲区大小。
如果设置“0L”，函数会设置缓冲区大小，并在“lpszHelp_Action”中存储一个指针。

lpszHelp_Consequence

指向用于存储"Consequnce"帮助文本的文本缓冲区的指针。

如果设置"NULL"，函数会将指针设为在内部为"Description"分配的缓冲区。
"dwHelp_ConsequenceCount"会设为内部缓冲区大小。

dwHelp_ConsequenceCount

以字符数表示的缓冲区大小。
如果设置"0L"，函数会设置缓冲区大小，并在"lpszHelp_Consequence"中存储一个指针。

lpszHelp_Reserved

保留供以后使用。

dwHelp_Reserved

保留供以后使用。

注释

通过此结构，应用程序可使用 API 函数来确定为单个消息的指定语言组态的帮助文本。

所需文件

CCMSRTCLIPlus.h

API 函数

MSRTGetMsgHelptextsPlus (页 2834)	查询报警的组态帮助文本

3.10.2.8 MSG_INFOTEXT_STRUCT_PLUS

声明

```
typedef struct {
    DWORD    dwMsgNrLow;
    DWORD    dwMsgNrHigh;
    WCHAR    szText[MSG_MAX_TB_CONTENT+1];
    }
    MSG_INFOTEXT_STRUCT_PLUS;
```

成员

dwMsgNrLow

64 位报警编号的最低有效 DWORD。

dwMsgNrHigh

64 位报警编号的最高有效 DWORD，一般设为 0L。

szText

信息文本

所需文件

CCMSRTCLIPlus.h

API 函数

MSRTGetInfotextPlus (页 2864)	查询信息文本
MSRTSetInfotextPlus (页 2865)	指定信息文本

参见

MSRTGetInfotextPlus (页 2864)

MSRTSetInfotextPlus (页 2865)

结构概览 (页 2733)

3.10.2.9 MSG_RTCREATE_STRUCT_PLUS

声明

```
typedef struct {
    DWORD                          dwMsgState;
    DWORD                          dwMsgNrLow;
    DWORD                          dwMsgNrHigh;
    DWORD                          dwFlags;
    SYSTEMTIME                     stMsgTime;
    double                         dPValue[MSG_MAX_PVALUE];
    WORD                           wPValueUsed;
    MSG_TEXTVAL256_STRUCT_PLUS     mtTextValueW[MSG_MAX_PVALUE]
    WORD                           wTextValueUsed;
    }
    MSG_RTCREATE_STRUCT_PLUS;
```

成员

dwMsgState

遵照"m_global.h"文件中的常量的报警状态：

MSG_STATE_COME	到达报警
MSG_STATE_GO	离去报警
MSG_STATE_QUIT	报警已确认
MSG_STATE_LOCK	报警已锁定
MSG_STATE_UNLOCK	报警已释放
MSG_STATE_HIDE	报警已隐藏
MSG_STATE_UNHIDE	报警已显示
MSG_STATE_HIDE_MANUAL	报警已手动隐藏
MSG_STATE_UNHIDE_MANUAL	报警已手动显示

dwMsgNrLow

64 位报警编号的最低有效 DWORD。

dwMsgNrHigh

64 位报警编号的最高有效 DWORD（一般设为 0L）。

dwFlags

时基设置。“MSG_TIMESTAMP_LOCAL”默认有效。利用“MSG_TIMESTAMP_UTC”，可将每种情况下使用的函数单独切换为“UTC”。

MSG_TIMESTAMP_LOCAL	将时基切换为“LOCALTIME”。
MSG_TIMESTAMP_UTC	将时基切换为“UTC”。

stMsgTime

报警到达或离去的日期和时间。确认时将使用 PLC 或操作系统的系统时间。

如果需要一个用于传递 SYSTEMTIME 参数的当前定时器，则必须使用 GetLocalTime 而非 GetSystemTime。通常，这两个函数的时间有很大差异。

dPValue

数字过程值。

wPValueUsed

按位编码形式使用的过程值。

应仅在 wPValueUsed 或 wTextValueUsed 两个成员的一个成员中设置各个位，因为仅数字或文本在技术上可作为关联值。对于错误的双重条目，dPValue 具有优先权。

mtTextValue

具有相关文本值的 MSG_TEXTVAL256_STRUCT_PLUS (页 2788) 类型结构。

wTextValueUsed

按位编码形式使用的文本过程值。

应仅在 wPValueUsed 或 wTextValueUsed 两个成员的一个成员中设置各个位，因为仅数字或文本在技术上可作为关联值。对于错误的双重条目，dPValue 具有优先权。

所需文件

CCMSRTCLIPlus.h

API 函数

MSRTCreateMsgInstanceWithCommentPlus (页 2808)	创建报警

参见

MSG_TEXTVAL256_STRUCT_PLUS (页 2788)

结构概览 (页 2733)

MSRTCreateMsgInstanceWithCommentPlus (页 2808)

MSRTCreateMsgPlus (页 2812)

3.10.2.10 MSG_RTDATA_INSTANCE_STRUCT_PLUS

声明

```
typedef struct {
    DWORD                   dwMsgState;
    DWORD                   dwMsgNrLow;
    DWORD                   dwMsgNrHigh;
    SYSTEMTIME              stMsgTime;
    DWORD                   dwTimeDiff;
    ULONGLONG               ullCounter;
    DWORD                   dwFlags;
    WORD                    wPValueUsed;
    WORD                    wTextValueUsed;
    double                  dPValue[MSG_MAX_PVALUE];
    MSG_TEXTVAL256_STRUCT_PLUS    mtTextValue[MSG_MAX_PVALUE]
    TCHAR                   szInstance[MSG_MAX_INSTANCE+1];
    }
    MSG_RTDATA_INSTANCE_STRUCT_PLUS;
```

成员

dwMsgState

遵照“m_global.h”文件中的常量的报警状态：

列出的状态仅供系统内部使用且只能由 API 通过 MSRTCreateMsgInstancePlus 设置。

MSG_STATE_COME	到达报警
MSG_STATE_GO	离去报警
MSG_STATE_QUIT	报警已确认
MSG_STATE_LOCK	报警已锁定
MSG_STATE_UNLOCK	报警已释放

MSG_STATE_HIDE	报警已隐藏
MSG_STATE_UNHIDE	报警已显示
MSG_STATE_HIDE_MANUAL	报警已手动隐藏

dwMsgNrLow

64 位报警编号的最低有效 DWORD。

dwMsgNrHigh

64 位报警编号的最高有效 DWORD（一般设为 0L）。

stMsgTime

报警到达或离去的日期和时间。

dwTimeDiff

到达或离去持续时间，报文时间（以秒为单位）。

ullCounter

内部 64 位报警计数器。

dwFlags

遵照"m_global.h"文件中的常量的标记：

报警视图中报警的标记		
MSG_RT_FLAG_COME	0x00000001	到达报警
MSG_RT_FLAG_GO	0x00000002	离去报警
MSG_RT_FLAG_QUIT_COME	0x00000004	到达确认
MSG_RT_FLAG_QUIT_GO	0x00000008	离去确认
MSG_RT_FLAG_NOQUIT	0x00000010	无确认
MSG_RT_FLAG_QUIT_NOGO	0x00000020	TM II 确认
MSG_RT_FLAG_FLASH	0x00000040	闪烁
MSG_RT_FLAG_FIRSTFLASH	0x00000080	仅允许第一个报警闪烁
MSG_RT_FLAG_DELETE	0x00010000	可以删除报警
MSG_RT_FLAG_SUMTIME	0x00020000	夏令时标记
MSG_RT_FLAG_SQUIT	0x00040000	报警具有组确认
MSG_RT_FLAG_QUITNR	0x00080000	报警编号确认
MSG_RT_FLAG_MSGLIST	0x00100000	报警列表中的报警
MSG_RT_FLAG_COMMENT	0x00200000	报警具有注释

报警视图中报警的标记		
MSG_RT_FLAG_ARCHIV	0x00400000	记录报警
MSG_RT_FLAG_PROTOCOL	0x00800000	报告报警
MSG_RT_FLAG_INSTANCE	0x01000000	实例报警的标记
MSG_RT_FLAG_TIMEINVALID	0x02000000	无效日期/时间戳的标记
MSG_RT_FLAG_LOCKGROUP	0x04000000	组锁定的标记
MSG_RT_FLAG_INSTANCE10	0x08000000	过程值为 10 的新实例报警的位。
MSG_RT_FLAG_TEXTREF1	0x10000000	文本参考 1 的位。
MSG_RT_FLAG_TEXTREF2	0x20000000	文本参考 2 的位。
MSG_RT_FLAG_QUITCOUNTER	0x40000000	通过计数器确认的位
MSG_RT_FLAG_QUITGROUP	0x80000000	通过计数器确认的位

日志中报警的标记		
MSG_FLAG_SUMTIME	0x00000001	夏令时激活
MSG_FLAG_COMMENT	0x00000002	报警具有注释
MSG_FLAG_ARCHIV	0x00000004	日志
MSG_FLAG_PROTOCOL	0x00000008	报告
MSG_FLAG_TEXTVALUES	0x00000010	报警具有相关联的文本值
MSG_FLAG_TIMEINVALID	0x00000020	无效日期/时间戳位
MSG_FLAG_INSTANCE	0x00000040	实例报警标识符
MSG_FLAG_INSTANCE10	0x08000000	过程值为 10 的实例报警，从 V5.0 SP2 起可用
MSG_FLAG_TEXTREF1	0x10000000	文本参考 1
MSG_FLAG_TEXTREF2	0x20000000	文本参考 2

用于设置时基的标记	
MSG_TIMESTAMP_LOCAL	将时基切换为“LOCALTIME”。
MSG_TIMESTAMP_UTC	将时基切换为“UTC”。
“MSG_TIMESTAMP_LOCAL”默认有效。	

wPValueUsed

按位编码形式使用的过程值。

应仅在 wPValueUsed 或 wTextValueUsed 两个成员的一个成员中设置各个位，因为仅数字或文本在技术上可作为关联值。对于错误的双重条目，dPValue 具有优先权。

wTextValueUsed

使用的文本值，按位编码。

应仅在 wPValueUsed 或 wTextValueUsed 两个成员的一个成员中设置各个位，因为仅数字或文本在技术上可作为关联值。

对于错误的双重条目，dPValue 具有优先权。

dPValue

过程值。

mtTextValue

具有相关文本值的 MSG_TEXTVAL256_STRUCT_PLUS (页 2788) 类型结构。

szInstance

实例名称。如果未提供 szInstance（空字符串），则创建常规报警

所需文件

CCMSRTCLIPlus.h

API 函数

MSRTCreateMsgInstancePlus (页 2810)	创建报警
MSG_SERVICE_NOTIFY_PROC_PLUS (页 2798)	发送和接收服务（回调）

参见

MSRTCreateMsgInstancePlus (页 2810)

MSG_TEXTVAL256_STRUCT_PLUS (页 2788)

MSGULONGLONG (页 2790)

3.10.2.11 MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS

声明

```
typedef struct {
    DWORD                       dwMsgState;
    DWORD                       dwMsgNrLow;
    DWORD                       dwMsgNrHigh;
    SYSTEMTIME                  stMsgTime;
    DWORD                       dwTimeDiff;
    DWORD                       ullCounter;
    DWORD                       dwFlags;
    WORD                        wPValueUsed;
    WORD                        wTextValueUsed;
    double                      dPValue[MSG_MAX_PVALUE];
    MSG_TEXTVAL256_STRUCT_PLUS  mtTextValue[MSG_MAX_PVALUE];
    WCHAR                       szInstance[MSG_MAX_INSTANCE+1];
    WCHAR                       szComment[MSG_MAX_TB_CONTENT+1];
    WCHAR                       szUser[MSG_MAX_USERNAME+1];
    WCHAR                       szComputerName[MAX_COMPUTERNAME_LENGTH+1];
    WCHAR                       szApplicationName[MSG_MAX_APPLNAME+1];
    WCHAR                       szServerPrefix[_MAX_PATH+1];
    }
    MSG_RTDATA_INSTANCECOMMENT_STRUCTPlus;
```

成员

dwMsgState

遵照“m_global.h”文件中的常量的报警状态：

列出的状态仅由系统内部使用并且只能由 API 通过 MSRTCreateMsgInstanceWithCommentPlus 设置。

MSG_STATE_COME	到达报警
MSG_STATE_GO	离去报警
MSG_STATE_QUIT	报警已确认
MSG_STATE_LOCK	报警已锁定
MSG_STATE_UNLOCK	报警已释放
MSG_STATE_HIDE	报警已隐藏
MSG_STATE_UNHIDE	报警已显示

MSG_STATE_HIDE_MANUAL	报警已手动隐藏
MSG_STATE_UNHIDE_MANUAL	报警已手动显示

dwMsgNrLow

64 位报警编号的最低有效 DWORD。

dwMsgNrHigh

64 位报警编号的最高有效 DWORD（一般设为 0L）。

stMsgTime

报警到达或离去的日期和时间。

dwTimeDiff

到达或离去持续时间，报文时间（以秒为单位）。

ullCounter

内部 64 位报警计数器。

dwFlags

遵照"m_global.h"文件中的常量的标记：

报警视图中报警的标记		
MSG_RT_FLAG_COME	0x00000001	到达报警
MSG_RT_FLAG_GO	0x00000002	离去报警
MSG_RT_FLAG_QUIT_COME	0x00000004	到达确认
MSG_RT_FLAG_QUIT_GO	0x00000008	离去确认
MSG_RT_FLAG_NOQUIT	0x00000010	无确认
MSG_RT_FLAG_QUIT_NOGO	0x00000020	TM II 确认
MSG_RT_FLAG_FLASH	0x00000040	闪烁
MSG_RT_FLAG_FIRSTFLASH	0x00000080	仅允许第一个报警闪烁
MSG_RT_FLAG_DELETE	0x00010000	可以删除报警
MSG_RT_FLAG_SUMTIME	0x00020000	夏令时标记
MSG_RT_FLAG_SQUIT	0x00040000	报警具有组确认
MSG_RT_FLAG_QUITNR	0x00080000	报警编号确认
MSG_RT_FLAG_MSGLIST	0x00100000	报警列表中的报警
MSG_RT_FLAG_COMMENT	0x00200000	报警具有注释
MSG_RT_FLAG_ARCHIV	0x00400000	记录报警

报警视图中报警的标记		
MSG_RT_FLAG_PROTOCOL	0x00800000	报告报警
MSG_RT_FLAG_INSTANCE	0x01000000	实例报警的标记
MSG_RT_FLAG_TIMEINVALID	0x02000000	无效日期/时间戳的标记
MSG_RT_FLAG_LOCKGROUP	0x04000000	组锁定的标记
MSG_RT_FLAG_INSTANCE10	0x08000000	过程值为 10 的新实例报警的位。
MSG_RT_FLAG_TEXTREF1	0x10000000	文本参考 1 的位。
MSG_RT_FLAG_TEXTREF2	0x20000000	文本参考 2 的位。
MSG_RT_FLAG_QUITCOUNTER	0x40000000	通过计数器确认的位
MSG_RT_FLAG_QUITGROUP	0x80000000	通过计数器确认的位

日志中报警的标记		
MSG_FLAG_SUMTIME	0x00000001	夏令时激活
MSG_FLAG_COMMENT	0x00000002	报警具有注释
MSG_FLAG_ARCHIV	0x00000004	归档
MSG_FLAG_PROTOCOL	0x00000008	报告
MSG_FLAG_TEXTVALUES	0x00000010	报警具有相关联的文本值
MSG_FLAG_TIMEINVALID	0x00000020	无效日期/时间戳位
MSG_FLAG_INSTANCE	0x00000040	实例报警标识符
MSG_FLAG_INSTANCE10	0x08000000	过程值为 10 的实例报警，从 V5.0 SP2 起可用
MSG_FLAG_TEXTREF1	0x10000000	文本参考 1
MSG_FLAG_TEXTREF2	0x20000000	文本参考 2

用于设置时基的标记	
MSG_TIMESTAMP_LOCAL	将时基切换为“LOCALTIME”。
MSG_TIMESTAMP_UTC	将时基切换为“UTC”。
“MSG_TIMESTAMP_LOCAL”默认有效。	

通过 MSRTCreateMsgInstanceWithCommentPlus 函数，无法将标记发送到报警服务器。会忽略标记。

wPValueUsed

按位编码形式使用的过程值。

应仅在 wPValueUsed 或 wTextValueUsed 两个成员的一个成员中设置各个位，因为仅数字或文本在技术上可作为关联值。对于错误的双重条目，dPValue 具有优先权。

wTextValueUsed

使用的文本值，按位编码。

应仅在 wPValueUsed 或 wTextValueUsed 两个成员的一个成员中设置各个位，因为仅数字或文本在技术上可作为关联值。

对于错误的双重条目，dPValue 具有优先权。

dPValue

过程值。

mtTextValue

具有相关文本值的 MSG_TEXTVAL256_STRUCT_PLUS (页 2788) 类型结构。

szInstance

实例名称。如果未提供 szInstance（空字符串），则创建常规报警

szComment

注释名称

szUser

用户名。

szComputerName

计算机名称。

szApplicationName

应用程序名称。

szServerPrefix

服务器前缀。

所需文件

CCMSRTCLIPlus.h

API 函数

MSRTCreateMsgInstanceWithCommentPlus (页 2808)	创建报警
MSRTGetLastMsgWithCommentPlus (页 2821)	查询最后一个已归档报警
MSG_SERVICE_NOTIFY_PROC_PLUS (页 2798)	回调函数

参见

MSRTCreateMsgInstanceWithCommentPlus (页 2808)

MSRTGetLastMsgWithCommentPlus (页 2821)

MSG_SERVICE_NOTIFY_PROC_PLUS (页 2798)

MSRTGetSelectedMsgPlus (页 2839)

MSRTCheckWinFilterPlus (页 2852)

结构概览 (页 2733)

MSGULONGLONG (页 2790)

3.10.2.12 MSG_RTGROUPENUM_STRUCT_PLUS

声明

```
typedef struct {
        BOOL     fIDUsed;
        DWORD    dwID;
        TCHAR    szName[MSG_MAX_TEXTLEN+1];
        DWORD    dwMsgCount;
        DWORD    dwMsg[MSG_MAX_GROUPITEMS];
        }
        MSG_RTGROUPENUM_STRUCT_PLUS;
```

成员

fIDUsed

指定是使用组 ID (fIDUsed = TRUE) 还是名称 (fIDUsed = FALSE) 来指定报警组。

dwID

组 ID

szName

组名称

dwMsgCount

dwMsg 中的单个报警数

dwMsg

锁定的单个报警

所需文件

CCMSRTCLIPlus.h

API 函数

MSRTEnumGroupMsgPlus (页 2845)	列出报警组单个报警
MSG_SERVICE_NOTIFY_PROC_PLUS (页 2798)	回调函数

参见

MSG_SERVICE_NOTIFY_PROC_PLUS (页 2798)

MSRTEnumGroupMsgPlus (页 2845)

结构概览 (页 2733)

3.10.2.13 MSG_RTGROUPSET_STRUCT_PLUS

声明

```
typedef struct {
    BOOL          fIDUsed;
    DWORD         dwID;
    WCHAR         szName[MSG_MAX_TEXTLEN+1];
    SYSTEMTIME    stTime;
    DWORD         dwData;
    }
    MSG_RTGROUPSET_STRUCT_PLUS;
```

成员

fIDUsed

指定是使用组 ID (fIDUsed = TRUE) 还是名称 (fIDUsed = FALSE) 来指定报警组。

dwID

组 ID（对于报警类别、报警类型）

szName

组名称

stTime

发生的时间

dwData

如果在 MSRTLockGroupPlus 范围内使用该结构，则 dwData = 1 表示锁定，dwData = 0 表示释放

所需文件

CCMSRTCLIPlus.h

API 函数

MSRTLockGroupPlus (页 2847)	锁定报警组
MSRTEnumGroupMsgPlus (页 2845)	列出报警组的各个报警
MSRTQuitGroupPlus (页 2849)	确认报警组

参见

MSRTEnumGroupMsgPlus (页 2845)

MSRTLockGroupPlus (页 2847)

MSRTQuitGroupPlus (页 2849)

结构概览 (页 2733)

3.10.2.14 MSG_RTLOCK_STRUCT_PLUS

声明

```
typedef struct {
        WCHAR    szName[MSG_MAX_TEXTLEN+1];
        WCHAR    szParent[MSG_MAX_TEXTLEN+1];
        DWORD    dwLockID;
        BOOL     fLock;
        DWORD    dwMsgNum;
        DWORD    dwMsgLock[MSG_MAX_LOCKITEMS];
        }
        MSG_RTLOCK_STRUCT_PLUS;
```

成员

szName

锁定名称

szParent

父组

dwLockID

锁定 ID

fLock

锁定和释放

dwMsgNum

锁定的报警数

dwMsgLock

锁定的单个报警

所需文件

CCMSRTCLIPlus.h

API 函数

MSG_SERVICE_NOTIFY_PROC_PLUS (页 2798)	回调函数

参见

MSG_SERVICE_NOTIFY_PROC_PLUS (页 2798)

结构概览 (页 2733)

3.10.2.15 MSG_TEXTVAL256_STRUCT_PLUS

声明

```
typedef struct {
WCHAR szText[MSG_MAX_TEXTVAL256+1];
}
MSG_TEXTVAL256_STRUCT_PLUS;
```

注释

在以下结构中使用此结构：

- MSG_RTCREATE_STRUCT_PLUS (页 2774)
- MSG_RTDATA_INSTANCE_STRUCT_PLUS (页 2776)
- MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (页 2780)

成员

szText

文本

所需文件

CCMSRTCLIPlus.h

API 函数

MSRTCreateMsgInstanceWithCommentPlus (页 2808)	创建报警
MSRTGetLastMsgWithCommentPlus (页 2821)	查询最后一个未决报警
MSG_SERVICE_NOTIFY_PROC_PLUS (页 2798)	发送和接收服务（回调）

3.10.2.16 MSGULONGLONG

声明

```
typedef union tagMSGULONGLONG {
#ifndef _CISS_
  struct
  {
    DWORD     dwLow;
    DWORD     dwHigh;
  };
#endif
  struct
  {
    DWORD     dwLow;
    DWORD     dwHigh;
  } dwdw;
#ifndef _CISS_
  ULONGLONG   ull;
  Currency    cy; //for COLEVariant transfers
#endif
}MSGULONGLONG;
```

成员

dwLow

64 位值的最低有效 DWORD。结合 dwLow 和 dwHigh 两个参数可得出 64 位报警编号。

dwHigh

64 位报警编号的最高有效 DWORD。结合 dwLow 和 dwHigh 两个参数可得出 64 位报警编号。

dwdw.Low

请参见 dwLow

dwdw.dwHigh

请参见 dwHigh

ull

无符号 64 位值

cy

用于 COLEVariant 处理的 CURRENCY

所需文件

CCMSRTCLIPlus.h

API 函数

MSG_FILTER_STRUCT_PLUS (页 2765)	
MSG_RTDATA_INSTANCE_STRUCT_PLUS (页 2776)	
MSG__RTDATA_INSTANCECOMMENT_STRUCT_PLUS (页 2780)	

3.10.3 常规函数

3.10.3.1 MSRTStartMsgServicePlus

说明

启动发送和接收报警系统中的通知的服务。

声明

```
BOOL MSRTStartMsgServicePlus (
    LPDWORD                          lpdwServiceID,
    LPCWSTR                          lpszServer,
    DWORD                            dwFlags,
    DWORD                            dwFilterCount
    LPMSG_FILTER_STRUCT_PLUS*        lppMsgFilterPlus,
    DWORD                            dwNotifyMask,
    MSG_SERVICE_NOTIFY_PROC_PLUS     lpfnNotifyProcPlus
    LPVOID                           lpvUserNotify,
    MSG_TAID_COMPLETION_PROC_PLUS    lpfnTAIDProcPlus
    LPVOID                           lpvUserTAID
    LPCMN_ERRORW                     lpError );
```

参数

lpdwServiceID

成功调用函数后，此参数包含很多函数需要使用的服务 ID。

lpszServer

指向报警服务器符号名称（不带服务器定界符 ::）的指针
如果指定空字符串、NULL 或“@default”，则将使用设置的默认服务器。

如果指定的服务器名称不正确，则会使用组态的标准服务器，或者返回 MSG_ERR_API_SERVICE 错误。

dwFlags

该参数为将来的改进预留，必须预设为 0L。

dwFilterCount

指针列表 lppMsgFilterPlus 中的消息过滤器数。

lppMsgFilter

指向报警过滤器结构指针列表的指针 MSG_FILTER_STRUCT_PLUS (页 2765)。

该列表必须至少包含 dwFilterCount 中所指定的条目数。如果 dwFilterCount 等于 0，则 lppMsgFilterPlus 应被设为零。

dwNotifyMask

指定通知的种类（遵循“msrtapi.h”文件中的常量定义）：

MSG_NOTIFY_MASK_MSGLIST	有关报警的通知
MSG_NOTIFY_MASK_LOCK	有关锁定的通知
MSG_NOTIFY_MASK_ARCHIV	有关已记录报警的通知
MSG_NOTIFY_MASK_ALL	所有类型的通知
MSG_TIMESTAMP_LOCAL	时基“LOCALTIME”
MSG_TIMESTAMP_UTC	时基“UTC”

lpfnNotifyProcPlus

指向用于按服务传送报警的 MSG_SERVICE_NOTIFY_PROCPlus 类型回调函数的指针。

如果 lpfnNotifyProcPlus = NULL，则服务不返回任何通知。

说明

如果程序声明了“通知”例程，必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。
在极少数情况下，在函数调用返回之前已经传送了“通知”。

lpvUserNotify

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpfnTAIDProcPlus

用于传送异步函数执行状态的回调函数。
调用会标记各自的 TAID (TransactionID)，这些 TAID 是由各自的异步函数调用分配的。这些 TAID 可实现反馈消息的分配。

lpvUserTAID

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

服务已启动。

FALSE

错误

注释

应用程序最多可安装 16 个服务。总共最多可安装 128 个服务。

错误消息

MSG_ERR_RT_NOTCONNECTED	未与报警系统建立连接
MSG_ERR_RT_NOINIT	报警系统未初始化
MSG_ERR_RT_SERVICEMAX	达到最大服务数
MSG_ERR_NOSERVER	未指定默认服务器，且无可用本地服务器

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStopMsgServicePlus (页 2795)	停止服务
MSG_SERVICE_NOTIFY_PROC_PLUS (页 2798)	回调函数（数据）
MSG_TAID_COMPLETION_PROC_PLUS (页 2802)	回调函数，包含 TAID 的异步反馈消息

参见

MSRTCreateMsgInstanceWithCommentPlus (页 2808)

MSRTCreateMsgInstancePlus (页 2810)

MSRTExportPlus (页 2869)

MSRTEnumBackupListPlus (页 2867)

MSRTGetInfotextPlus (页 2864)

MSRTSetInfotextPlus (页 2865)

MSRTGetClassInfoPlus (页 2819)

MSRTSetMsgFilterPlus (页 2854)

MSG_FILTER_STRUCT_PLUS (页 2765)

MSRTGetMsgCSDataPlus (页 2827)

MSRTCheckWinFilterPlus (页 2852)

MSRTGetMsgPriorityPlus (页 2836)

MSRTGetMsgQuitPlus (页 2838)

MSRTGetSelectedMsgPlus (页 2839)

MSRTResetMsgPlus (页 2843)

MSRTLoopInAlarmPlus (页 2842)

MSRTGetFilterDataPlus (页 2850)

MSRTEnumArchivDataPlus (页 2870)

MSRTStopMsgServicePlus (页 2795)

MSRTSetCommentInstancePlus (页 2861)

MSG_SERVICE_NOTIFY_PROC_PLUS (页 2798)

MSRTQuitHornPlus (页 2797)

MSRTGetCommentInstancePlus (页 2860)

MSRTEnumGroupMsgPlus (页 2845)

MSRTLockGroupPlus (页 2847)

MSRTGetMsgActualPlus (页 2825)

函数概览 (页 2731)

MSRTGetMsgHelptextsPlus (页 2834)

3.10.3.2 MSRTStopMsgServicePlus

说明

停止用于发送和接收消息的服务

声明

```
BOOL MSRTStopMsgServicePlus (
    DWORD          dwServiceID,
    LPCMN_ERRORW   lpError );
```

参数

dwServiceID

当调用 MSRTStartMsgServicePlus 时返回的发送和接收服务的标识编号。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

服务已结束。

FALSE

错误

注释

说明

不应在应用程序析构函数（EXE、DLL、OCX...）中使用此调用，因为 Microsoft 特定的机制有时会导致调用挂起，进而导致程序挂起。

错误消息

MSG_ERR_RT_NOTCONNECTED	未与报警系统建立连接
MSG_ERR_RT_NOINIT	报警系统未初始化

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSRTEnumBackupListPlus (页 2867)

MSRTStartMsgServicePlus (页 2791)

函数概览 (页 2731)

3.10.3.3 MSRTQuitHornPlus

说明

类别特定的中央报警器确认。

声明

```
BOOL MSRTQuitHornPlus(
    DWORD          dwServiceID,
    LPDWORD      lpdwTAID,
    LPCMN_ERRORW    lpError );
```

参数

dwServiceID

当调用 MSRTStartMsgServicePlus 时返回的发送和接收服务的标识编号。

lpdwTAID

指向可存储异步状态反馈消息 TAID (TransactionID) 的 DWORD 的指针。如果传送的是 NULL，则此次调用不会调用 MSG_TAID_COMPLETION_PROC_PLUS 通知。

lpError

指向 CMN_ERRORW 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

报警已确认。

FALSE

错误

错误消息

MSG_ERR_API_PARAM	参数无效

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSRTStartMsgServicePlus (页 2791)

函数概览 (页 2731)

3.10.3.4 MSG_SERVICE_NOTIFY_PROC_PLUS

说明

发送/接收服务的回调函数。

报警系统的枚举函数异步运行。因此，必须使用信号量同步调用。

声明

```
BOOL ( * MSG_SERVICE_NOTIFY_PROC) (
    DWORD     dwNotify,
    LPBYTE    lpbyData,
    DWORD     dwItems,
    LPVOID    lpvUserNotify);
```

参数

dwNotify

通过"MSRTAPI.h"文件中的常量指定通知类型：

常规通知		lpbyData 指向结构
MSG_NOTIFY_CONNECT	已建立连接	-
MSG_NOTIFY_DISCONNECT	连接被终止	-
MSG_NOTIFY_RESTART	新报警系统运行系统数据	-
MSG_NOTIFY_SHUTDOWN	将停止报警系统	-
MSG_NOTIFY_MSGENUM	报警列表的枚举	MSG_RTDATA_INSTANCE_STRUCT_PLUS（以块为单位）
MSG_NOTIFY_ARCHIVENUM	日志报警的枚举	MSG_RTDATA_INSTANCE_STRUCT_PLUS（以块为单位）
MSG_NOTIFY_ARCHIVENUMINST	具有实例的日志报警的枚举	MSG_RTDATA_INSTANCE_STRUCT_PLUS （以块为单位）
MSG_NOTIFY_LOCKENUM	锁定的枚举	MSG_RTGROUPENUM_STRUCT_PLUS（以块为单位）
MSG_NOTIFY_WINENUM	报警窗口枚举	MSG_WIN_STRUCT_PLUS（以块为单位）
MSG_NOTIFY_BACKUPENUM	导出列表的枚举	MSG_BACKUP_STRUCT_PLUS（以块为单位）
MSG_NOTIFY_ARCDAYENUM	日历列表的枚举	MSG_ARCHIVDAY_STRUCT_PLUS
MSG_NOTIFY_ENUMMAX	达到最大数量时停止枚举	-
MSG_NOTIFY_LANGUAGESET	语言已切换	-

单个报警		lpbyData 指向结构
MSG_NOTIFY_MSGADD	已添加新报警	MSG_RTDATA_INSTANCE_STRUCT_PLUS（以块为单位）
MSG_NOTIFY_MSGDELETE	报警已删除	MSG_RTDATA_INSTANCE_STRUCT_PLUS（以块为单位）

单个报警		lpbyData 指向结构
MSG_NOTIFY_MSGMODIFY	报警已更改	MSG_RTDATA_INSTANCE_STRUCT_PLUS（以块为单位）
MSG_NOTIFY_MSGCOUNT	当前报警数目	MSG_RTDATA_INSTANCE_STRUCT_PLUS（以块为单位）

报警组		lpbyData 指向结构
MSG_NOTIFY_GROUPENUM	报警组中各个报警的枚举	MSG_RTGROUPENUM_STRUCT_PLUS（以块为单位）

报警的锁定		lpbyData 指向结构
MSG_NOTIFY_LOCKMODIFY	已设置、更改、删除锁定	MSG_RTGROUPENUM_STRUCT_PLUS（以块为单位）

日志报警		lpbyData 指向结构
MSG_NOTIFY_ARCHIVADD	已添加日志报警	MSG_RTDATA_INSTANCE_STRUCT_PLUS（以块为单位）
MSG_NOTIFY_ARCHIVDELETE	日志报警已删除	MSG_RTDATA_INSTANCE_STRUCT_PLUS（以块为单位）

lpbyData

lpbyData 包含根据通知类型指向数据的指针。

常规通知	无数据
单个报警	存储在结构中的运行系统数据，结构类型为 • MSG_RTDATA_INSTANCE_STRUCT_PLUS (页 2776)
报警组	结构中的单个报警数据，结构类型为 • MSG_RTGROUPENUM_STRUCT_PLUS (页 2784)
锁定	结构中的锁定数据，结构类型为 • MSG_RTLOCK_STRUCT_PLUS (页 2787)
日志报警	结构中的日志数据，结构类型为 • MSG_RTDATA_INSTANCE_STRUCT_PLUS (页 2776)

dwItems

包含 lpbyData 中返回的对象数。如果 dwItems = 0，则 lpbyData = NULL。将结束通知并释放信号量。

lpvUserNotify

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

已识别并处理通知。

FALSE

错误

注释

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

如果程序声明了“通知”例程，必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

在极少数情况下，在函数调用返回之前已经传送了“通知”。

如果在脚本中将回调函数编程为项目函数，则异步调用还会导致函数在脚本上下文中不完全可用，特别是 GetTag... 和 SetTag 调用将不会工作。如果可能，则必须在其它仅通过回调触发的动作中执行处理。

所需文件

CCMSRTCLIPlus.h

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSG_RTDATA_INSTANCE_STRUCT_PLUS (页 2776)

MSG_RTGROUPENUM_STRUCT_PLUS (页 2784)

MSG_RTLOCK_STRUCT_PLUS (页 2787)

MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (页 2780)

MSRTStartMsgServicePlus (页 2791)

MSG_TEXTVAL256_STRUCT_PLUS (页 2788)

MSRTEnumGroupMsgPlus (页 2845)

函数概览 (页 2731)

3.10.3.5 MSG_TAID_COMPLETION_PROC_PLUS

说明

用于反馈异步执行状态的回调函数。

报警系统的枚举函数异步运行。因此，必须使用信号量同步调用。此处介绍的 TAID_COMPLETION_PROC 可简化异步函数之间的同步。

声明

```
BOOL ( * MSG_TAID_COMPLETITION_PROC_PLUS) (
    DWORD     dwTAID,
    DWORD     dwTAIDNotify,
    Variant*  pvTAIDError,
    LPVOID    lpvUserTAID);
```

参数

dwTAID

用于分配给已调用异步函数的 TAID (TransactionID)。

dwTAIDNotify

异步函数的状态

日志报警	值	说明
MSG_TAIDCOMPLETION_NOTIFY_SUCCEEDED	0x00000000	函数执行完毕，没有任何错误
MSG_TAIDCOMPLETION_NOTIFY_INFO_RUN	0x00000001	信息：函数仍在运行，随后将发出其它通知
MSG_TAIDCOMPLETION_NOTIFY_INFO_READY	0x00000002	信息：函数已就绪且执行完毕
MSG_TAIDCOMPLETION_NOTIFY_WARNING_RUN	0x00000010	警告：函数仍在运行，但出现了部分错误
MSG_TAIDCOMPLETION_NOTIFY_WARNING_READY	0x00000020	警告：函数已就绪且执行完毕，但出现了部分错误
MSG_TAIDCOMPLETION_NOTIFY_ERROR	0x00000100	错误：测试因错误异常终止
MSG_TAIDCOMPLETION_NOTIFY_FATALERROR	0x00000200	错误：致命错误，服务器可能存在较大问题，因此很有可能出现其它错误

pvTAIDError

指向 VARIANT 的指针。一般是包含关于状态的 TAID 消息的 VT_BSTR 类型。

lpvUserTAID

指向应用程序特定数据的指针。该指针在回调函数中重新可用。

返回值

TRUE

已识别并处理通知。

FALSE

错误

注释

说明

仅应在这里复制数据（如可能）。回调中的以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问消息循环的函数，例如：GetMessage
- 相同 DLL 中的 API 函数
- 调用其它枚举的枚举

如果程序声明了“通知”例程，必须定期清空其消息队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

在极少数情况下，在函数调用返回之前已经传送了“通知”。

如果在脚本中将回调函数编程为项目函数，则异步调用还会导致函数在脚本上下文中不完全可用，特别是 GetTag... 和 SetTag 调用将不会工作。如果可能，则必须在其它仅通过回调触发的动作中执行处理。

所需文件

CCMSRTCLIPlus.h

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSRTStartMsgServicePlus (页 2791)

3.10.3.6 MSRTWebClientPlus

说明

初始化 WebClient 的 DLL

声明

```
VOID WINAPI MSRTWebClientPlus(
    LPCWSTR         pszProjectPath,
    DWORD           dwDataLocale);
```

参数

pszProjectPath

Web 项目的路径

dwDataLocale

使用语言的本地 ID

返回值

此函数没有返回值。

3.10.4 用于协议处理的函数

3.10.4.1 MSRTActivateMProtPlus

说明

暂停或继续用于输出报警序列报告的报告服务。

声明

```
BOOL MSRTActivateMProtPlus(
    BOOL            fActive,
    LPCMN_ERROR     lpError );
```

参数

fActive

确定处理报告服务的方法：

TRUE	继续报告服务
FALSE	暂停报告服务

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

函数已成功完成。

FALSE

错误

错误消息

MSG_ERR_PROT_ACTIVE	报告服务已激活

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

3.10.4.2 MSRTPrintMProtPlus

说明

即使页面尚未填满，也打印报警序列报告中当前未决的报警。

声明

```
BOOL MSRTPrintMProtPlus(
    DWORD*          pdwLines,
    LPCMN_ERRORW    lpError );
```

参数

pdwLines

指向行数的指针。这里返回已打印行数。因此，在调用前，应给参数预分配 NULL。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已打印当前未决报警。

FALSE

错误

错误消息

MSG_ERR_NOPROT	报警序列报告未激活

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

3.10.5 用于报警处理的函数

3.10.5.1 MSRTCreateMsgInstanceWithCommentPlus

说明

生成具有指定报警编号（具有实例名称和注释）的报警。相对于其它大部分 Create 调用，关联文本值最多可包含 256 个字符 (255 + EndOfString)。当前报警列表中包括带有指定数据的报警。

声明

```
BOOL WINAPI MSRTCreateMsgInstanceWithCommentPlus(
    DWORD                                       dwServiceID,
    LPMSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS    lpMsgCreatePlus,
    LPDWORD                                     lpdwTAID,
    LPCMN_ERRORW                                lpError );
```

参数

dwServiceID

服务 ID - 服务。

lpMsgCreatePlus

指向用于创建报警的 Createstruct 的指针

针对不同的消息使用不同的时间戳。

lpdwTAID

指向异步处理反馈的 TransactionID 的指针

lpError

包含扩展错误消息的指针

返回值

TRUE

函数成功

FALSE

错误

注释

Create 函数异步执行。因此，必须使用信号量同步调用。

如果使用此调用来确认报警，则指定的时间必须是反映最接近毫秒的到达报警时间。

⚠ 警告
应用程序可使用此函数来确认报警，但不能确保设备操作员已发现这些报警。此情况可能导致死亡和残疾或财产损失！

错误消息

MSG_ERR_API_PARAM	无效参数
MSG_ERR_API_SERVICE	dwServiceID 无效

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTCreateMsgInstancePlus (页 2810)	创建实例报警
MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (页 2780)

MSRTStartMsgServicePlus (页 2791)

MSRTCreateMsgInstancePlus (页 2810)

MSRTSetCommentInstancePlus (页 2861)

MSG_TEXTVAL256_STRUCT_PLUS (页 2788)

MSG_RTCREATE_STRUCT_PLUS (页 2774)

3.10.5.2 MSRTCreateMsgInstancePlus

说明

创建具有指定报警编号的报警。当前报警列表中包括带有指定数据的报警。

声明

```
BOOL WINAPI MSRTCreateMsgInstancePlus (
    DWORD                                dwServiceID,
    LPMSG_RTDATA_INSTANCE_STRUCT_PLUS    lpMsgCreatePlus,
    LPDWORD                              lpdwTAID,
    LPCMN_ERRORW                         lpError );
```

参数

dwServiceID

所安装服务的 ID

lpMsgCreatePlus

指向用于创建报警的 Createstruct 的指针

针对不同的消息使用不同的时间戳。

lpdwTAID

指向用于异步处理反馈的 TransactionID 的指针

lpError

指向扩展错误结构的指针

返回值

TRUE

函数成功

FALSE

错误

注释

Create 函数异步执行。因此，必须使用信号量同步调用。

如果使用此调用来确认报警，则指定的时间必须是反映最接近毫秒的到达报警时间。

⚠ 警告
应用程序可使用此函数来确认报警，但不能确保设备操作员已发现这些报警。此情况可能导致死亡和残疾或财产损失！

错误消息

MSG_ERR_API_PARAM	参数无效
MSG_ERR_API_SERVICE	dwServiceID 无效

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTCreateMsgInstanceWithCommentPlus (页 2808)	创建具有注释的实例报警
MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSRTCreateMsgInstanceWithCommentPlus (页 2808)

MSRTStartMsgServicePlus (页 2791)

MSG_RTDATA_INSTANCE_STRUCT_PLUS (页 2776)

MSRTSetCommentInstancePlus (页 2861)

3.10.5.3 MSRTCreateMsgPlus

说明

创建具有指定报警编号的报警。当前报警列表中包括带有指定数据的报警。

声明

```
BOOL WINAPI MSRTCreateMsgPlus(
    DWORD                          dwServiceID,
    LPMSG_RTCREATE_STRUCT_PLUS     lpMsgCreatePlus,
    LPDWORD                        lpdwTAID,
    LPCMN_ERRORW                   lpError );
```

参数

dwServiceID

所安装服务的 ID

lpMsgCreatePlus

指向用于创建报警的 Createstruct 的指针

针对不同的消息使用不同的时间戳。

lpdwTAID

指向用于异步处理的 TransactionID 的指针

lpError

指向扩展错误结构的指针

返回值

TRUE

函数成功

FALSE

错误

注释

Create 函数异步执行。因此，必须使用信号量同步调用。

⚠ 警告
应用程序可使用此函数来确认报警，但不能确保设备操作员已发现这些报警。此情况可能导致死亡和残疾或财产损失！

错误消息

MSG_ERR_API_PARAM	参数无效
MSG_ERR_API_SERVICE	dwServiceID 无效

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTCreateMsgInstanceWithCommentPlus (页 2808)	创建实例报警
MSRTCreateMsgInstancePlus (页 2810)	创建具有注释的实例报警
MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSRTCreateMsgInstanceWithCommentPlus (页 2808)

MSRTCreateMsgInstancePlus (页 2810)

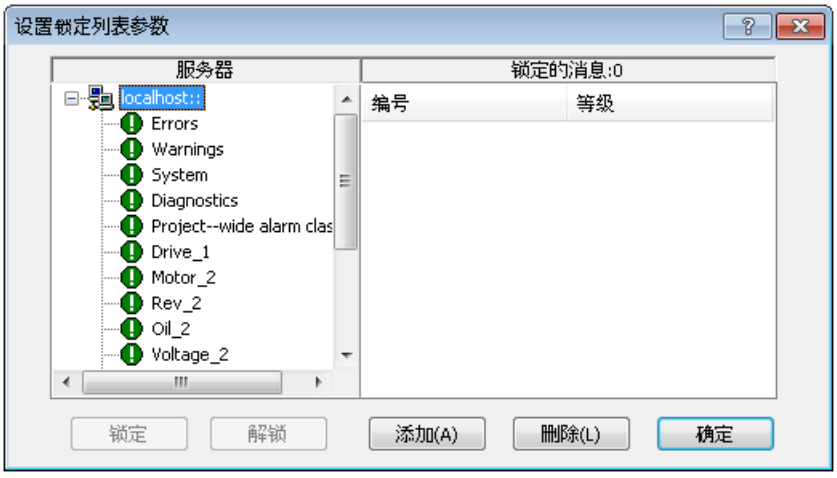
MSRTStartMsgServicePlus (页 2791)

MSG_RTCREATE_STRUCT_PLUS (页 2774)

3.10.5.4 MSRTDialogMsgLockPlus

说明

用来锁定和释放报警的对话框。



具有以下选择：

- 锁定单个报警
- 锁定报警组的报警类别
- 通过报警编号锁定报警

说明

锁定的报警：

锁定的报警既不在报警列表中输入、也不记录或报告！

发生这种情况后，锁定的报警发送回其确认信号！

重新启动 WinCC 运行系统时，系统会自动释放锁定的报警。只有在 AS 中通过数据块直接锁定的报警才继续处于锁定状态（在数据源处锁定）。

锁定的报警类别/报警组：

报警类别和报警组的锁定状态不受重新启动 WinCC 运行系统的影响。

声明

```
BOOL MSRTDialogMsgLockPlus(
    HWND           hwndParent,
    LPCMN_ERRORW   lpError );
```

参数

hwndParent

包含对话框的父窗口。

lpError

指向 CMN_ERRORW 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

对话框已成功关闭。

FALSE

出错或对话框已取消。

错误消息

MSG_ERR_API_PARAM	参数无效

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

3.10.5.5 MSRTEnumArchivInstancePlus

说明

通过实例名枚举指定日志中的所有报警。已返回 MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS 结构。

声明

```
BOOL WINAPI MSRTEnumArchivInstancePlus(
    DWORD           dwServiceID,
    BOOL            fArchiv,
    DWORD           dwMaxRecords,
    DWORD           dwParams,
    LPDWORD         lpdwTAID,
LPCMN_ERRORW        lpError );
```

参数

dwServiceID

所安装服务的 ID

fArchiv

循环或顺序日志

dwMaxRecords

枚举记录的最大数值。超出最大数值将终止枚举。

dwParams

辅助参数，MSG_ENUM_ARCHIV_DESC = 降序排列

lpdwTAID

指向用于异步处理反馈的 TransactionID 的指针

lpError

指向扩展错误结构的指针

返回值

TRUE

函数成功

FALSE

错误

3.10.5.6 MSRTEnumLockedMsgPlus

说明

枚举所有锁定的报警。

枚举函数异步执行。 因此，必须使用信号量同步调用。

声明

```
BOOL WINAPI MSRTEnumLockedMsgPlus(
    DWORD         dwServiceID,
    LPDWORD       lpdwTAID,
    LPCMN_ERRORW  lpError);
```

参数

dwServiceID

所安装服务的 ID

lpdwTAID

指向用于异步处理的 TransactionID 的指针

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。

返回值

TRUE

报警已枚举

FALSE

错误。

3.10.5.7 MSRTEnumMsgRTDataPlus

说明

枚举当前报警列表中存在的所有报警。其回调函数 MSG_SERVICE_NOTIFY_PROCPlus 通过通知类型 MSG_NOTIFY_MSGENUM（报警的枚举）来调用。lpbyData 指向 MSG_RTDATA_STRUCTPlus 结构的数据。

枚举函数异步执行。因此，必须使用信号量同步调用。

声明

```
BOOL MSRTEnumMsgRTData(
    DWORD          dwServiceID,
    LPDWORD        pdwTAID
    LPCMN_ERRORW   lpError );
```

参数

dwServiceID

当调用 MSRTStartMsgServicePlus 时返回的发送和接收服务的标识编号。

lpdwTAID

指向用于异步处理的 TransactionID 的指针。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

报警已枚举

FALSE

错误。

错误消息

MSG_ERR_API_PARAM	参数无效
MSG_ERR_API_SERVICE	dwServiceID 无效

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSRTStartMsgServicePlus (页 2791)

3.10.5.8 MSRTGetClassInfoPlus

说明

获得报警类别的相关信息。

声明

```
BOOL WINAPI MSRTGetClassInfoPlus(
    DWORD                    dwServiceID,
    LPCWSTR                  lpszServer,
    DWORD                    dwMsgNrLow,
    DWORD                    dwMsgNrHigh,
    LPMSG_CLASS_STRUCT_PLUS  lpClassPlus,
    LPCMN_ERRORW             lpError );
```

参数

dwServiceID

服务 ID - 所连接的服务器对应的服务

lpszServer

服务器前缀无 ::
如果输入空字符串或 NULL，则使用默认服务器设置。

dwMsgNrLow

DWORD 下限报警编号

dwMsgNrHigh

WinCC DWORD 上限报警编号

lpClassPlus

包含信息的指针

lpError

包含扩展错误消息的指针

返回值

TRUE

函数成功

FALSE

错误

注释

也可不通过 MSRTStartMsgServicePlus 调用 MSRTGetClassInfoPlus 函数。但是，这些调用明显会更慢一些。可以通过事先执行 MSRTStartMsgServicePlus 并为所有调用保留此函数来提高性能。在这种情况下，仅会降低首次调用 MSRTGetClassInfoPlus 的速度。

错误消息

MSG_ERR_RT_NOINIT	报警记录 RT 未初始化
MSG_ERR_API_PARAM	无效参数
MSG_ERR_MSG_NOEXIST	报警不存在

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSG_CLASS_STRUCT_PLUS (页 2755)

MSRTStartMsgServicePlus (页 2791)

3.10.5.9 MSRTGetLastMsgWithCommentPlus

说明

从基于过滤条件和实例名称的日志检索最后一个报警。

声明

```
BOOL WINAPI MSRTGetLastMsgWithCommentPlus(
    LPCWSTR                                  lpszServer,
    LPCWSTR                                  lpszInstance,
    LPMSG_FILTER_STRUCT_PLUS*                lppFilter,
    DWORD                                    dwFilterCount,
    LPMSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS  lpMsg,
    LPCMN_ERRORW                             lpError );
```

参数

lpszServer

服务器前缀（不带"::"）。如果输入空字符串或 NULL，则使用默认服务器设置。

lpszInstance

实例名称。针对该过滤条件，将添加空字符串或 NULL。

lppFilter

指向用于传送过滤条件的 MSG_FILTER_STRUCT_PLUS 结构指针列表的指针。

dwFilterCount

通过 lppFilter 指定的过滤结构数目至少须指定一个过滤结构。

lpMsg

返回消息数据。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。

返回值

TRUE

报警已查询。

FALSE

错误。

错误消息

MSG_ERR_API_PARAM	无效参数
MSG_ERR_RT_NOTCONNECTED	未进行连接

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTGetMsgAttributesPlus (页 2823)	查询消息属性

参见

MSG_FILTER_STRUCT_PLUS (页 2765)

MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (页 2780)

MSG_TEXTVAL256_STRUCT_PLUS (页 2788)

3.10.5.10 MSRTGetMsgAttributesPlus

说明

基于过滤条件、实例名称和 NotifyMask，从单个消息、锁定列表、归档、统计列表或隐藏列表中检索最后一条消息的属性。

声明

```
BOOL WINAPI MSRTGetMsgAttributesPlus(
    LPCWSTR                                 lpszServer,
    LPCWSTR                                 lpszInstance,
    LPMSG_FILTER_STRUCT_PLUS*               lppFilter,
    DWORD                                   dwFilterCount,
    DWORD                                   dwNotifyMask,
    LPDWORD                                 lpdwAttributeList,
    DWORD                                   dwAttributeListCount,
    VARIANT*                                lpvAttributeValues,
    LPCMN_ERRORW                            lpError );
```

参数

lpszServer

服务器前缀无 ::
如果输入空字符串或 NULL，则使用默认服务器设置。

lpszInstance

实例名称。针对该过滤条件，将添加空字符串或 NULL。

lppFilter

指向用于传送过滤条件的 MSG_FILTER_STRUCT_PLUS 结构指针列表的指针。

由于此函数始终仅允许查询一个报警，dwMsgNrLow 和 dwMsgNrHigh 必须包含要查询的报警的编号。

dwFilterCount

通过 lppFilter 指定的过滤结构数目至少须指定一个过滤结构。

dwNotifyMask

通过"CCMsRtCliPlus.h"并使用文件"msrtapi.h"中的以下常量，定义消息所属的消息类型的相关信息：

MSG_NOTIFY_MASK_MSGLIST	单个消息通知
MSG_NOTIFY_MASK_LOCK	锁定消息
MSG_NOTIFY_MASK_ARCHIV	归档消息
MSG_NOTIFY_MASK_HITLIST	来自统计列表
MSG_NOTIFY_MASK_HIDE	
MSG_TIMESTAMP_LOCAL	过滤条件中的时间规范并返回当地时间
MSG_TIMESTAMP_UTC	过滤条件中的时间规范并返回 UTC 时间

只能使用可通过“或”函数与 MSG_TIMESTAMP_xxx 相关联的指定函数 MSG_NOTIFY_MASK_xxx。

lpdwAttributeList

指向变量 dwAttributeListCount 的 DWORD 数组的指针（其中要查询的消息属性必须指定）。至少须指定一个属性。

通过"CCMsRtCliPlus.h"并使用"msrtapi.h"文件中的常量 MSG_ATTR_xxx，指定可能的属性。

dwAttributeListCount

指定在列表 lpdwAttributeList 中指定的属性数目。至少须指定一个属性。

lpvAttributeValues

指向必须使用 VT_EMPTY 进行初始化的 VARIANT 的指针。此处将返回所请求属性的一维 (VT_ARRAY|VT_VARIANT)。

前几个值始终被分配为消息编号（索引 0）、时间戳（索引 1）、状态（索引 2）和实例（索引 3）。所请求的属性存储在数组的索引 4 中。更多相关信息，请参见"m_global.h"中的偏移量定义 STARTINDEX_MSG_ATTR。

如果 VARIANT 保持为空且函数未报告错误，则通过过滤条件不会找到任何结果。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。

返回值

TRUE

已查询消息数据。

FALSE

错误。

错误消息

MSG_ERR_API_PARAM	无效参数
MSG_ERR_RT_NOTCONNECTED	未进行连接

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTGetLastMsgWithCommentPlus (页 2821)	查询最后一个报警

参见

MSG_FILTER_STRUCT_PLUS (页 2765)

3.10.5.11 MSRTGetMsgActualPlus

说明

该函数说明报警列表中当前未决报警的数量。

声明

```
BOOL MSRTGetMsgActualPlus (
    LPDWORD        lpdwCount,
    LPCMN_ERRORW    lpError );
```

参数

lpdwCount

指向存储当前未决报警数的缓冲区的指针。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

数量已说明。

FALSE

错误

注释

也可不通过 MSRTStartMsgServicePlus 调用 MSRTGetMsgActualPlus 函数。但是，这些调用明显会更慢一些。可以通过事先执行 MSRTStartMsgServicePlus 并为所有调用保留此函数来提高性能。在这种情况下，仅会降低首次调用 MSRTGetMsgActualPlus 的速度。

错误消息

MSG_ERR_RT_NOINIT	报警系统未初始化

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSRTStartMsgServicePlus (页 2791)

3.10.5.12 MSRTGetMsgCSDataPlus

说明

查询报警的组态数据。

声明

```
BOOL MSRTGetMsgCSDataPlus (
    DWORD                       dwServiceNr,
    LPCWSTR                     lpszServer,
    DWORD                       dwMsgNrLow,
    DWORD                       dwMsgNrHigh,
    LPMSG_CSDATA_STRUCT_PLUS    lpmCSDataPlus,
    LPCMN_ERRORW                lpError );
```

参数

dwServiceID

当调用 MSRTStartMsgServicePlus 时返回的发送和接收服务的标识编号。

lpszServer

服务器前缀无 ::
如果输入空字符串或 NULL，则使用默认服务器设置。

dwMsgNrLow

第一个报警的编号

dwMsgNrHigh

最后一个报警的编号

lpmCSDataPlus

指向 MSG_CSDATA_STRUCT_PLUS (页 2759) 结构中针对此报警的组态数据（报警级别、报警类型、文本块索引等）的指针。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。如果发生错误，则系统将向该结构写入错误信息。

返回值

TRUE

组态数据已查询。

FALSE

错误

注释

也可不通过 MSRTStartMsgServicePLus 调用 MSRTGetMsgCSDataPlus 函数。但是，这些调用明显会更慢一些。可以通过事先执行 MSRTStartMsgServicePlus 并为所有调用保留此函数来提高性能。在这种情况下，仅会降低首次调用 MSRTGetMsgCSDataPlus 的速度。

错误消息

MSG_ERR_RT_NOINIT	报警系统未初始化
MSG_ERR_API_PARAM	参数无效
MSG_ERR_MSG_NOEXIST	报警不存在

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

示例

以下示例展示了如何在 C 函数中嵌入函数调用。

```
#include "GlobalDefinitions.h"
void OnClick(char* screenName, char* objectName, char* propertyName)
{
//add your Code hiere
#pragma code("CCMSRTCLIPlus.dll")
#include "CCMSRTCLIPlus.h"
#pragma code()
BOOL ret = FALSE;
CMN_ERRORW Error;
// DWORD dwMsgNrU;
// DWORD dwMsgNrO;
int i;
WCHAR lpszMsgText[255];
DWORD lpdwCount = 255;
MSG_CSDATA_STRUCT_PLUS msgCSData;
MSGULONGLONG msgID;
msgID.dwdw.dwLow  = 1;
msgID.dwdw.dwHigh = 0;
memset( &msgCSData, 0, sizeof( msgCSData ));
// dwMsgNrU = 1;
// dwMsgNrO = 0;
ret = MSRTGetMsgCSDataPlus(0L, NULL, msgID.dwdw.dwLow, msgID.dwdw.dwHigh,
&msgCSData, &Error );
if(TRUE==ret)
  {
  for (i = 0; i < (sizeof(msgCSData.dwTextID)/sizeof(DWORD)); i++){
    printf("TextID[%d] = %d,\r\n",i, msgCSData.dwTextID[i]);
    }
  }
else
  {
  printf("Error = %ls\r\t", Error.szErrorText);
  }
}
```

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSG_CSDATA_STRUCT_PLUS (页 2759)

MSRTStartMsgServicePlus (页 2791)

MSGULONGLONG (页 2790)

3.10.5.13 MSRTGetMsgCSDataExPlus

说明

查询指定报警服务器中报警的组态数据。此函数返回的变量名称包含 MSG_CSDATA_EX_STRUCT_PLUS 结构中的服务器前缀。

声明

```
BOOL MSRTGetMsgCSDataExPlus (
    DWORD                       dwServiceNr,
    LPCWSTR                     lpszServer,
    DWORD                       lpdwMsgNrLow,
    DWORD                       lpdwMsgNrHigh,
    DWORD                       dwMsgNrCount,
    MSG_CSDATA_EX_CALLBACK_PROC_PLUS    lpfnCSDataCallbackPlus,
    LPVOID                      lpvUser,
    LPCMN_ERRORW                lpError );
```

参数

dwServiceID

当调用 MSRTStartMsgServicePlus 时返回的发送和接收服务的标识编号。

lpszServer

指向报警服务器符号名称（不带服务器定界符 ::）的指针

在此指定的报警服务器仅用于提供的 dwServiceID 为 0L 的情况。如果指定空字符串或 NULL，则将使用设置的默认服务器。

lpdwMsgNrLow

指向包含 64 位报警编号的最低有效 DWORD 的 DWORD 列表的指针。

lpdwMsgNrHigh

指向包含 64 位报警编号的最高有效 DWORD 的 DWORD 列表的指针（通常被 0L 占用）。

dwMsgNrCount

列表中报警编号的数量

lpfnCSDataCallbackPlus

接收报警组态数据的回调函数。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。如果发生错误，则系统将向该结构写入错误信息。

返回值

TRUE

组态数据已查询。

FALSE

错误

注释

也可不通过 MSRTStartMsgServicePLus 调用 MSRTGetMsgCSDataPlus 函数。但是，这些调用明显会更慢一些。可以通过事先执行 MSRTStartMsgServicePlus 并为所有调用保留此函数来提高性能。在这种情况下，仅会降低首次调用 MSRTGetMsgCSDataPlus 的速度。

错误消息

MSG_ERR_RT_NOINIT	报警系统未初始化
MSG_ERR_API_PARAM	参数无效
MSG_ERR_MSG_NOEXIST	报警不存在

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

示例

以下示例展示了如何在 C 函数中嵌入函数调用。

```
#include "GlobalDefinitions.h"
void OnClick(char* screenName, char* objectName, char* propertyName)
{
//add your Code hiere
#pragma code("CCMSRTCLIPlus.dll")
#include "CCMSRTCLIPlus.h"
#pragma code()
BOOL ret = FALSE;
CMN_ERRORW Error;
// DWORD dwMsgNrU;
// DWORD dwMsgNrO;
int i;
WCHAR lpszMsgText[255];
DWORD lpdwCount = 255;
MSG_CSDATA_STRUCT_PLUS msgCSData;
MSGULONGLONG msgID;
msgID.dwdw.dwLow  = 1;
msgID.dwdw.dwHigh = 0;
memset( &msgCSData, 0, sizeof( msgCSData ));
// dwMsgNrU = 1;
// dwMsgNrO = 0;
ret = MSRTGetMsgCSDataPlus(0L, NULL, msgID.dwdw.dwLow, msgID.dwdw.dwHigh,
&msgCSData, &Error );
if(TRUE==ret)
  {
  for (i = 0; i < (sizeof(msgCSData.dwTextID)/sizeof(DWORD)); i++){
    printf("TextID[%d] = %d,\r\n",i, msgCSData.dwTextID[i]);
    }
  }
else
  {
  printf("Error = %ls\r\t", Error.szErrorText);
  }
}
```

相关函数

MSG_CSDATA_EX_CALLBACK_PROC_PLUS (页 2833)	回调函数
MSG_CSDATA_EX_STRUCT_PLUS (页 2762)	组态数据的结构

3.10.5.14 MSG_CSDATA_EX_CALLBACK_PROC_PLUS

说明

为了评估系统列出的报警组态数据，必须提供 MSG_CSDATA_EX_CALLBACK_PROC_PLUS 类型的回调函数。

声明

```
BOOL ( * MSG_CSDATA_EX_CALLBACK_PROC_PLUS) (
    DWORD                       dwIndex,
    LPMSG_CSDATA_EX_STRUCT_PLUS    lpmCSDataExPlus,
    LPVOID                      lpvUser );
```

参数

dwIndex

当调用 MSRTStartMsgServicePlus 时返回的发送和接收服务的标识编号。

lpmsgCSDataExPlus

指向具有报警组态数据的 MSG_CSDATA_EX_STRUCT_PLUS (页 2762) 类型结构的指针。

lpvUser

指向应用程序特定数据的指针。函数不会评估该指针，但在回调函数中会重新提供。

注释

仅应在此处复制数据（如可能）。回调中以下类型的函数调用可能导致死锁或堆栈溢出：

- 在其中访问报警循环的函数，例如：GetMessage
- 调用其它枚举的枚举

如果程序注册通知例程，则必须定期清空其报警队列。未读取的消息可阻塞 WinCC 通知，或者阻塞整个 WinCC。

指向结构中变量名称的指针并非必须启用。回调返回后，会在内部自动启用此指针。因此，指针本身不能复制，因为它们在回调或函数执行结束后是无效的。如有需要，必须将变量名称作为字符串复制。

返回值

TRUE

继续枚举。

FALSE

取消枚举。

所需文件

CCMSRTCLIPlus.h

相关函数

MSRTGetMsgCSDataExPlus (页 2830)	查询报警的组态数据

3.10.5.15 MSRTGetMsgHelptextsPlus

说明

函数会确定为指定报警服务器中的报警组态的帮助文本。

声明

```
BOOL MSRTGetMsgHelptextsPlus(
    DWORD                           dwServiceID;
    LPCWSTR                         lpszServer,
    LPMSG_HELPTEXTS_STRUCT_PLUS     lpHelptexts,
    LPCMN_ERROR                     lpError );
```

参数

dwServiceID

服务 ID、“MSRTStartMsgServicePlus”或“0L”。

如果指定了服务 ID，则此 ID 优先级较高，并会使用关联“MSRTStartMsgServicePlus”的相应报警服务器。

lpszServer

指向报警服务器符号名称（不带服务器定界符 ::、“NULL”或空字符串）的指针。

在此指定的报警服务器仅用于提供的“dwServiceID”为“0L”的情况。

lpHelptexts

指向返回帮助文本“ResponseTime”、“Description”、“Causes”、“Action”、“Consequence”的 MSG_HELPTEXTS_STRUCT_PLUS (页 2769) 类型结构的指针。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

已查询组态的帮助文本。

FALSE

错误

注释

也可不通过 MSRTStartMsgServicePlus 调用 MSRTGetMsgHelptextsPlus 函数。但是，这些调用明显会更慢一些。可以通过事先执行 MSRTStartMsgServicePlus 并为所有调用保留此函数来提高性能。在这种情况下，仅会降低首次调用 MSRTGetMsgPriorityPlus 的速度。

可选择为每个帮助文本指定自己的固定或已分配文本缓冲区，帮助文本会存储在这些缓冲区中。随后必须指定“lpszHelp_xxx”和“dwHelp_xxx”。如果为“lpszHelp_xxx”指定“NULL”，并且/或者为“dwHelp_xxx”指定“0L”，“MSRTGetMsgHelptextsPlus”函数会自行分配文本缓冲区，还会在“lpHelptexts”结构中输入该分配值。调用后，可使用“dwFlags”对此进行评估

错误消息

MSG_ERR_API_PARAM	无效参数
MSG_ERR_MSG_NOEXIST	运行系统未初始化
MSG_ERR_RT_NOINIT	报警不存在

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

3.10.5.16 MSRTGetMsgPriorityPlus

说明

查询给定报警的优先级。

声明

```
BOOL MSRTGetMsgPriorityPlus (
    DWORD                   dwServiceID,
    LPCWSTR                 lpszServer,
    DWORD                   dwMsgNrLow,
    DWORD                   dwMsgNrHigh,
    long*                   plPriority,
    LPCMN_ERRORW            lpError );
```

参数

dwServiceID

当调用 MSRTStartMsgService 时返回的发送和接收服务的标识编号。

lpszServer

服务器前缀无 ::
如果输入空字符串或 NULL，则使用默认服务器设置。

dwMsgNrLow

dwMsgNrLow 第一个报警的编号

dwMsgNrHigh

dwMsgHigh 最后一个报警的编号

plPriority

指向存储优先级的 long 数据类型变量的指针。

此参数不得为 NULL！

lpError

指向 CMN_ERRORW 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

优先级已查询。

FALSE

错误

注释

也可不通过 MSRTStartMsgServicePlus 调用 MSRTGetMsgPriorityPlus 函数。但是，这些调用明显会更慢一些。可以通过事先执行 MSRTStartMsgServicePlus 并为所有调用保留此函数来提高性能。在这种情况下，仅会降低首次调用 MSRTGetMsgPriorityPlus 的速度。

错误消息

MSG_ERR_RT_NOINIT	报警系统未初始化
MSG_ERR_API_PARAM	参数无效
MSG_ERR_MSG_NOEXIST	报警不存在

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSRTStartMsgServicePlus (页 2791)

3.10.5.17 MSRTGetMsgQuitPlus

说明

查询需要确认的报警数。

声明

```
BOOL MSRTGetMsgQuitPlus(
    LPDWORD        lpdwCount,
    LPCMN_ERRORW   lpError );
```

参数

lpdwCount

指向需要确认的报警数的指针

lpError

指向 CMN_ERRORW 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

数量已说明。

FALSE

错误

注释

也可不通过 MSRTStartMsgServicePlus 调用 MSRTGetMsgPriorityPlus 函数。但是，这些调用明显会更慢一些。可以通过事先执行 MSRTStartMsgServicePlus 并为所有调用保留此函数来提高性能。在这种情况下，仅会降低首次调用 MSRTGetMsgPriorityPlus 的速度。

错误消息

MSG_ERR_RT_NOINIT	报警系统未初始化

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSRTStartMsgServicePlus (页 2791)

3.10.5.18 MSRTGetSelectedMsgPlus

说明

此函数确定在报警窗口或报警视图中所选报警的报警编号。使用 lpszTemplate 来指定视图。

在单站或多站项目中，查询使用此模板不受限制。显示中的查询不适用于工程数据，只适用于显示的现有过程值。

声明

```
BOOL MSRTGetSelectedMsgPlus(
    LPCWSTR                                      lpszTemplate,
    LPMSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS     lpMsgRTPlus,
    LPCMN_ERROR                                  lpError );
```

参数

lpszTemplate

指向在其中选择报警的视图名称的指针。

lpMsgRTPlus

指向返回报警编号的 MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (页 2780) 类型结构的指针。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

数据已查询。

FALSE

错误

注释

也可不通过 MSRTStartMsgServicePlus 调用 MSRTGetMsgPriorityPlus 函数。但是，这些调用明显会更慢一些。可以通过事先执行 MSRTStartMsgServicePlus 并为所有调用保留此函数来提高性能。在这种情况下，仅会降低首次调用 MSRTGetMsgPriorityPlus 的速度。

说明

根据在显示中设置的视图，MSG_RTDATA_INSTANCECOMMENT_STRUCTPlus 结构的 dwFlags 中将分配不同的标记：

- 用于在线视图：MSG_RT_FLAG_xxx 标记，
- 用于日志视图：MSG_FLAG_ 标记，以及
- 用于统计列表：不分配并且将忽略标记。

错误消息

MSG_ERR_API_PARAM	无效参数

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

示例

Get selected Msg "MS02.cpp"

参见

MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (页 2780)

MSRTStartMsgServicePlus (页 2791)

3.10.5.19 MSRTLoopInAlarmPlus

说明

报警循环的 API 函数。报警编号是确定的。

声明

```
BOOL MSRTLoopInAlarmPlus(
    DWORD          dwServiceID,
    LPCWSTR        lpszServer,
    DWORD          dwMsgNrLow,
    DWORD          dwMsgNrHigh,
    LPCMN_ERRORW   lpError );
```

参数

dwServiceID

服务 ID - 所连接的服务器对应的服务

lpszServer

服务器前缀无 ::
如果输入空字符串或 NULL，则使用默认服务器设置。

dwMsgNrLow

DWORD 下限报警编号

dwMsgNrHigh

DWORD WinCC 上限的报警编号

lpError

包含扩展错误消息的指针

返回值

TRUE

函数成功

FALSE

错误

错误消息

MSG_ERR_API_PARAM	无效参数
MSG_ERR_MSG_NOEXIST	报警不存在

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSRTStartMsgServicePlus (页 2791)

3.10.5.20 MSRTResetMsgPlus

说明

将确认已说明的报警。

声明

```
BOOL MSRTResetMsgPlus(
    DWORD          dwServiceID,
    DWORD          dwMsgNrLow,
    DWORD          dwMsgNrHigh,
    LPDWORD        lpdwTAID
    LPCMN_ERRORW   lpError );
```

参数

dwServiceID

当调用 MSRTStartMsgService 时返回的发送和接收服务的标识编号。

dwMsgNrLow

dwMsgNrLow 第一个报警的编号

dwMsgNrHigh

dwMsgHigh 最后一个报警的编号

如果您使用 MSG_FILTER_NR、MSG_FILTER_NR_FROM 或 MSG_FILTER_NR_TO 过滤条件，则必须分配 dwMsgNrLow 和 dwMsgNrHigh。

lpdwTAID

指向用于异步处理的 TransactionID 的指针。

lpError

指向 CMN_ERRORW 结构中的扩展错误消息数据的指针。 出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

报警已确认。

FALSE

错误

注释

⚠ 警告
应用程序可使用此函数来确认报警，但不能确保设备操作员已发现这些报警。 此情况可能导致死亡和残疾或财产损失！

错误消息

MSG_ERR_RT_NOINIT	报警系统未初始化
MSG_ERR_API_PARAM	参数无效

MSG_ERR_MSG_NOEXIST	报警不存在
MSG_ERR_MSG_NOTFOUND	未找到报警
MSG_ERR_MSG_NOQUIT	报警无需确认
MSG_ERR_MSG_ALREADYQUIT	报警已确认

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSRTStartMsgServicePlus (页 2791)

3.10.6 用于报警组处理的函数

3.10.6.1 MSRTEnumGroupMsgPlus

说明

枚举报警组的所有报警。MSG_MAX_GROUPITEMS 设置可枚举的单个报警的最大数量。使用 MSG_NOTIFY_GROUPENUM 通知类型调用 MSG_SERVICE_NOTIFY_PROCPlus 其回调函数。lpbyData 指向 MSG_RTGROUPENUM_STRUCTPlus 结构的数据。

报警系统的枚举函数异步运行。因此，必须使用信号量同步调用。

声明

```
BOOL WINAPI MSRTEnumGroupMsgPlus(
     DWORD                              dwServiceID,
     LPMSG_RTGROUPSET_STRUCT_PLUS       lpmGroupPlus,
     LPDWORD                            lpdwTAID,
     LPCMN_ERRORW                       lpError );
```

参数

dwServiceID

服务 ID - 服务。

lpmGroupPlus

指向报警组信息的指针

lpdwTAID

指向用于异步处理反馈的 TransactionID 的指针

lpError

包含扩展错误消息的指针

返回值

TRUE

函数成功

FALSE

错误

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSG_SERVICE_NOTIFY_PROC_PLUS (页 2798)	回调函数
MSRTStartMsgServicePlus (页 2791)	启动服务
MSG_RTGROUPENUM_STRUCT_PLUS (页 2784)	报警组的单个报警

参见

MSG_RTGROUPSET_STRUCT_PLUS (页 2786)

MSRTStartMsgServicePlus (页 2791)

MSG_SERVICE_NOTIFY_PROC_PLUS (页 2798)

MSG_RTGROUPENUM_STRUCT_PLUS (页 2784)

3.10.6.2 MSRTLockGroupPlus

说明

锁定组态的报警组。锁定报警组中的所有单个报警以及基础组中的单个报警。

声明

```
BOOL MSRTLockGroupPlus (
    DWORD                           dwServiceID,
    LPMSG_RTGROUPSET_STRUCTPlus     lpmGroupPlus,
    LPDWORD                         pdwTAID
    LPCMN_ERRORW                    lpError );
```

参数

dwServiceID

调用 MSRTStartMsgServicePlus 时返回的服务 ID。

lpmGroupPlus

指向具有报警组信息的 MSG_RTGROUPSET_STRUCT_PLUS (页 2786) 结构的指针。

锁定：dwData != 0；启用：dwData = 0

pdwTAID

指向用于异步处理反馈消息的事务 ID 的指针

lpError

指向 CMN_ERRORW 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

报警组中的报警已锁定。

FALSE

错误

注释

仅会锁定或释放已组态的报警组。在运行系统中定义的组不包含在内，并且此函数对其没有效果。

错误消息

MSG_ERR_RT_NOINIT	报警系统未初始化
MSG_ERR_API_PARAM	参数无效
MSG_ERR_API_SERVICE	dwServiceID 无效

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSG_RTGROUPSET_STRUCT_PLUS (页 2786)

MSRTStartMsgServicePlus (页 2791)

函数概览 (页 2731)

3.10.6.3 MSRTQuitGroupPlus

说明

确认报警组。确认报警组的所有单个报警以及从属组的报警。

声明

```
BOOL MSRTQuitGroupPlus (
    DWORD                          dwServiceID,
    LPMSG_RTGROUPSET_STRUCTPlus    lpmGroupPlus,
    LPDWORD                        pdwTAID
    LPCMN_ERRORW                   lpError );
```

参数

dwServiceID

调用 MSRTStartMsgServicePlus 时返回的服务 ID。

lpmGroupPlus

指向具有报警组信息的 MSG_RTGROUPSET_STRUCT_PLUS (页 2786) 类型结构的指针。

锁定：dwData != 0；启用：dwData = 0

pdwTAID

指向用于异步处理反馈消息的事务 ID 的指针

lpError

指向 CMN_ERRORW 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

将确认报警组中的报警。

FALSE

错误

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

参见

MSG_RTGROUPSET_STRUCT_PLUS (页 2786)

3.10.7 用于报警过滤器处理的函数

3.10.7.1 MSRTGetFilterDataPlus

说明

读取报警视图的选择条件并将结果写入 MSG_FILTER_STRUCT_PLUS 结构。

声明

```
BOOL WINAPI MSRTGetFilterDataPlus(
   DWORD            dwServiceID,
   LPCWSTR            lpszName,
   LPMSG_FILTER_STRUCT_PLUS**    lpppMsgFilter,
   LPDWORD              lpdwFilterCount,
   LPCMN_ERRORW lpError );
```

参数

dwServiceID

服务 ID - 服务。

lpszName

报警视图名称。将在报警视图属性中分配名称。

lppMsgFilterPlus

指向过滤结构的数组过滤器指针的指针

lpdwFilterCount

lpError

包含扩展错误消息的指针

返回值

TRUE

函数成功

FALSE

错误

注释

仅确定已组态的报警过滤器。使用 MSRTStartMsgServicePlus 或 MSRTSetMsgFilterPlus 设置的报警过滤器不包括在内。

错误消息

MSG_ERR_API_PARAM	无效参数
MSG_ERR_API_SERVICE	dwServiceID 无效
MSG_ERR_API_NODATA	无数据

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTSetMsgFilterPlus (页 2854)	设置报警过滤器
MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSG_FILTER_STRUCT_PLUS (页 2765)

MSRTStartMsgServicePlus (页 2791)

MSRTSetMsgFilterPlus (页 2854)

3.10.7.2 MSRTCheckWinFilterPlus

说明

如果已说明的报警对应于过滤参数，则进行测试。

声明

```
BOOL MSRTCheckWinFilterPlus  (
    DWORD                                       dwServiceID,
    LPCWSTR                                     lpszServer,
    LPMSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS    lpMsgRTPlus,
    LPMSG_FILTER_STRUCT_PLUS*                   lppMsgFilter,
    DWORD                                       dwFilterCount,
    LPWORD                                      lpwReturn,
    LPCMN_ERRORW                                lpError );
```

参数

dwServiceID

调用 MSRTStartMsgServicePlus 时返回的服务 ID。

lpszServer

服务器前缀无 ::
如果输入空字符串或 NULL，则使用默认服务器设置。

lpMsgRTPlus

指向具有报警运行系统数据的 MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (页 2780) 类型结构的指针。

lppMsgFilter

指向过滤结构上的过滤器指针数组的指针。

lpwReturn

指向函数结果的指针。

TRUE	报警对应于过滤条件。
FALSE	报警不对应于过滤条件。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

将执行测试。

FALSE

错误

错误消息

MSG_ERR_API_PARAM	参数无效

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSG_FILTER_STRUCT_PLUS (页 2765)

MSRTStartMsgServicePlus (页 2791)

MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS (页 2780)

3.10.7.3 MSRTSetMsgFilterPlus

说明

重置指定服务的过滤器。

声明

```
BOOL WINAPI MSRTSetMsgFilterPlus(
        DWORD                           dwServiceID,
        LPMSG_FILTER_STRUCT_PLUS*       lppMsgFilterPlus,
        DWORD                           dwFilterCount,
        LPCMN_ERRORW                    lpError );
```

参数

dwServiceID

当调用 MSRTStartMsgServicePlus 时返回的发送和接收服务的标识编号。

lppMsgFilterPlus

指向过滤条件的 ARRAY 指针的指针

dwFilterCount

ARRAY 中过滤器的数量

lpError

指向 CMN_ERRORW 结构中的扩展错误消息数据的指针。 出现错误时，系统会将错误信息写入该结构。

返回值

TRUE

函数成功

FALSE

错误

错误消息

MSG_ERR_API_PARAM	无效参数
MSG_ERR_API_SERVICE	dwServiceID 无效

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务
MSRTGetFilterDataPlus (页 2850)	读取报警视图的选择条件

参见

MSRTStartMsgServicePlus (页 2791)

MSG_FILTER_STRUCT_PLUS (页 2765)

MSRTGetFilterDataPlus (页 2850)

3.10.7.4 MSRTSetMsgWinFilterPlus

说明

定义报警视图的新过滤条件。将更新所有使用此名称预设的当前报警视图。

声明

```
BOOL MSRTSetMsgWinFilterPlus (
    LPMSG_FILTER_STRUCT_PLUS    lppMsgFilterPlus,
    DWORD                   dwFilterCount,
    LPCMN_ERRORW             lpError );
```

参数

lppMsgFilterPlus

指向具有报警过滤数据的 MSG_FILTER_STRUCT_PLUS (页 2765) 类型结构的指针。过滤器结构的名称与窗口名称相同。

dwFilterCount

过滤器数

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

过滤器已设置

FALSE

错误

错误消息

MSG_ERR_API_PARAM	参数无效
MSG_ERR_API_SERVICE	dwServiceID 无效

参见

MSG_FILTER_STRUCT_PLUS (页 2765)

3.10.8 用于处理报警视图的函数

3.10.8.1 MSRTMsgWinCommandPlus

说明

在报警视图中执行控制函数。如果您使用此函数，则仅可在画面中使用报警窗口模板一次。这些命令也将传递给标题名称与模板名称相同的控件。（当具有报警模板的画面转换为具有控件的画面时，生成的控件具有与模板名称相同的标题名称。）

声明

```
BOOL MSRTMsgWinCommandPlus(
    LPTSTR          lpszTemplate,
    DWORD           dwCommandID,
    LPCMN_ERROR     lpError );
```

参数

lpszTemplate

指向窗口名称的指针

dwCommandID

通过“m_global.h”文件中的常量指定控制函数的 ID：

MSG_TB_MSGWIN	打开过程报警窗口
MSG_TB_ARC_S	打开循环日志
MSG_TB_ARC_L	打开顺序日志
MSG_TB_QH	启动报警器确认
MSG_TB_QM	启动单个确认
MSG_TB_QS	启动组确认
MSG_TB_SCROLL	打开/关闭滚动功能。
MSG_TB_SELECT	显示选择对话框
MSG_TB_LOCK	显示锁定对话框
MSG_TB_PROTOCOL	显示报表对话框
MSG_TB_RESET	启动紧急确认
MSG_TB_MSGFIRST	移动到列表的开始 （不得激活自动滚动）
MSG_TB_MSGLAST	移动到列表的结束 （不得激活自动滚动）
MSG_TB_MSGNEXT	移动到下一个报警 （不得激活自动滚动）
MSG_TB_MSGPREV	移动到上一个报警 （不得激活自动滚动）
MSG_TB_INFO	显示当前所选报警的信息文本对话框

MSG_TB_COMMENT	输入当前所选报警的注释
MSG_TB_LOOPINALARM	启动当前所选报警的报警循环
MSG_TB_PRINT	启动打印

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

控制函数已执行

FALSE

错误

注释

所有具有指定名称的打开报警视图执行此控制函数。

错误消息

MSG_ERR_API_PARAM	参数无效

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

3.10.9 用于处理注释的函数

3.10.9.1 MSRTGetCommentInstancePlus

说明

获取已记录的报警的注释文本。

声明

```
BOOL MSRTGetCommentInstancePlus (
    DWORD                                dwServiceID,
    LPMSG_COMMENT_INSTANCE_STRUCT_PLUS    lpmCommentPlus,
    LPCMN_ERRORW                         lpError );
```

参数

dwServiceID

当调用 MSRTStartMsgServicePlus 时返回的发送和接收服务的标识编号。

lpmCommentPlus

指向具有注释数据的 MSG_COMMENT_INSTANCE_STRUCT_PLUS (页 2757) 类型结构的指针。结构中需要以下各值：报警编号、日期和时间以及实例名称。

说明

对于注释文本将要精确到毫秒级的消息，其时间戳必须与日期和时间精确匹配。

lpError

指向 CMN_ERROR W 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

注释文本已确定。

如果不存在注释，则调用返回 TRUE，但是 szText 元素为空。

FALSE

错误

错误消息

MSG_ERR_API_PARAM	参数无效
MSG_ERR_API_SERVICE	dwServiceID 无效

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSRTStartMsgServicePlus (页 2791)

MSG_COMMENT_INSTANCE_STRUCT_PLUS (页 2757)

3.10.9.2 MSRTSetCommentInstancePlus

说明

指定已记录的报警的注释文本。

声明

```
Bool MSRTSetCommentInstancePlus (
    DWORD                                dwServiceID,
    LPMSG_COMMENT_INSTANCE_STRUCT_PLUS   lpmCommentPlus,
    DWORD*                               pdwTAID,
    LPCMN_ERRORW                         lpError );
```

参数

dwServiceID

调用 MSRTStartMsgServicePlus 时返回的服务 ID。

lpmCommentPlus

指向具有注释数据的 MSG_COMMENT_INSTANCE_STRUCT_PLUS (页 2757) 类型结构的指针。结构中需要以下各值：报警编号、实例名称、日期、时间、文本和用户名称。

pdwTAID

指向用于异步处理反馈消息的事务 ID 的指针

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

成功发送给服务器。

FALSE

错误

注释

异步写入 MSRTSetCommentPlus 函数的数据。因此，并非所有错误都返回给用户。使用 MSRTGetCommentInstancePlus 来检查如何写入数据。

如果在 MSRTCreateMsgPlus 或 MSRTCreateMsgInstancePlus 之后执行 MSRTSetCommentInstancePlus，则异步写入操作以及系统利用率情况可能会导致有些数据被其它数据“取代”。因此，在创建报警之前指定注释。

使用 MSRTCreateMsgInstanceWithCommentPlus 函数替换调用组合。

错误消息

MSG_ERR_API_PARAM	参数无效
MSG_ERR_API_SERVICE	dwServiceID 无效

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务
MSRTCreateMsgInstanceWithCommentPlus (页 2808)	创建报警
MSRTCreateMsgInstancePlus (页 2810)	创建报警

参见

MSG_COMMENT_INSTANCE_STRUCT_PLUS (页 2757)

MSRTStartMsgServicePlus (页 2791)

MSRTCreateMsgInstanceWithCommentPlus (页 2808)

MSRTCreateMsgInstancePlus (页 2810)

3.10.10 用于信息文本处理的函数

3.10.10.1 MSRTGetInfotextPlus

说明

获取报警的信息文本。为此，需要传送以下值：报警编号

声明

```
BOOL WINAPI MSRTGetInfotextPlus (
    DWORD                           dwServiceID,
    LPMSG_INFOTEXT_STRUCT_PLUS      lpmInfotext,
    LPCMN_ERRORW                    lpError );
```

参数

dwServiceID

服务 ID - 服务。

lpmInfotext

指向信息文本结构的指针

lpError

包含扩展错误消息的指针

返回值

TRUE

函数成功

FALSE

错误

错误消息

MSG_ERR_API_PARAM	无效参数
MSG_ERR_API_SERVICE	dwServiceID 无效

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSG_INFOTEXT_STRUCT_PLUS (页 2772)

MSRTStartMsgServicePlus (页 2791)

3.10.10.2 MSRTSetInfotextPlus

说明

设置报警的信息文本。

声明

```
BOOL WINAPI MSRTSetInfotextPlus(
    DWORD                           dwServiceID,
    LPMSG_INFOTEXT_STRUCT_PLUS      lpmInfotextPlus,
    DWORD*                          lpdwTAID,
    LPCMN_ERRORW                    lpError );
```

参数

dwServiceID

服务 ID - 服务。

lpmInfotextPlus

指向信息文本结构的指针

lpdwTAID

指向用于异步处理的且具有反馈的 TransactionID 的指针。

lpError

包含扩展错误消息的指针。

返回值

TRUE

函数成功

FALSE

错误

注释

新的信息文本将从切换后的下一分钟生效。 在此时间之前，将使用上一个信息文本。

错误消息

MSG_ERR_API_PARAM	无效参数
MSG_ERR_API_SERVICE	dwServiceID 无效

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSG_INFOTEXT_STRUCT_PLUS (页 2772)

MSRTStartMsgServicePlus (页 2791)

3.10.11 归档函数

3.10.11.1 MSRTEnumBackupListPlus

说明

列出顺序归档的导出文件的条目。

声明

```
BOOL MSRTEnumBackupListPlus (
    DWORD          dwServiceID,
    LPCMN_ERROR    lpError );
```

参数

dwServiceID

当调用 MSRTStartMsgServicePlus 时返回的发送和接收服务的标识编号。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

导出列表已创建。

FALSE

错误

注释

Enum 函数异步执行。这使得任何调用都必须通过信号量进行同步。

使用 MSG_NOTIFY_BACKUPENUM 参数调用回调函数 MSG_SERVICE_NOTIFY_PROCPlus。lpbyData 参数指向 MSG_BACKUP_STRUCT_PLUS (页 2753) 结构中的数据。

错误消息

MSG_ERR_API_PARAM	参数无效
MSG_ERR_API_SERVICE	dwServiceID 无效

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务
MSG_SERVICE_NOTIFY_PROC_PLUS (页 2798)	回调函数

参见

MSG_BACKUP_STRUCT_PLUS (页 2753)

MSRTStartMsgServicePlus (页 2791)

MSG_SERVICE_NOTIFY_PROC_PLUS (页 2798)

3.10.11.2 MSRTExportPlus

说明

从循环日志以及相关的日志备份文件导出报警。

声明

```
BOOL MSRTExportPlus (
    DWORD                       dwServiceID,
    LPMSG_BACKUP_STRUCT_PLUS      lpMsgBackup,
    LPCMN_ERRORW                lpError );
```

参数

dwServiceID

当调用 MSRTStartMsgServicePlus 时返回的发送和接收服务的标识编号。

lpMsgBackup

指向 MSG_BACKUP_STRUCT_PLUS (页 2753) 类型结构的指针。

lpError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

报警已导出。

FALSE

错误

错误消息

MSG_ERR_API_PARAM	参数无效
MSG_ERR_API_SERVICE	dwServiceID 无效

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSRTStartMsgServicePlus (页 2791)

MSG_BACKUP_STRUCT_PLUS (页 2753)

3.10.11.3 MSRTEnumArchivDataPlus

说明

枚举指定日志中的所有报警。使用 MSG_NOTIFY_ARCHIVENUM 通知类型调用其回调函数 MSG_SERVICE_NOTIFY_PROC_PLUS。lpbyData 指向 MSG_RTDATA_INSTANCECOMMENT_STRUCT_PLUS 结构的数据。

枚举函数异步执行。因此，必须使用信号量同步调用。

声明

```
BOOL MSRTEnumArchivDataPlus(
    DWORD           dwServiceID,
    BOOL            fArchiv,
    DWORD           dwMaxRecords
    DWORD           dwParams
    LPDWORD         lpdwTAID
    LPCMN_ERROR     lpError );
```

参数

dwServiceID

当调用 MSRTStartMsgServicePlus 时返回的发送和接收服务的标识编号。

fArchiv

指定日志类型。

TRUE	顺序日志

FALSE	循环日志

dwMaxRecords

要列出的最大数据记录数。如果超出 dwMaxRecords，编号将停止。如果 dwMaxRecords = 0xFFFFFFFF，将列出所有报警。

dwParams

排序顺序参数：

MSG_ARCHIV_ENUM_DESC	按降序排序

lpdwTAID

指向用于异步处理的 TransactionID 的指针。

lpError

指向 CMN_ERRORW 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

报警已列出。

FALSE

错误

错误消息

MSG_ERR_API_PARAM	参数无效

所需文件

CCMSRTCLIPlus.h

CCMSRTCLIPlus.lib

CCMSRTCLIPlus.dll

相关函数

MSG_SERVICE_NOTIFY_PROC_PLUS (页 2798)	回调函数
MSRTStartMsgServicePlus (页 2791)	启动服务

参见

MSG_SERVICE_NOTIFY_PROC_PLUS (页 2798)

MSRTStartMsgServicePlus (页 2791)

函数概览 (页 2731)

3.11 用于显示 PLC 代码的函数

3.11.1 在 STEP 7 中显示

3.11.1.1 基础知识

常规信息

利用该函数，可直接从 WinCC Runtime 的屏幕切换到 STEP 7 程序代码中过程变量的使用位置。这样就可以快捷地进行故障诊断。要加快输入点函数的速度，可启动 TIA Portal 并打开项目。通常在启动 WinCC Runtime 时完成。

首先在输入点执行检查以确定是否已打开 TIA Portal。否则，将自动启动 TIA Portal。TIA Portal 中将打开相应编辑器，并对使用、分配、调用或设置的点执行搜索。

头文件

此处对头文件中的函数和结构声明进行解释：

kopapi.h	API 接口定义头文件

库

此处解释的函数 DLL 链接在 kopapi.dll"lib"文件中提供

kopapi.lib	库
kopapi.dll	

参见

OpenTIAPortalProject (页 2874)

OpenTIAPortalIECPLByCall (页 2875)

OpenTIAPortalIECPLByAssignment (页 2878)

OpenTIAPortalS7GraphByBlock (页 2880)

示例： WinCC 函数的集成 (页 2883)

3.11.1.2 OpenTIAPortalProject

描述

可使用该函数启动 TIA Portal 并打开项目。这会加快各跳转函数的速度。但同时应考虑到在后台打开程序需要一定内存和计算时间。

声明

```
BOOL OpenTIAPortalProject (
    DWORD      dwFlags,
    LPCTSTR    lpszTiaPortalProjectPath,
    LPCTSTR    lpszErrorTag,
    LPCMN_ERROR    lpdmError);
```

参数

dwFlags

位数组，每一位的值逐位进行“或”运算。默认情况下，dwFlags 应为 0。

- KOPAPI_FLAG_TIAPORTAL_CHECK_PROJECT_STATE：如果设置了该位，则将对项目状态进行检查。项目未打开。
- 项目已经打开。
  返回值 FALSE
  error.dwError1 = KOPAPI_E_TIAPORTAL_PROJECT_NOT_OPEN
- 项目已打开且未做任何更改。
  返回值 TRUE
  error.dwError1 = 0
- 项目已打开且已进行了更改。
  返回值 FALSE
  error.dwError1 = KOPAPI_E_TIAPORTAL_PROJECT_MODIFIED
- 项目以只读模式打开。
  返回值 FALSE
  error.dwError1 = KOPAPI_E_TIAPORTAL_PROJECT_READ_ONLY

KOPAPI_FLAG_TIAPORTAL_DONT_USE_MODIFIED_PROJECT：如果该位置位，在项目已打开并包含更改时将取消调用。

KOPAPI_FLAG_TIAPORTAL_OPEN_READONLY：当此位设为 TRUE 时，在 TIA Portal 工程系统中以写保护模式打开项目。

lpszTiaPortalProjectPath

项目文件的名称，其中包括绝对路径的规范，例如"D:\TIAProjects\Project1\Project1.ap14"

请注意，在 C 脚本内必须以转义序列形式书写反斜线：
FunctionX(..., "D:\\TIAProjects\\Project1\\Project1.ap14", ...);

lpszErrorTag

字符串数据类型的内部 WinCC 变量名称。如果调用不提供即时结果的异步函数，则 lpszErrorTag (页 2899) 中将返回错误信息。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。如果发生错误，则系统将向该结构写入错误信息。

返回值

TRUE

函数已执行，未发生任何错误。

FALSE

发生错误。

所需文件

kopapi.h

kopapi.lib

kopapi.dll

参见

基础知识 (页 2873)

3.11.1.3 OpenTIAPortalIECPLByCall

描述

该函数适用于 LAD 和 FBD 语言，显示标准块程序段输入之前的逻辑。

声明

```
BOOL OpenTIAPortalIECPLByCall (
    DWORD      dwFlags,
    LPCTSTR    lpszTiaPortalProjectPath,
    LPCTSTR    lpszCpuName,
    LPCTSTR    lpszContainingBlock,
    LPCTSTR    lpszCalledBlock,
    LPCTSTR    lpszPin,
    LPCTSTR    lpszErrorTag,
    LPCMN_ERROR    lpdmError);
```

参数

dwFlags

位数组，每一位的值逐位进行“或”运算。默认情况下，dwFlags 应为 0。

- IECPLVIEWER_PIN_SUBSTRING_SEARCH=0x0001：搜索针脚名称时搜索子串，即针脚名称以 lpszPin 中传送的字符串开头。如果不设置该位，则将整个针脚名称与 lpszPin 进行比较。
- KOPAPI_FLAG_TIAPORTAL_SUPPRESS_PROGRAM_STATUS=0x0004：TIA Portal 不会在块打开后进入在线模式。如果不设置该位，则块打开后将启动在线模式。
- KOPAPI_FLAG_TIAPORTAL_DONT_USE_MODIFIED_PROJECT 0x0008L： 如果该位置位，在项目已打开并包含更改时将取消调用。
- KOPAPI_FLAG_TIAPORTAL_OPEN_READONLY：当此位设为 TRUE 时，在 TIA Portal 工程系统中以写保护模式打开项目。

lpszTiaPortalProjectPath

项目文件的名称，其中包括绝对路径的规范，例如“D:\TIAProjects\Project1\Project1.ap14”

请注意，在 C 脚本内必须以转义序列形式书写反斜线：
FunctionX(..., "D:\\TIAProjects\\Project1\\Project1.ap14", ...);

lpszCpuName

S7 CPU 的名称。该名称与 TIA Portal 项目树中显示的站名称相同。

lpszContainingBlock

要打开或显示的块的名称，或 FB 实例的名称。

可用名称如下所示：

- 单个背景数据块的名称。随后显示其 FB。示例“Station1”
- 背景数据块中多重背景的名称。随后显示其 FB。指定多重背景名称路径后，将为诸如 DB 编辑器（而非调用结构）中所显示的数据层级。名称的第一部分（“Line1”）不包含在引号中，因为这是全局图标，上下文可识别。对于各个名称组件，如果包含空格、句点等特殊字符，则需要加引号。示例："Line1.Cell1.Station1"
- FC 或 OB 的名称

不允许使用 FB 的名称。

lpszCalledBlock

属于 lpszContainingBlock 的代码块所调用的局部或全局实例的名称。

- 对于局部实例，此处必须指定井号 #，如“#feeder1”。
- 对于全局实例，此处必须指定不包含井号 # 的全局名称，如“feeder3”。

不允许使用 FC 的名称。

如果在 lpszContainingBlock 或其 FB 中多次调用 lpszCalledBlock，则输入点始终为首次调用 lpszCalledBlock。

仅当 lpszCalledBlock=0 时，状态中才打开并显示 lpszContainingBlock，但不会搜索特定块调用或特定程序段。这种情况下，将忽略 lpszPin。

lpszPin

lpszCalledBlock 输入针脚的名称。该参数用于在编辑器中显示程序段内的指定针脚。

仅当 lpszPin=0 时，lpszCalledBlock 才显示为可见。

lpszErrorTag

字符串数据类型的内部 WinCC 变量名称。如果调用不提供即时结果的异步函数，则 lpszErrorTag (页 2899) 中将返回错误信息。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

函数已执行，未发生任何错误。

FALSE

发生错误。

所需文件

kopapi.h

kopapi.lib

kopapi.dll

参见

基础知识 (页 2873)

3.11.1.4 OpenTIAPortalIECPLByAssignment

描述

该函数支持 LAD 和 FBD 语言，用于显示操作数的分配情况及其之前的逻辑。

声明

```
BOOL OpenTIAPortalIECPLByAssignment (
    DWORD      dwFlags,
    LPCTSTR    lpszTiaPortalProjectPath,
    LPCTSTR    lpszCpuName,
    LPCTSTR    lpszContainingBlock,
    LPCTSTR    lpszOperand,
    LPCTSTR    lpszErrorTag,
    LPCMN_ERROR    lpdmError);
```

参数

dwFlags

位数组，每一位的值逐位进行“或”运算。默认情况下，dwFlags 应为 0。

- KOPAPI_FLAG_TIAPORTAL_SUPPRESS_PROGRAM_STATUS=0x0004：TIA Portal 不会在块打开后进入在线模式。如果不设置该位，则块打开后将启动在线模式。
- KOPAPI_FLAG_TIAPORTAL_DONT_USE_MODIFIED_PROJECT 0x0008L： 如果该位置位，在项目已打开并包含更改时将取消调用。
- KOPAPI_FLAG_TIAPORTAL_OPEN_READONLY：当此位设为 TRUE 时，在 TIA Portal 工程系统中以写保护模式打开项目。

lpszTiaPortalProjectPath

项目文件的名称，其中包括绝对路径的规范，例如“D:\TIAProjects\Project1\Project1.ap14”

请注意，在 C 脚本内必须以转义序列形式书写反斜线：
FunctionX(..., "D:\\TIAProjects\\Project1\\Project1.ap14", ...);

lpszCpuName

S7 CPU 的名称。该名称与 TIA Portal 项目树中显示的站名称相同。

lpszContainingBlock

要打开或显示的块的名称，或 FB 实例的名称。

可用名称如下所示：

- 单个背景数据块的名称。随后显示其 FB。示例“Station1”
- 背景数据块中多重背景的名称。随后显示其 FB。指定多重背景名称路径后，将为诸如 DB 编辑器（而非调用结构）中所显示的数据层级。名称的第一部分（“Line1”）不包含在引号中，因为这是全局图标，上下文可识别。对于各个名称组件，如果包含空格、句点等特殊字符，则需要加引号。示例："Line1.Cell1.Station1"
- FC 或 OB 的名称

不允许使用 FB 的名称。

lpszOperand

执行分配的局部或全局操作符的名称。

属于 lpszContainingBlock 的代码块所调用的局部或全局实例的名称。

- 对于局部操作符，此处还必须指定井号 #。
- 对于全局操作符，此处必须指定不包含井号 # 的全局名称。

如果在 lpszContainingBlock 或其 FB 中多次写入 lpszOperand，则输入点始终为首次写入 lpszOperand 访问权限。

lpszErrorTag

字符串数据类型的内部 WinCC 变量名称。如果调用不提供即时结果的异步函数，则 lpszErrorTag (页 2899) 中将返回错误信息。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。出现错误时，系统向该结构中写入错误信息。

返回值

TRUE

函数已执行，未发生任何错误。

FALSE

发生错误。

所需文件

kopapi.h

kopapi.lib

kopapi.dll

参见

基础知识 (页 2873)

3.11.1.5 OpenTIAPortalS7GraphByBlock

描述

该函数适用于 S7 Graph 语言，可以显示顺控程序中的步长。

声明

```
BOOL OpenTIAPortalS7GraphByBlock (
    DWORD      dwFlags,
    LPCTSTR    lpszTiaPortalProjectPath,
    LPCTSTR    lpszCpuName,
    LPCTSTR    lpszBlock,
    DWORD      dwStepNumber,
    LPCTSTR    lpszErrorTag,
    LPCMN_ERROR    lpdmError);
```

参数

dwFlags

位数组，每一位的值逐位进行“或”运算。默认情况下，dwFlags 应为 0。

- KOPAPI_FLAG_TIAPORTAL_SUPPRESS_PROGRAM_STATUS=0x0004：TIA Portal 不会在块打开后进入在线模式。如果不设置该位，则块打开后将启动在线模式。
- KOPAPI_FLAG_TIAPORTAL_DONT_USE_MODIFIED_PROJECT 0x00000008L： 如果该位置位，在项目已打开并包含更改时将取消调用。
- KOPAPI_FLAG_TIAPORTAL_OPEN_READONLY 当此位设为 TRUE 时，在 TIA Portal 工程系统中以写保护模式打开项目。

lpszTiaPortalProjectPath

项目文件的名称，其中包括绝对路径的规范，例如“D:\TIAProjects\Project1\Project1.ap14”

请注意，在 C 脚本内必须以转义序列形式书写反斜线：
FunctionX(..., "D:\\TIAProjects\\Project1\\Project1.ap14", ...);

lpszCpuName

S7 CPU 的名称。该名称与 TIA Portal 项目树中显示的站名称相同。

lpszBlock

要显示的 S7 Graph 块的实例名称。

dwStepNumber

要显示的步数

dwStepNumber=0 将自动搜索活动步并启用“跟踪活动步”模式。

lpszErrorTag

字符串数据类型的内部 WinCC 变量名称。如果调用不提供即时结果的异步函数，则 lpszErrorTag (页 2899) 中将返回错误信息。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。如果发生错误，则系统将向该结构写入错误信息。

返回值

TRUE

函数已执行，未发生任何错误。

FALSE

发生错误。

所需文件

kopapi.h

kopapi.lib

kopapi.dll

参见

基础知识 (页 2873)

3.11.1.6 示例： WinCC 函数的集成

说明

以下示例展示了在用户自定义函数中嵌入函数调用。为便于维护，应在全局脚本函数中执行外部调用，随后确定指向 TIA Portal 的路径和 PLC 站的名称。

```
void OnClick(char* screenName, char* objectName, char* propertyName)
    {
    // Funktionen bekannt machen'
    #pragma code("KOPAPI.dll")
    #include "KOPAPI.h"
    #pragma code()

    BOOL useTiaPortal = TRUE; // use the TIA Portal or the viewer control?
    char* pTiaPortalProject = "c:\\Projects\\myproject\\myproject.ap14";
    char* pStationName = "PLC_Name"; // TODO: read from internal tag
   char* pContainingBlock = "Block_IDB"; // TODO: get from current selection
    char* pOperand = "OUT"; // TODO: get from current selection
    CMN_ERROR error;
    BOOL result;

    if(useTiaPortal)
        {
        result = OpenTIAPortalIECPLByAssignment(0, pTiaPortalProject,
pStationName, pContainingBlock, pOperand, &error);
        }
    else
        {
        char *pServerPrefix = NULL, *pTagPrefix = NULL, *pWindowPrefix =
NULL;
        // determine ServerPrefix of the current environment
        GetServerTagPrefix(&pServerPrefix, &pTagPrefix, &pWindowPrefix);
        // make the screen which contains the viewer control visible
        SetVisible("SYSTEM#Basic_Screen", "Screen_window_IECPLViewer",
TRUE);
        result = OpenViewerIECPLByAssignment(0, pServerPrefix,
"SYSTEM#IECPLViewer", "IECPLViewerObject", pStationName, pContainingBlock,
pOperand, &error);
        }
}
```

参见

基础知识 (页 2873)

3.11.2 在 PLC 代码显示中显示

3.11.2.1 基础知识

常规信息

在 PLC 代码视图中，可使用这些函数来显示 PLC 程序的当前程序状态。

必须已打开包含 PLC 代码视图的画面，例如可通过 ActivateScreen() 打开。

头文件

此处对头文件中的函数和结构声明进行解释：

kopapi.h	API 接口定义头文件

库

此处解释的函数 DLL 链接在"lib"文件中提供：pdecscli.dll

kopapi.lib	库
kopapi.dll	

关于定义多重背景名称的注意事项

指定多重背景名称路径后，将为诸如 DB 编辑器（而非调用结构）中所显示的数据层级。名称的第一部分（"Line1"）不包含在引号中，因为这是全局图标，上下文可识别。对于单独的名称组件，如果包含空格、点等内容，则需要引号。示例："Line1.Cell1.Station1"。不包含特殊字符的名称不得放在引号中。

参见

OpenViewerIECPLByCall (页 2887)

OpenViewerIECPLByAssignment (页 2894)

OpenViewerS7GraphByBlock (页 2885)

3.11.2.2 OpenViewerS7GraphByBlock

说明

此函数会在 PLC 代码视图画面中以 PLC 语言 S7 Graph 显示顺控程序所调用的程序步。

声明

```
BOOL OpenViewerS7GraphByBlock (
    DWORD      dwFlags,
    LPCTSTR    lpszServerPrefix,
    LPCTSTR    lpszPictureName,
    LPCTSTR    lpszObjectName,
    LPCTSTR    lpszCpuName,
    LPCTSTR    lpszBlock,
    DWORD      dwStepNumber,
    LPCMN_ERROR    lpdmError);
```

参数

dwFlags

位数组，每一位的值逐位进行“或”运算。

说明

要显示层级注释，需要将位 0x4 置 1。

lpszServerPrefix

该参数为以后升级预留。

lpszPictureName

具有 PLC 代码视图的画面的名称。

lpszObjectName

PLC 代码视图的名称。

lpszCpuName

S7 CPU 的名称。该名称与 TIA Portal 项目树中显示的站名称相同。名称中不允许使用逗号。

lpszBlock

要显示的 S7 Graph 块的实例名称。如果名称中出现空格、句点等特殊字符，则需要使用引号。

dwStepNumber

要显示的步数。

dwStepNumber=0 将自动搜索活动步并启用“跟踪活动步”模式。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。如果发生错误，则系统将向该结构写入错误信息。

返回值

TRUE

函数已执行，未发生任何错误。

FALSE

发生错误。

所需文件

kopapi.h

kopapi.lib

kopapi.dll

示例

以下示例展示了在用户自定义 C 函数中嵌入函数调用。为便于维护，应在全局脚本函数中执行外部调用，随后确定 PLC 站的名称。

```
BOOL OpenViewerS7GraphByBlock(char* screenName, char* objectName, char*
cpuName, char* instanceDBName)
{
#pragma code("kopapi.dll")
#include "kopapi.h"
#pragma code()
BOOL result;
CMN_ERROR error;
char* serverPrefix = "";
DWORD dwFlags = 0;
DWORD stepNumber = 0;
result = OpenViewerS7GraphByBlock(dwFlags, serverPrefix, screenName,
objectName,
  cpuName, instanceDBName, stepNumber, &error);
if(!result)
{
  // there are only few reasons why the call to OpenViewerS7GraphByBlock will
  // fail, in most cases the viewer control could not be found
  // most of the errors have to be handled in the OnError event of the viewer
  printf("OpenViewerS7GraphByBlock failed: err1=%ld, err2=%ld, err3=%ld,
err4=%ld,   err5=%ld, text=\"%s\"\r\n", result, error.dwError1,
error.dwError2,   error.dwError3, error.dwError4, error.dwError5,
error.szErrorText);
}
return result;
}
```

参见

错误处理 (页 2899)

基础知识 (页 2884)

3.11.2.3 OpenViewerIECPLByCall

说明

该函数适用于 LAD 和 FBD 语言，并会在 PLC 代码查看器中显示标准块程序段输入之前的逻辑。

.

说明

如果该输入被常量（TRUE、FALSE）占用，则不会在 PLC 代码视图中显示该值。要在 PLC 代码视图中正确显示值，应为具有适当值的输入分配一个变量。

声明

```
BOOL OpenViewerIECPLByCall (
    DWORD      dwFlags,
    LPCTSTR    lpszServerPrefix,
    LPCTSTR    lpszPictureName,
    LPCTSTR    lpszObjectName,
    LPCTSTR    lpszCpuName,
    LPCTSTR    lpszContainingBlock,
    LPCTSTR    lpszCalledBlock,
    LPCTSTR    lpszPin,
    LPCMN_ERROR    lpdmError);
```

参数

dwFlags

位数组，每一位的值逐位进行“或”运算。

说明

要显示层级注释，需要将位 0x4 置 1。

- IECPLVIEWER_PIN_SUBSTRING_SEARCH=0x0001：搜索针脚名称时搜索子串，即针脚名称以 lpszPin 中传送的字符串开头。如果不设置该位，则将整个针脚名称与 lpszPin 进行比较。

lpszServerPrefix

该参数为以后升级预留。

lpszPictureName

具有 PLC 代码视图的画面的名称。

lpszObjectName

PLC 代码视图的名称。

lpszCpuName

S7 CPU 的名称。该名称与 TIA Portal 项目树中显示的站名称相同。名称中不允许使用逗号。

lpszContainingBlock

要打开或显示的块的名称，或 FB 实例的名称。

可用名称如下所示：

- 单个背景数据块的名称。随后显示其 FB。示例“Station1”
- 背景数据块中多重背景的名称。随后显示其 FB。指定多重背景名称路径后，将为诸如 DB 编辑器（而非调用结构）中所显示的数据层级。名称的第一部分（“Line1”）不包含在引号中，因为这是全局图标，上下文可识别。对于单独的名称组件，如果包含空格、点等内容，则需要引号。示例："Line1.Cell1.Station1"
- FC 或 OB 的名称

不允许使用 FB 的名称。

lpszCalledBlock

属于 lpszContainingBlock 的代码块所调用的局部或全局实例的名称。

- 对于局部实例，此处必须指定井号 #，如“#feeder1”。
- 对于全局实例，此处必须指定不包含井号 # 的全局名称，如“feeder3”。

不允许使用 FC 的名称。

如果在 lpszContainingBlock 或其 FB 中多次调用 lpszCalledBlock，则输入点始终为首次调用 lpszCalledBlock。

lpszPin

lpszCalledBlock 输入针脚的名称。该参数用于视图与 PLC 代码显示中的输入针脚互连的网络。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。如果发生错误，则系统将向该结构写入错误信息。

返回值

TRUE

函数已执行，未发生任何错误。

FALSE

发生错误。

所需文件

kopapi.h

kopapi.lib

kopapi.dll

示例

以下示例展示了在用户自定义 C 函数中嵌入函数调用。为便于维护，应在全局脚本函数中执行外部调用，随后确定 PLC 站的名称。

```
BOOL OpenCodeViewerByCall(BOOL matchSubstringPin, char* screenName, char*
objectName, char* cpuName, char* containingBlock, char* calledBlock, char*
pinName)
{
#pragma code("kopapi.dll")
#include "kopapi.h"
#pragma code()
BOOL result;
CMN_ERROR error;
char* serverPrefix = "";
DWORD dwFlags = 0;
if(matchSubstringPin)
  dwFlags |= KOPAPI_FLAG_TIAPORTAL_PIN_SUBSTRING_SEARCH;
result = OpenViewerIECPLByCall(dwFlags, serverPrefix, screenName,
objectName,
  cpuName, containingBlock, calledBlock, pinName, &error);
if(!result)
{
  // there are only few reasons why the call to OpenViewerIECPLByCall will
fail, in   // most cases the viewer control could not be found
  // most of the errors have to be handled in the OnError event of the viewer
  printf("OpenViewerIECPLByCall failed: err1=%ld, err2=%ld, err3=%ld,
err4=%ld,   err5=%ld, text=\"%s\"\r\n", result, error.dwError1,
error.dwError2,   error.dwError3, error.dwError4, error.dwError5,
error.szErrorText);
}
return result;
}
```

参见

错误处理 (页 2899)

基础知识 (页 2884)

3.11.2.4 OpenViewerIECPLByFCCall

说明

该函数适用于 LAD 和 FBD 语言，会在 PLC 代码查看器中显示标准块程序段输入之前的逻辑，同时还会考虑 UDT 实例。

说明

如果该输入被常量（TRUE、FALSE）占用，则不会在 PLC 代码视图中显示该值。要在 PLC 代码视图中正确显示值，应为具有适当值的输入分配一个变量。

声明

```
BOOL OpenViewerIECPLByFCCall (
    DWORD      dwFlags,
    LPCTSTR    lpszServerPrefix,
    LPCTSTR    lpszPictureName,
    LPCTSTR    lpszObjectName,
    LPCTSTR    lpszCpuName,
    LPCTSTR    lpszContainingBlock,
    LPCTSTR    lpszCalledBlock,
    LPCTSTR    lpszPin,
    LPCTSTR    lpszUdtInstance
    LPCMN_ERROR    lpdmError);
```

参数

dwFlags

位数组，每一位的值逐位进行“或”运算。

说明

要显示层级注释，需要将位 0x4 置 1。

- IECPLVIEWER_PIN_SUBSTRING_SEARCH=0x0001：搜索针脚名称时搜索子串，即针脚名称以 lpszPin 中传送的字符串开头。如果不设置该位，则将整个针脚名称与 lpszPin 进行比较。

lpszServerPrefix

该参数为将来开发预留，必须预设为空字符串 ("")。

lpszPictureName

具有 PLC 代码视图的画面的名称。

lpszObjectName

PLC 代码视图的名称。

lpszCpuName

S7 CPU 的名称。该名称与 TIA Portal 项目树中显示的站名称相同。名称中不允许使用逗号。

lpszContainingBlock

要打开或显示的块的名称，或 FB 实例的名称。

可用名称如下所示：

- 单个背景数据块的名称。随后显示其 FB。示例“Station1”
- 背景数据块中多重背景的名称。随后显示其 FB。指定多重背景名称路径后，将为诸如 DB 编辑器（而非调用结构）中所显示的数据层级。名称的第一部分（“Line1”）不包含在引号中，因为这是全局图标，上下文可识别。对于单独的名称组件，如果包含空格、点等内容，则需要引号。示例："Line1.Cell1.Station1"
- FC 或 OB 的名称

不允许使用 FB 的名称。

lpszCalledBlock

属于 lpszContainingBlock 的代码块所调用的局部或全局实例的名称。

- 对于局部实例，此处必须指定井号 #，如"#feeder1"。
- 对于全局实例，此处必须指定不包含井号 # 的全局名称，如"feeder3"。

不允许使用 FC 的名称。

如果在 lpszContainingBlock 或其 FB 中多次调用 lpszCalledBlock，则输入点始终为首次调用 lpszCalledBlock。

lpszPin

lpszCalledBlock 输入针脚的名称。该参数用于视图与 PLC 代码显示中的输入针脚互连的网络。

lpszUdtInstance

此参数用于限制多次调用的 FB 或 FC 的显示。此限制基于与给定输入针脚或 inout 针脚互联的 UDT 实例。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。如果发生错误，则系统将向该结构写入错误信息。

返回值

TRUE

函数已执行，未发生任何错误。

FALSE

发生错误。

所需文件

kopapi.h

kopapi.lib

kopapi.dll

示例

以下示例展示了在用户自定义 C 函数中嵌入函数调用。为便于维护，应在全局脚本函数中执行外部调用，随后确定 PLC 站的名称。

```
BOOL OpenCodeViewerByFCCall(BOOL matchSubstringPin, char* screenName,
char* objectName, char* cpuName, char* containingBlock, char* calledBlock,
char* pinName, char* udtInstance)
{
#pragma code("kopapi.dll")
#include "kopapi.h"
#pragma code()
BOOL result;
CMN_ERROR error;
char* serverPrefix = "";
DWORD dwFlags = 0;
if(matchSubstringPin)
  dwFlags |= KOPAPI_FLAG_TIAPORTAL_PIN_SUBSTRING_SEARCH;
result = OpenViewerIECPLByFCCall(dwFlags, serverPrefix, screenName,
objectName, cpuName, containingBlock, calledBlock, pinName, udtInstance,
&error);
if(!result)
{
  // there are only few reasons why the call to OpenViewerIECPLByFCCall will
fail,
// in most cases the viewer control could not be found
// most of the errors have to be handled in the OnError event of the viewer
control
  printf("OpenViewerIECPLByFCCall failed: err1=%ld, err2=%ld, "" err3=%ld,
err4=%ld,   err5=%ld, text=\"%s\"\r\n", result, error.dwError1,
error.dwError2,   error.dwError3, error.dwError4, error.dwError5,
error.szErrorText);
}
return result;
}
```

3.11.2.5 OpenViewerIECPLByAssignment

说明

此函数适用于 LAD 和 FBD 语言，在 PLC 代码查看器中显示操作数分配情况及其之前的逻辑。

声明

```
BOOL OpenViewerIECPLByAssignment (
    DWORD      dwFlags,
    LPCTSTR    lpszServerPrefix,
    LPCTSTR    lpszPictureName,
    LPCTSTR    lpszObjectName,
    LPCTSTR    lpszCpuName,
    LPCTSTR    lpszContainingBlock,
    LPCTSTR    lpszOperand,
    LPCMN_ERROR    lpdmError);
```

参数

dwFlags

位数组，每一位的值逐位进行“或”运算。

说明

要显示层级注释，需要将位 0x4 置 1。

lpszServerPrefix

该参数为以后升级预留。

lpszPictureName

具有 PLC 代码视图的画面的名称。

lpszObjectName

PLC 代码视图的名称。

lpszCpuName

S7 CPU 的名称。该名称与 TIA Portal 项目树中显示的站名称相同。名称中不允许使用逗号。

lpszContainingBlock

要打开或显示的块的名称，或 FB 实例的名称。

可用名称如下所示：

- 单个背景数据块的名称。随后显示其 FB。示例“Station1”
- 背景数据块中多重背景的名称。随后显示其 FB。指定多重背景名称路径后，将为诸如 DB 编辑器（而非调用结构）中所显示的数据层级。名称的第一部分（“Line1”）不包含在引号中，因为这是全局图标，上下文可识别。对于单独的名称组件，如果包含空格、点等内容，则需要引号。示例："Line1.Cell1.Station1"
- FC 或 OB 的名称

不允许使用 FB 的名称。

lpszOperand

执行分配的局部或全局操作符的名称。

属于 lpszContainingBlock 的代码块所调用的局部或全局实例的名称。对于全局操作符，此处必须指定不包含井号 # 的全局名称。

操作数存在以下限制：

- 无法使用“TEMP”区域的变量
- 无法使用输入变量
- 只允许在 FB 中使用静态变量
- 只有在网络中未使用本地变量的情况下才能显示 FC 中的网络
- 无法显示使用“TEMP”区域的变量的 FB 中的网络

如果在 lpszContainingBlock 或其 FB 中多次写入 lpszOperand，则输入点始终为首次写入 lpszOperand 访问权限。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。如果发生错误，则系统将向该结构写入错误信息。

返回值

TRUE

数据已传送到 PLC 代码视图。“操作数不受支持”(Operand not supported) 或“未找到操作数”(Operand not found) 等错误信息在 PLC 代码视图中输出，并且无法通过该函数进行查询。

FALSE

发生错误。

所需文件

kopapi.h

kopapi.lib

kopapi.dll

示例

以下示例展示了在用户自定义 C 函数中嵌入函数调用。为便于维护，应在全局脚本函数中执行外部调用，随后确定 PLC 站的名称。

```
BOOL OpenCodeViewerByAssignment(char* screenName, char* objectName, char*
cpuName, char* containingBlock, char* operand)
{
#pragma code("kopapi.dll")
#include "kopapi.h"
#pragma code()
BOOL result;
CMN_ERROR error;
char* serverPrefix = "";
DWORD dwFlags = 0;
result = OpenViewerIECPLByAssignment(dwFlags, serverPrefix, screenName,
objectName, cpuName, containingBlock, operand, &error);
if(!result)
{
  // there are only few reasons why the call to OpenViewerIECPLByAssignment
will
  // fail, in most cases the viewer control could not be found
  // most of the errors have to be handled in the OnError event of the viewer
  printf("OpenViewerIECPLByAssignment failed: err1=%ld, err2=%ld, err3=%ld,
   err4=%ld, err5=%ld, text=\"%s\"\r\n", result, error.dwError1,
error.dwError2,   error.dwError3, error.dwError4, error.dwError5,
error.szErrorText);
}
return result;
}
```

参见

错误处理 (页 2899)

基础知识 (页 2884)

3.11.2.6 IsJumpableProDiagAlarm

核心消息

该函数返回布尔值。

布尔值包含有关是否可跳转至所选报警的 PLC 代码的信息。

定义

```
BOOL WINAPI IsJumpableProDiagAlarm (
    LPCSTR screenName,
    LPCSTR objectName,
    long id,
    LPCMN_ERRORA lpdmError)
```

参数

screenName

指向在其中组态相关对象的画面数据的指针。

objectName

指向具有相应对象（报警）名称的数据的指针。

id

报警列表中报警的数目，而不是报警的 ID。

如果选择多个报警，该数值为"0"。

lpdmError

指向 CMN_ERROR 结构中的扩展错误消息数据的指针。如果发生错误，则系统将向该结构写入错误信息。

返回值

TRUE

可跳转至所选报警的 PLC 代码。

FALSE

无法跳转至所选报警的 PLC 代码。

所需文件

kopapi.h

kopapi.lib

kopapi.dll

示例

以下示例展示的是如何在用户自定义 C 函数中嵌入函数调用，其在报警视图中选择报警时即启动。

```
#include "GlobalDefinitions.h"
void OnSelectedIdChanged(char* screenName, char* objectName, long id) {
    #pragma code("KOPAPI.dll")
    #include "kopapi.h"
    #pragma code()
    CMN_ERROR errorStruct;
    BOOL bResult;

    bResult = IsJumpableProDiagAlarm ( screenName, objectName, id,
&errorStruct);
    SetPropBOOL ( screenname, "Button_Jump", "Enabled", bResult);
}
```

3.11.3 错误处理

常规信息

执行网络条目函数可分为同步部分和异步部分。同步部分对参数进行检查，然后将参数传送至异步部分。

函数的返回值可确定函数异步部分是否发生了错误。如果发生错误，则返回值为 FALSE。当未提供参数（如 lpszErrorTag 参数）或提供了错误参数时，会发生此类错误。

异步函数发生的错误由 lpszErrorTag 参数报告。此类错误包括：未找到项目、块不可用等，lpszErrorTag 中包含 String 数据类型的变量名称。要对返回值进行响应，必须组态一个函数，错误变量值更改时使用“待更改”周期触发。例如，当错误变量值为“RUNNING”时，还应使用错误变量防止通过按钮进一步调用函数。

lpszErrorTag

String 数据类型的变量名称。如果不需要变量，则可传送 NULL 作为参数。必要时，变量名称中可包含 ServerPrefix。

错误变量值的变更方式如下：

- 异步部分开始时，错误变量值设为“RUNNING”。
- 异步部分完成后，如果未发生错误，则错误变量值设为“OK”。
- 如果异步部分出错而终止，则错误变量将包含多行字符串，行与行之间以换行符 ('\n') 分隔。

如果发生错误，则错误变量的结构如下所示：

- 第 1 行：“ERROR”
- 第 2 行 - 第 6 行：十进制数字，数据类型：32 位无符号；DWORD
- 第 7 行：错误文本

错误文本

如果异步部分出错而终止，则错误变量将包含多行字符串，行与行之间以换行符 ('\n') 分隔。第七行将包含下列错误文本之一：

IDS_E_IS_TIA_PROJECT	KOPAPI：TIA Portal 项目无法使用该函数！
IDS_E_NO_TIA_PROJECT	KOPAPI：未打开任何 TIA Portal 项目，因此函数调用被禁用！
IDS_E_TIAPORTAL_UNKNOWN_FLAGS	dwFlags 含未知的值。
IDS_E_TIAPORTAL_ERRORTAG_NOT_EXIST	错误变量 [%s] 不存在。
IDS_E_TIAPORTAL_PREVIOUS_CALL_IS_RUNNING	先前的调用仍在运行 (RUNNING)。请先将其终止。
IDS_E_TIAPORTAL_CANNOT_WRITE_ERRORTAG	错误变量 [%s] 无法写入。
IDS_E_TIAPORTAL_COMACCESS_REGISTERPS_FAILED	无法建立与 TIA Portal 的连接。
IDS_E_TIAPORTAL_CANNOT_START_PORTAL	无法启动 TIA Portal。
IDS_E_TIAPORTAL_CANNOT_SEARCH_STARTED_PORTAL	无法对已启动的 Portal 执行搜索。

IDS_E_TIAPORTAL_NO_PORTAL_STARTED	无法启动 Portal。
IDS_E_TIAPORTAL_EXCEPTION_SYNC_PART	同步处理部分出现异常。
IDS_E_TIAPORTAL_EXCEPTION_ASYNC_PART	异步处理部分出现异常。
IDS_E_TIAPORTAL_NOT_INSTALLED	未安装 TIA Portal。
IDS_E_TIAPORTAL_PROJECT_CANNOT_OPEN	无法打开项目 [%s]。
IDS_E_TIAPORTAL_ALREADY_OPENED_WITH_OTHER_PROJECT	另一个项目已打开 TIA Portal。
IDS_E_TIAPORTAL_CREATECOMMAND	CreateCommand 错误。
IDS_E_TIAPORTAL_COMMAND_ADDARGUMENT	命令 AddArgument 错误。
IDS_E_TIAPORTAL_EXECUTECOMMAND	ExecuteCommand 发生严重错误。
IDS_E_TIAPORTAL_COMMAND_ERROR	TIA 项目命令错误：[%s,%ld]。
IDS_E_TIAPORTAL_COMMAND_UNKNOWN	TIA 项目未知命令：[%s,%ld]。

3.12 用于在外部应用程序中显示 PLC 代码的函数

3.12.1 基本知识

简介

WinCC 提供了相关库以支持用户通过运行系统在外部应用程序中显示 PLC 代码。块接口处未决的程序段或数值的相关信息将以"XML"格式返回。

支持 FBD 和 LAD 编程语言。下表列出了这两种编程语言所支持的指令：

指令	FBD	LAD
Contact	不支持	支持
NegContact	不支持	支持
Coil	支持	支持
NegCoil	支持	支持
RCoil	支持	支持
SCoil	支持	支持
PinCoil	支持	支持
OR	支持	支持

指令	FBD	LAD
AND	支持	不支持
XOR	支持	不支持
NOT	不支持	支持

如果程序段中包含其它指令，则会在"NetworkHeaderInfo"部分以 XML 格式返回一条错误。

推荐的开发环境

西门子建议您采用以下开发环境：

- MS Visual Studio 2012
- 工艺：COM 对象

要求

为了能够通过在 MS Visual Studio 中编程的应用程序显示 PLC 代码，需满足以下要求：

- 该应用程序位于已启动运行系统的 PC 上。
- 已在 HMI 与 PLC 之间组态了 HMI 连接。
- PLC 处于运行模式（用于查询动态数据）。

步骤

按照以下三个步骤查询数据：

1. 调用静态信息
   在该步骤中，使用"RequestByAssignment"或"RequestByCall"函数获取可用程序段的相关信息
2. 动态信息
   在该步骤中，将调用其中一个已确定程序段的当前数据流或未决状态的相关信息。
3. 响应事件（可选）
   PLC 的状态变化
   - 更改了用户程序
   - 与 PLC 的连接被中断
   - 与 PLC 的连接已恢复

   这些状态变化采用"OnEvent"方法并通过"IPlcCodeViewerExternalClientCB"接口提供：

```
HRESULT OnEvent( [in] PlcCodeViewerExternalClientEvent event );
```

在 MS Visual Studio 2012 中的执行过程

要访问"IPlcCodeViewerExternalClientAccess"接口，请按照以下顺序执行相关函数：

1. Initialize
2. SetLanguage
3. RequestByAssignment | RequestByCall
4. CancelRequest
5. Terminate

参见

XML 数据的结构 (页 2912)

3.12.2 查询程序段的统计信息

3.12.2.1 Initialize

描述

该函数用于初始化接口。

说明

对于 V16 版本，建议使用"InitializeEx"函数。新版本不再需要"serverString"参数。

声明

```
HRESULT Initialize( [in] IPlcCodeViewerExternalClientCB* pCallback,
                    [in] BSTR serverString);
```

参数

IPlcCodeViewerExternalClientCB

实现接口以提供查询程序段信息的方法。

serverString

正在运行运行系统的 PC 名称。

返回值

-

参见

Terminate (页 2911)

3.12.2.2 InitializeEx

描述

该函数用于初始化接口。

声明

```
HRESULT InitializeEx( [in] IPlcCodeViewerExternalClientCB* pCallback);
```

参数

IPlcCodeViewerExternalClientCB
实现接口以提供查询程序段信息的方法。

返回值

-

参见

Terminate (页 2911)

3.12.2.3 SetLanguage

描述

该函数用于定义要以 XML 文件格式输出的程序段变量注释的语言。

声明

```
HRESULT SetLanguage( [in] BSTR language);
```

参数

language

定义语言。根据如下参考页面，使用"Culture Name"进行表示：National Language Support (NLS) API Reference (https://msdn.microsoft.com/de-de/goglobal/bb896001.aspx)

必须始终为层级注释指定另一种语言，且仅输出此语言的层级注释。

如果没有指定语言的注释，则会采用以下语言之一输出其它注释：

1. 德语 (de-DE)
2. 英语（en-US 或 en-GB）
3. 如果未定义语言，则会以支持的全部语言输出注释。

返回值

-

3.12.2.4 RequestByAssignment

描述

该函数用于显示操作数的分配情况及其之前的逻辑。将输出写入了操作数的所有程序段。

说明

此函数是一个遗留函数，可使用它调用简单注释。

声明

```
HRESULT RequestByAssignement( [in] BSTR PLCName,
                              [in] BSTR ContainingBlock,
                              [in] BSTR OperandName,
                              [out] unsigned long* RequestId);
```

参数

PLCName

用户程序中包含相应块的 PLC 的名称。该名称与 TIA Portal 项目树中显示的站名称相同。

ContainingBlock

要显示的块的名称，或 FB 实例的名称。

可用名称如下所示：

- 单个背景数据块的名称。随后显示其 FB。示例“Station1”
- 背景数据块中多重背景的名称。随后显示其 FB。指定多重背景名称路径后，将作为 DB 编辑器中显示的数据层级，而非调用结构。名称的第一部分（“Line1”）不需要括在引号中，因为可以从上下文中清楚地看出其是全局图标。对于单独的名称组件，如果包含空格、点等内容，则需要引号。示例：“Line1.Cell1.Station1”
- FC 或 OB 的名称

OperandName

要在属于“ContainingBlock”的代码块中搜索的块的名称。

- 对于局部操作符，此处必须指定井号 #。

返回值

用于对请求进行唯一标识的 ID。调用该函数时将生成并返回 ID。

参见

RequestByCall (页 2908)

StartDynamicDataSubscription (页 2916)

CancelRequest (页 2910)

3.12.2.5 RequestByAssignmentEx

描述

该函数用于显示操作数的分配情况及其之前的逻辑。将输出写入了操作数的所有程序段。该函数返回层级注释。

声明

```
HRESULT RequestByAssignementEx( [in] BSTR PLCName,
                                [in] BSTR ContainingBlock,
                                [in] BSTR OperandName,
                                [out] unsigned long* RequestId);
```

参数

PLCName

用户程序中包含相应块的 PLC 的名称。该名称与 TIA Portal 项目树中显示的站名称相同。

ContainingBlock

要显示的块的名称，或 FB 实例的名称。

可用名称如下所示：

- 单个背景数据块的名称。随后显示其 FB。示例“Station1”
- 背景数据块中多重背景的名称。随后显示其 FB。指定多重背景名称路径后，将作为 DB 编辑器中显示的数据层级，而非调用结构。名称的第一部分（“Line1”）不需要括在引号中，因为可以从上下文中清楚地看出其是全局图标。对于单独的名称组件，如果包含空格、点等内容，则需要引号。示例：“Line1.Cell1.Station1”
- FC 或 OB 的名称

OperandName

要在属于“ContainingBlock”的代码块中搜索的块的名称。

- 对于局部操作符，此处必须指定井号 #。

返回值

用于对请求进行唯一标识的 ID。调用该函数时将生成并返回 ID。

参见

RequestByCallEx (页 2909)

StartDynamicDataSubscriptionEx (页 2917)

CancelRequest (页 2910)

3.12.2.6 RequestByCall

描述

该函数用于显示标准块程序段输入之前的逻辑。将返回包含输入针脚的所有程序段。

说明

此函数是一个遗留函数，可使用它调用简单注释。

声明

```
HRESULT RequestByCall( [in] BSTR PLCName,
                       [in] BSTR ContainingBlock,
                       [in] BSTR CalledBlock,
                       [in] BSTR PinName,
                       [in] boolean PartialPinNameMatch,
                       [out] unsigned long* RequestId);
```

参数

PLCName

用户程序中包含相应块的 PLC 的名称。该名称与 TIA Portal 项目树中显示的站名称相同。

ContainingBlock

要显示的块的名称，或 FB 实例的名称。

CalledBlock

“ContainingBlock”的程序段中调用的 FB 或 FC 的名称。或者，也可以组态一个 DB 或多重背景块。

可用名称如下所示：

- 单个背景数据块的名称。随后显示其 FB。示例“Station1”
- 背景数据块中多重背景的名称。随后显示其 FB。指定多重背景名称路径后，将作为 DB 编辑器中显示的数据层级，而非调用结构。名称的第一部分（“Line1”）不需要括在引号中，因为可以从上下文中清楚地看出其是全局图标。对于单独的名称组件，如果包含空格、点等内容，则需要引号。示例：“Line1.Cell1.Station1”
- FC 或 OB 的名称

PinName

"CalledBlock"输入针脚的名称。该参数用于确保与输入针脚互连的程序段包含在 XML 数据中。

PartialPinNameMatch

如果还考虑到以"PinName"下指定的名称为开头的输入针脚，则为 TRUE。

返回值

用于对请求进行唯一标识的 ID。调用该函数时将生成并返回 ID。

参见

RequestByAssignment (页 2905)

StartDynamicDataSubscription (页 2916)

CancelRequest (页 2910)

3.12.2.7 RequestByCallEx

说明

该函数用于显示标准块程序段输入之前的逻辑。将返回包含输入针脚的所有程序段。该函数返回层级注释。

声明

```
HRESULT RequestByCallEx( [in] BSTR PLCName,
                         [in] BSTR ContainingBlock,
                         [in] BSTR CalledBlock,
                         [in] BSTR PinName,
                         [in] boolean PartialPinNameMatch,
                         [out] unsigned long* RequestId);
```

参数

PLCName

用户程序中包含相应块的 PLC 的名称。该名称与 TIA Portal 项目树中显示的站名称相同。

ContainingBlock

要显示的块的名称，或 FB 实例的名称。

CalledBlock

"ContainingBlock"的程序段中调用的 FB 或 FC 的名称。或者，也可以组态一个 DB 或多重背景块。

可用名称如下所示：

- 单个背景数据块的名称。随后显示其 FB。示例"Station1"
- 背景数据块中多重背景的名称。随后显示其 FB。指定多重背景名称路径后，将作为 DB 编辑器中显示的数据层级，而非调用结构。名称的第一部分（"Line1"）不需要括在引号中，因为可以从上下文中清楚地看出其是全局图标。对于单独的名称组件，如果包含空格、点等内容，则需要引号。示例："Line1.Cell1.Station1"
- FC 或 OB 的名称

PinName

"CalledBlock"输入针脚的名称。该参数用于确保与输入针脚互连的程序段包含在 XML 数据中。

PartialPinNameMatch

如果还考虑到以"PinName"下指定的名称为开头的输入针脚，则为 TRUE。

返回值

用于对请求进行唯一标识的 ID。调用该函数时将生成并返回 ID。

参见

RequestByAssignmentEx (页 2906)

StartDynamicDataSubscriptionEx (页 2917)

CancelRequest (页 2910)

3.12.2.8 CancelRequest

说明

该函数用于取消进行中的请求。

声明

```
HRESULT CancelRequest( [in] unsigned long RequestId);
```

参数

RequestId

所取消请求的 ID。

返回值

-

参见

RequestByAssignment (页 2905)

RequestByCall (页 2908)

3.12.2.9 Terminate

描述

该函数用于结束执行通过“Initialize”实现的 COM 对象：

- 结束所有活动的请求和订阅。
- 结束回调函数“OnModeXML”、“OnDynamicDataXML”和“OnEvent”。

声明

```
HRESULT Terminate();
```

参数

-

返回值

-

参见

Initialize (页 2903)

3.12.2.10 XML 数据的结构

结构

```
<PlcCodeViewerExternalClientModel Version="1.0.0.0">
    <CommonHeaderInfo>
    </CommonHeaderInfo>
    <Networks>
        <Networks>
            <NetworkHeaderInfo>
            <NetworkHeaderInfo>
            <FlgNet>
                <Parts>
                    <Part>
                        <Parameters>
                        <Parameters>
                    </Part>
                </Parts>
                <Wires>
                    <Wire>
                    </Wire>
                </Wires>
            </FlgNet>
        <Networks>
    </Networks>
</PlcCodeViewerExternalClientModel>
```

默认情况下，可以在以下路径中找到 XSD 文件：
C:\Users\Public\Documents\Siemens\WinCC。

“<PlcCodeViewerExternalClientModel>”部分

包含：[1]

包含“CommonHeaderInfo”和“Networks”部分。

```
<PlcCodeViewerExternalClientModel Version="1.0.0.0">
   <CommonHeaderInfo>
          <HeaderInfo Description="PlcName" Text="Plc_1"/>
   </CommonHeaderInfo>
   <Networks>
          ….
   </Networks>
</PlcCodeViewerExternalClientModel>
```

“<Networks>”部分

包含：[0..n]

表示通过“NetId”属性进行唯一标识的程序段。包含“NetworkHeaderInfo”和“FlgNet”部分。

```
<Networks>
   <Network NetId="1">
          <NetworkHeaderInfo>
                 <HeaderInfo Description="FBNum" Text="3"/>
                 <HeaderInfo Description="FBName" Text="Block1"/>
                 <HeaderInfo Description="Title">
                        <DictEntry Language="DE-EN">First part of
                 Headers</DictEntry>
                 </HeaderInfo>
                 <HeaderInfo>
                        <DictEntry Language="DE-EN">Second part of
                 Headers</DictEntry>
                 </HeaderInfo>
          </NetworkHeaderInfo>

          <FlgNet Version="1.0.0.0" Lang="LAD">
                 <Parts>
                        ….
                 </Parts>
                 <Wires>
                        ….
                 </Wires>
          </FlgNet>
   </Network>
</Networks>
```

“<NetworkHeaderInfo>”部分

包含：[1]

包含程序段的相关信息，例如，指令名称或程序段标题。

“<FlgNet>”部分

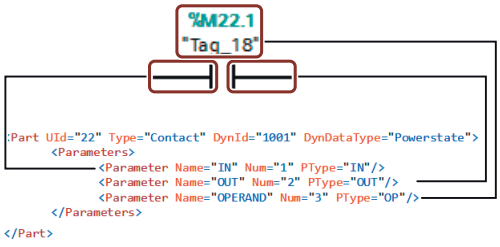
包含：[1]

表示程序段的描述。包含“Parts”和“Wires”部分。

“<Part>”部分

包含：[0..n]

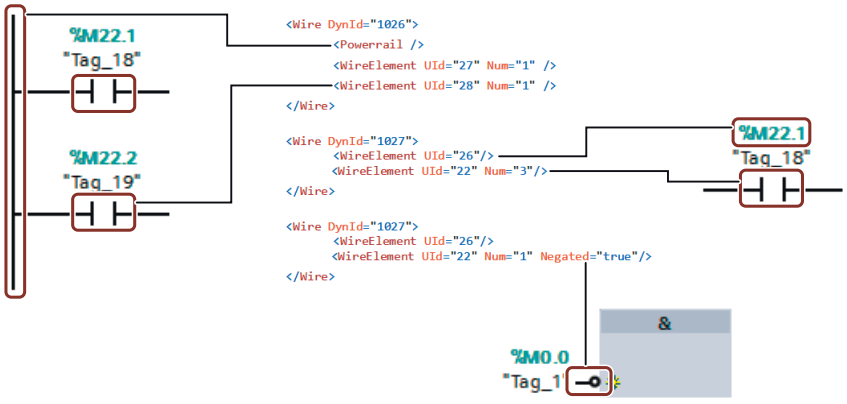
表示指令或操作数。



“<Wire>”部分

包含：[0..n]

表示指令和操作数之间的关联。



相关属性

下表列出了 XML 数据的属性。

属性	描述	包含于
AbsAdress	绝对地址	Operand
DynDataType	检索动态信息时为数据类型。具体取决于对象类型： • Operand – Bool：“True”或“False” • Part – Powerflow 当前数据流的状态：“On”、“Off”或“Undef” – Bool：“True”或“False” 参数或指令的状态： • Wire – Powerstate	Operand Part Wire
DynID	引用通过函数“StartDynamicDataSubscription”查询其动态数据的对象	Operand Part Wire
Name	对象名称。对于操作数，将显示符号地址。	Operand Parameter
Negated	如果输入取反，则为 True。	
Num	引用 Parameter。	Parameter
PType	定义针脚为输入或输出，或者表示开关状态。	Parameter
Type	定义类型。	
UId	引用对象。用于描述对象之间的关系（例如程序段）。	Operand Part Wire WireElement

3.12.3 查询程序段的动态统计信息

3.12.3.1 StartDynamicDataSubscription

描述

该函数用于查询指定程序段的周期性信息。该信息将整合到您事先通过“RequestByAssignment”或“RequestByCall”查询的 XML 数据中。由于支持多重静态订阅，因此可使用上次调用的订阅。

如果针对“DynIDCount”传递值“0”并针对“DynIDs”传递值“NULL”，则将输出所有已分配“DynIDs”的数据。要为“DynIDs”指定动态数据，则必须为“DynIdCount”和“DynIDs”传递相应 ID。

声明

```
HRESULT StartDynamicDataSubscription( [in] int NetworkID,
                                      [in] long DynIDCount,
                                   [in, [size_is(DynIDCount)] long* DynIDs,
                                      [out] long* SubscriptionId);
```

参数

NetworkID

静态数据先前被查询的程序段的 ID。该 ID 位于 XML 数据的“NetID”属性中。

DynIDCount

动态信息被查询的区域的数量。

DynIDs

当前数据流或当前值被显示的针脚或输入的 ID。这些 ID 位于 XML 数据的“DynID”属性中。

返回值

用于对订阅进行唯一标识的 ID。调用该函数时将生成并返回 ID。

参见

RequestByAssignment (页 2905)

RequestByCall (页 2908)

3.12.3.2 StartDynamicDataSubscriptionEx

说明

该函数用于查询指定程序段的周期性信息。该信息将整合到您事先通过“RequestByAssignment”或“RequestByCall”查询的 XML 数据中。借助该函数，用户可以通过“NetworkId” 和“RequestId”为请求申请静态订阅。

如果针对“DynIDCount”传递值“0”并针对“DynIDs”传递值“NULL”，则将输出所有已分配“DynIDs”的数据。要为“DynIDs”指定动态数据，则必须为“DynIdCount”和“DynIDs”传递相应 ID。

声明

```
HRESULT StartDynamicDataSubscriptionEx( [in] int NetworkID,
                                        [in] long DynIDCount,
                                  [in, [size_is(DynIDCount)] long* DynIDs,
                                        [out] long* SubscriptionId);
```

参数

NetworkID

静态数据先前被查询的程序段的 ID。该 ID 位于 XML 数据的“NetID”属性中。

DynIDCount

动态信息被查询的区域的数量。

DynIDs

当前数据流或当前值被显示的针脚或输入的 ID。这些 ID 位于 XML 数据的“DynID”属性中。

返回值

用于对订阅进行唯一标识的 ID。调用该函数时将生成并返回 ID。

参见

RequestByAssignmentEx (页 2906)

RequestByCallEx (页 2909)

3.12.3.3 StopDynamicDataSubscription

描述

该函数用于结束当前订阅。

声明

```
HRESULT StopDynamicSubscription( [in] long SubscriptionId);
```

参数

SubscriptionId
已结束订阅的 ID。

返回值

-

3.12.3.4 StopDynamicDataSubscriptionEx

说明

该函数用于结束使用“RequestId”和“SubscriptionId”定义的当前订阅。

声明

```
HRESULT StopDynamicSubscriptionEx( [in] unsigned long RequestId,
                                   [in] long SubscriptionId);
```

参数

RequestId

启动结束订阅的请求的 ID。

SubscriptionId

已结束订阅的 ID。

返回值

-

3.13 故障排除

3.13.1 CMN_ERROR

说明

扩展的错误结构 CMN_Error 包含已发生错误的错误代码和错误文本。每个应用都可以通过错误结构来判断或输出错误信息。

声明

```
Typedef struct {
     DWORD   dwError1;
     DWORD   dwError2;
     DWORD   dwError3;
     DWORD   dwError4;
     DWORD   dwError5;
     Char    szErrorText[512];
     }
     CMN_ERROR;
```

成员

dwError1 .. dwError5

每个 API 说明包含发生错误时有关条目值的信息。除非另有指定，否则错误代码存在于 dwError1 中。

szErrorText

有关错误原因文本说明的缓冲区。内容由资源决定，通常具有语言相关性。

注释

根据需要，一些模块使用下列错误代码分配

dwError1	API 错误代码
dwError2	附属错误代码（例如，CCStorageError.h）
dwError3	源代码的行号
dwError4	数组索引 1；行；记录/状态/....
dwError5	数组索引 2；列；记录元素/计数器/....

参见

错误消息 (页 1828)

3.13.2 CCStorageError.h

附属错误消息 CCStorageError.h

假如 dwError1 和 dwError2 在各自的模块中可用，则除了 dwError1 和 dwError2 中的实际错误消息外，下列错误消息可通过 CMN_ERROR 错误结构中的 API 函数返回。

错误消息	值	
CCF_STORAGE_PM_E_NO_PROJECT_OPEN	0x80046101L	无法打开 WinCC 项目。
CCF_STORAGE_PM_E_NO_PROJECT_OPEN_1	0x80046102L	无法打开/创建 WinCC 项目。项目管理器无法启动 SDIAGRT 基本应用程序。
CCF_STORAGE_PM_E_NO_PROJECT_OPEN_2	0x80046103L	无法打开/创建 WinCC 项目。项目管理器无法启动 PASSDBRTT 基本应用程序。
CCF_STORAGE_PM_E_NO_PROJECT_OPEN_3	0x80046104L	无法打开/创建 WinCC 项目。项目管理器无法启动 SCRIPT 基本应用程序。

错误消息	值	
CCF_STORAGE_PM_E_NO_PROJECT_OPEN_31	0x80046105L	无法打开/创建 WinCC 项目。项目管理器无法启动 XREF 基本应用程序。
CCF_STORAGE_PM_E_NO_PROJECT_OPEN_4	0x80046106L	无法打开/创建 WinCC 项目。项目管理器无法初始化 ASO 基本应用程序。
CCF_STORAGE_PM_E_NO_PROJECT_OPEN_5	0x80046107L	无法打开/创建 WinCC 项目。项目管理器一次只能打开一个 WinCC 项目。
CCF_STORAGE_PM_E_NO_PROJECT_OPEN_6	0x80046108L	无法打开/创建 WinCC 项目。项目管理器无法组态数据源的起始行。
CCF_STORAGE_PM_E_NO_PROJECT_OPEN_7	0x80046109L	无法打开/创建 WinCC 项目。项目管理器无法组态数据源。
CCF_STORAGE_PM_E_NO_PROJECT_OPEN_8	0x8004610AL	无法打开 WinCC 项目。已组态 WinCC 服务器的项目管理器尚未运行。启动已组态服务器的项目管理器。
CCF_STORAGE_PM_E_NO_PROJECT_OPEN_9	0x8004610BL	无法打开/创建 WinCC 项目。项目管理器无法创建分布式组态文件 (Project.dcf)。
CCF_STORAGE_PM_E_NO_PROJECT_OPEN_10	0x8004610CL	无法打开 WinCC 项目。 项目路径无效。
CCF_STORAGE_PM_E_NO_PROJECT_CLOSE_1	0x8004610DL	无法关闭 WinCC 项目。一些用户已通过网络连接到此 WinCC 项目。
CCF_STORAGE_PM_E_NO_PROJECT_CLOSE_2	0x8004610EL	无法关闭 WinCC 项目。项目管理器丢失有关此 WinCC 项目的特定信息。请检查项目 ID。
CCF_STORAGE_PM_E_NO_PROJECT_CLOSE_3	0x8004610FL	无法关闭 WinCC 项目。激活的 WinCC 项目已在 PC 上运行。

错误消息	值	
CCF_STORAGE_PM_E_NO_PROJECT_CLOSE_4	0x80046110L	无法关闭 WinCC 项目。项目管理器不支持此项目的关闭模式。
CCF_STORAGE_PM_E_NO_PROJECT_CLOSE_5	0x80046111L	无法关闭 WinCC 项目。一些用户已连接到 WinCC 项目。超时！
CCF_STORAGE_PM_E_NO_PROJECT_LANGUAGE_1	0x80046112L	项目不支持所请求的语言。
CCF_STORAGE_PM_E_NO_ACTIVATE	0x80046113L	没有激活的 WinCC 项目在此 PC 上运行。
CCF_STORAGE_PM_E_NO_ACTIVATE_1	0x80046114L	无法激活 WinCC 项目。 WinCC 服务器属性已更改。 关闭并再次打开 WinCC 项目。
CCF_STORAGE_PM_E_NO_PROJECT_CREATE	0x80046115L	无法创建 WinCC 项目。
CCF_STORAGE_PM_E_NO_PROJECT_OPEN_11	0x80046116L	无法打开 WinCC 项目。关闭正在运行的 WinCC 项目。
CCF_STORAGE_PM_E_NO_PROJECT_OPEN_12	0x80046117L	无法打开 WinCC 项目。 WinCC 项目路径超出允许的 255 个字符的最大长度。
CCF_STORAGE_PM_E_NO_PROJECT_OPEN_13	0x80046118L	无法打开 WinCC 项目。 WinCC 项目名称超出允许的 64 个字符的最大长度。
CCF_STORAGE_PM_E_NO_ACTIVATE_2	0x80046119L	无法禁用 WinCC 项目。超时时间间隔到期 (SCRIPT.exe)。
CCF_STORAGE_PM_E_NO_ACTIVATE_3	0x8004611AL	无法禁用 WinCC 项目。超时时间间隔到期 (PASSDBRT.exe)。
CCF_STORAGE_PM_E_NO_ACTIVATE_4	0x8004611BL	无法禁用 WinCC 项目。超时时间间隔到期 (SDIAGRT.exe)。
CCF_STORAGE_PM_E_NO_PROJECT_OPEN_14	0x8004611CL	无法打开 WinCC 项目。无法在 SIMATIC.cfg 文件中插入项目路径。

错误消息	值	
CCF_STORAGE_PM_E_NO_ACTIVATE_5	0x8004611DL	无法激活 WinCC 项目。不存在任何 WinCC 项目。
CCF_STORAGE_PM_E_NO_ACTIVATE_6	0x8004611EL	无法打开 WinCC 项目。 WinCC 客户端无法通过广播接收服务器激活状态。
CCF_STORAGE_PM_E_NO_ACTIVATE_7	0x8004611FL	正在激活 WinCC 项目。 WinCC 客户端激活尚未完成。
CCF_STORAGE_PM_E_CREATE_SHARE_FAILED	0x80046120L	无法创建 WinCC 项目共享。 无法打开 WinCC 项目。
CCF_STORAGE_PM_E_OPEN_FILE_FAILED	0x80046121L	无法打开 WinCC 项目文件。 无法打开 WinCC 项目。
CCF_STORAGE_PM_E_CLIENT_CANNOT_ACTIVATE	0x80046122L	由于客户端不包含在客户端列表中，因此该客户端无法激活此项目。
CCF_STORAGE_PM_E_SYBASE_MIGRATION	0x80046123L	此 WinCC 版本不支持较早版本的 Sybase 项目。移植项目
CCF_STORAGE_PM_E_NO_PROJECT_OPENED	0x80046124L	当前未打开任何项目。
CCF_STORAGE_PM_E_COMPUTER_NOT_IN_MACHINE_LIST	0x80046125L	此客户端不包含在指定服务器列表中。
CCF_SQL_SERVER_NOT_INSTALLED_ERROR	0x80046126L	SQL 服务器可能没有正确安装。检查 SQL 服务器是否已正确安装。
CCF_SQL_SERVER_DATABASE_ACCESS_ERROR	0x80046127L	SQL 服务器无法访问数据库。检查访问权限。
CCF_PM_E_PROJECT_IS_LOCKED	0x80046128L	项目已锁定。
CCF_SQL_SERVER_DATABASE_NTFS_ERROR	0x80046129L	SQL Server 2005 不支持压缩的 NTFS 卷。至少解压项目目录及其内容。
CCF_PM_OPEN_ERROR_RDP	0xC004612AL	此项目由 RDP 生成。不允许在 WinCC 中将其打开。
CCF_PM_OPEN_CODEPAGE_MISMATCH	0x4004612BL	此代码页与创建时的代码页不匹配。
CCF_PM_MIGRATION_NEEDED	0x8004612CL	所需的 PM 移植。

错误消息	值	
CCF_STORAGE_DBASO_E_NO_LICENSE	0x80046201L	此功能无可用的许可证。
CCF_STORAGE_DBASO_E_LIMIT	0x80046202L	已达到存储容量的上限。
CCF_STORAGE_DBASO_E_DATA_VERSION	0x80046203L	ASO 无法使用此文件版本。
CCF_STORAGE_DBASO_E_NO_CONNECTION	0x80046204L	未连接到数据库。
CCF_STORAGE_DBASO_E_NO_TABLE	0x80046205L	表不存在或无法打开。检查访问权限。
CCF_STORAGE_DBASO_E_NO_RECORD	0x80046206L	无数据记录
CCF_STORAGE_DBASO_E_NO_RESOURCE	0x80046207L	无资源
CCF_STORAGE_DBASO_E_NOT_CREATED	0x80046208L	未创建数据记录。
CCF_STORAGE_DBASO_E_NOT_DELETED	0x80046209L	未删除数据记录。
CCF_STORAGE_DBASO_E_NOT_MODIFIED	0x8004620AL	未更改数据记录。
CCF_STORAGE_DBASO_E_RECORD_EXIST	0x8004620BL	数据记录已存在。
CCF_STORAGE_DBASO_E_RECORD_IN_USE	0x8004620CL	数据记录已经使用。
CCF_STORAGE_DBASO_E_RECORD_FIELDSINVALID	0x8004620DL	计数或字段类型无效
CCF_STORAGE_DBASO_E_RECORD_PROTECTED	0x8004620EL	不能更改或移动数据记录。
CCF_STORAGE_DBASO_E_NO_CONTEXT	0x8004620FL	项目上下文不存在或无效。 ASO 未正常运行。
CCF_STORAGE_DBASO_E_ENUM_NOENTRIES	0x80046210L	未找到枚举的条目。
CCF_STORAGE_DBASO_E_NOT_FOUND	0x80046211L	无法找到指定的对象。
CCF_STORAGE_DBASO_E_ID_CHANGES_NOT_POSSIBLE	0x80046212L	禁止更改 ID。
CCF_STORAGE_DBASO_E_PARAMETER_TOO_LONG	0x80046213L	参数过长。
CCF_STORAGE_DBASO_E_DATA_TOO_OLD	0x80046214L	无法读取数据，因为数据来自较低版本的 WinCC。

错误消息	值	
CCF_STORAGE_DBASO_E_DATA_NEWER_THAN_SOFTWARE	0x80046215L	数据来自版本高于已安装软件版本的软件。请升级软件。
CCF_STORAGE_DBASO_E_ENUMSINK_RETURNED_WITH_ERROR	0x80046216L	枚举接收器执行时出错。
CCF_STORAGE_DBASO_E_DATA_EXIST_IN_REFERENCED_TABLE	0x80046217L	主键在另一个表中引用。
CCF_STORAGE_DBASO_E_DATA_OUT_OF_RANGE	0x80046218L	值超出取值范围。
CCF_STORAGE_DM_ASO_BULK_FAILED	0x80046219L	批处理失败。
CCF_STORAGE_DBASO_E_RT_LIMIT_WARNING	0x8004621AL	已达到运行系统的限值。
CCF_STORAGE_DBASO_E_BAD_LOCALE_ID	0x8004621BL	本地 ID 错误
CCF_STORAGE_FILEASO_E_NO_PICTURE	0x80046301L	无画面可用
CCF_STORAGE_GDO_BADTYPE_IN_VARIANT	0x80046405L	变体中的类型错误
CCF_STORAGE_GDO_DATAINCONSISTENT	0x80046406L	数据不一致： 有些数据无效或超出取值范围。
CCF_STORAGE_GDO_UNITID_INVALID	0x80046407L	数据不一致： 通道单元的 ID 无效。
CCF_STORAGE_GDO_USERTYPE_INVALID	0x80046408L	数据不一致： 用户类型无效。
CCF_STORAGE_GDO_ADDRESSBUILD_FAILURE	0x80046409L	数据不一致 变量寻址的形式不正确。
CCF_STORAGE_GDO_MODIFY_NOCHANGE	0x8004640AL	数据不一致： 未作出相应更改。
CCF_STORAGE_GDO_CHANNELWRAPPER_LOAD	0x8004640BL	无法找到 CCMfcChannelWrapper.DLL。
CCF_STORAGE_GDO_CHANNEL_LOAD	0x8004640CL	加载通道时出错： 未找到通道。
CCF_STORAGE_GDO_CHANNEL_FUNC	0x8004640DL	未找到通道入口点。

错误消息	值	
CCF_STORAGE_GDO_CHANNEL_UNITNR	0x8004640EL	通道单元的编号无效。
CCF_STORAGE_GDO_CONNECTION	0x8004640FL	连接无效或未找到。
CCF_STORAGE_GDO_TAGGROUP	0x80046410L	变量组无效或未找到。
CCF_STORAGE_GDO_NAMENOTFOUND	0x80046411L	未找到指定的名称。
CCF_STORAGE_GDO_TAGOFUSERTYPEEXISTS	0x80046412L	“用户类型”对象不可以移动。
CCF_STORAGE_GDO_PROJECTISACTIVE	0x80046413L	项目已激活。不允许更改或移动。
CCF_STORAGE_GDO_UNIT	0x80046414L	通道单元无效或未找到。
CCF_STORAGE_GDO_MINMAX_LIMIT	0x80046415L	超出下限或上限。
CCF_STORAGE_GDO_PROJECT_IN_RUNTIME	0x80046416L	当项目处于运行时不允许执行此过程。
CCF_STORAGE_GDO_BAD_POINTER	0x80046600L	不允许 NULL 指针指向 GetProperty。
CCF_STORAGE_GDO_UNEXPECTED	0x80046601L	发生意外错误。
CCF_STORAGE_GDO_NO_RECORD	0x80046602L	不存在数据记录。
CCF_STORAGE_GDO_OPERATION_NOT_COMMITTED	0x80046603L	不允许执行当前过程。
CCF_STORAGE_GDO_NO_PROJECT_CONNECTOR	0x80046604L	无可用的“项目连接器”(Project Connector)。
CCF_STORAGE_GDO_PROJECT_NOT_OPENED	0x80046605L	项目未打开。
CCF_STORAGE_GDO_GET_ASO_FAILED	0x80046606L	“GetAso”失败。
CCF_STORAGE_GDO_BAD_TYPEID	0x80046607L	类型 ID 不适合。
CCF_STORAGE_GDO_OTHER_TYPE_NOT_COMMITED	0x80046608L	不允许使用另一种数据类型。
CCF_STORAGE_GDO_CACHE_READ	0x80046609L	读取缓存时出错。
CCF_STORAGE_GDO_CACHE_WRITE	0x8004660AL	写入缓存时出错。
CCF_STORAGE_GDO_DM_GDO_ERROR	0x8004660BL	DM Gdo 中发生错误。
CCF_STORAGE_GDO_TEXT_GDO_ERROR	0x8004660CL	文本 Gdo 中发生错误。
CCF_STORAGE_GDO_ENUMERATOR_CREATION	0x8004660DL	无法创建 ASO 计数器。

错误消息	值	
CCF_STORAGE_GDO_ENUMDATA_FAILED	0x8004660EL	调用 ICCTxtAsoEnum::EnumData() 失败。
CCF_STORAGE_GDO_ENUMSINK_CREATION	0x8004660FL	创建枚举接收器对象失败。
CCF_STORAGE_GDO_WRONG_INTERFACE	0x80046610L	指针被发送到错误的接口。
CCF_STORAGE_GDO_BAD_LOCALE_ID	0x80046611L	在卡中未找到本地 ID。
CCF_STORAGE_GDO_CREATE_OBJECT	0x80046612L	CreateObject 失败。
CCF_STORAGE_GDO_MODIFY_OBJECT	0x80046613L	ModifyObject 失败。
CCF_STORAGE_GDO_READ_OBJECT	0x80046614L	ReadObject 失败。
CCF_STORAGE_GDO_NOT_IMPLEMENTED	0x80046615L	未执行。
CCF_STORAGE_GDO_REMOVE_OBJECT	0x80046616L	RemoveObject 失败。
CCF_STORAGE_GDO_REMOVE_ALL	0x80046617L	RemoveAll 失败。
CCF_STORAGE_GDO_ENUM_OBJECTS	0x80046618L	EnumObjects 失败。
CCF_STORAGE_GDO_GET_OBJECT_COUNT	0x80046619L	GetObjectCount 失败。
CCF_STORAGE_GDO_GET_RW_INTERFACE_FROM_ASO	0x8004661AL	ASO 的 GetRWInterface 失败。
CCF_STORAGE_GDO_ON_ENUM_DATA	0x8004661BL	“接收器”对象的 OnEnumData 失败。
CCF_STORAGE_GDO_ASO_CREATE	0x8004661CL	无法创建对 Aso 的调用。
CCF_STORAGE_GDO_CREATE_CACHED_OBJECT	0x8004661DL	CreateCachedObject 失败。
CCF_STORAGE_GDO_CREATE_SINK_OBJECT_FOR_ASO	0x8004661EL	Aso 的 CreateSinkObject 失败。
CCF_STORAGE_GDO_CREATE_DM_GDO	0x8004661FL	创建 DM Gdo 时出错。
CCF_STORAGE_GDO_CONNECT_DM_GDO	0x80046620L	连接 DM Gdo 时出错。
CCF_STORAGE_GDO_CREATE_TEXT_GDO	0x80046621L	创建文本 Gdo 时出错。
CCF_STORAGE_GDO_CONNECT_TEXT_GDO	0x80046622L	连接文本 Gdo 时出错。

错误消息	值	
CCF_STORAGE_GDO_INVALID_ENUM_SINK_POINTER	0x80046623L	枚举接收器指针出错。
CCF_STORAGE_GDO_VARIANT_CHANGE_TYPE	0x80046624L	Variant ChangeType 出错。
CCF_STORAGE_GDO_GET_COLLECT_ID	0x80046625L	GetCollectID 出错。
CCF_STORAGE_GDO_COMMIT_TO_TYPE_HANDLER	0x80046626L	CommitToTypeHandler 失败。
CCF_STORAGE_GDO_SET_TAG_ID	0x80046627L	SetTagId 出错。
CCF_STORAGE_GDO_SET_TEXT_ID	0x80046628L	SetTextId 出错。
CCF_STORAGE_GDO_COMMIT_CACHED_OBJECT	0x80046629L	CommitCachedObject 出错。
CCF_STORAGE_GDO_QUERY_INTERFACE_OF_SINK	0x8004662AL	接收器的 QueryInterface 失败。
CCF_STORAGE_GDO_BAD_PROPERTY_PARAMETER	0x8004662BL	为属性定义的参数无效。
CCF_STORAGE_GET_TAGNAME_FROM_ID	0x8004662CL	无法从变量 ID 中读取变量名称。
CCF_STORAGE_DELETE_RECORD_OF_CREATORID	0x8004662DL	无法删除创建者 ID 不等于 0 的数据。
CCF_STORAGE_CACHE_OVERFLOW	0x8004662EL	缓存溢出
CCF_STORAGE_SERVER_DOWN	0x8004662FL	服务器出现故障
CCF_STORAGE_ASO_PTR_NOT_INITIALIZED	0x80046630L	尚未初始化 Aso SmartPointer。
CCF_STORAGE_CREATE_CONNECTION_CONTROLLER	0x80046631L	无法创建连接控制器。
CCF_STORAGE_CONNECTION_CONTROLLER_CONNECT_TO_PROJECT	0x80046632L	连接控制器无法建立与项目的连接。
CCF_STORAGE_ASO_WRAPPER_SET_CONNECTION_CONTROLLER	0x80046633L	无法为 Aso wrapper 设置连接控制器。
CCF_STORAGE_ASO_WRAPPER_SET_PROJECT_STORAGE	0x80046634L	无法为 Aso wrapper 设置项目存储空间。
CCF_STORAGE_PROGID_FROM_CLSID	0x80046635L	ProgIDFromCLSID 出错。
CCF_STORAGE_ASO_WRAPPER_GET_ASO	0x80046636L	Aso wrapper 中“GetAso”调用出错。

错误消息	值	
CCF_STORAGE_ASO_WRAPPER_SET_PROJECT_STORAGE_NULL	0x80046637L	Aso wrapper 中 SetProjectStorage (NULL) 出错。
CCF_STORAGE_ASO_WRAPPER_GET_RWINTERFACE	0x80046638L	Aso wrapper 中 GetRWInterface 调用出错。
CCF_STORAGE_GDO_GET_PROJECT	0x80046639L	GdoCoreBase 中 GetProject 调用出错。
CCF_STORAGE_TLGGDO_VAR_NO_TRIGGER	0x8004663AL	没有可用的触发器
CCF_STORAGE_PACKAGES_PROJECT_ASO	0x80046501L	无法实例化项目 ASO。
CCF_STORAGE_PACKAGES_EXPORT_DATA	0x80046502L	数据导出失败。
CCF_STORAGE_PACKAGES_LIST	0x80046503L	枚举包失败。
CCF_STORAGE_CCPACKAGEMGR_EXPORT_IS_RUNNING	0x80046504L	正在导出数据。
CCF_STORAGE_CCPACKAGEMGR_UPDATE_IS_RUNNING	0x80046505L	正在更新数据。
CCF_STORAGE_PACKAGES_EXPORT_USER_CANCELED	0x80046506L	导出已被用户取消。
CCF_STORAGE_PACKAGES_IMPORT_ONLY_ONE_CAS_CAN_BE_IMPORTED	0x80046507L	仅可导入一个中央归档服务器。
CCF_STORAGE_PACKAGES_FORBID_STANDARD_SERVER	0x80046508L	不允许对此服务器类型执行标准服务器组态。
CCF_STORAGE_PACKAGES_IMPORT_IMPORT_RED	0x80046509L	不导入冗余服务器。
CCF_STORAGE_PACKAGES_IMPORT_CHANGE_RED	0x8004650AL	不更新冗余服务器的数据包。
CCF_STORAGE_PACKAGES_IMPORT_IMPORT_PHYS	0x8004650BL	不导入相同服务器的数据包。
CCF_STORAGE_PACKAGES_IMPORT_CHANGE_PHYS	0x8004650CL	不更新相同服务器的数据包。
CCF_STORAGE_PACKAGES_IMPORT_CHANGE_OWN	0x8004650DL	不导入或更新自己服务器的数据包。

错误消息	值	
CCF_STORAGE_PACKAGES_IMPORT_INCLUDE_OWN_RED	0x8004650EL	不导入或更新自己冗余服务器的数据包。
PCC_E_CANNOT_CONNECT_TO_DATABASE	0x80047000L	无法建立与数据库的连接。
PCC_E_RECONNECT_TO_DATABASE_FAILED	0x80047001L	重新连接到数据库失败。
PCC_E_JOB_ID_IS_INVALID	0x80047002L	作业 ID 无效。
PCC_E_INVALID_CONTEXT_INFO_PROVIDED	0x80047003L	上下文信息无效
PCC_E_DATABASE_ERROR_SEE_EVENT_LOG	0x80047004L	数据库出错。有关详细信息，请参见事件日志。
PCC_E_ERROR_SEE_EVENT_LOG	0x80047005L	未定义的错误，有关详细信息，请参见事件日志。
PCC_E_INVALID_PARAMETERS	0x80047006L	指定的参数无效。
PCC_E_ROLLBACK_TRANSACTION_FAILED	0x80047007L	事务无法撤消。
PCC_E_COMMIT_TRANSACTION_FAILED	0x80047008L	事务无法执行。
PCC_E_BEGIN_TRANSACTION_FAILED	0x80047009L	事务无法启动。
PCC_E_COPY_WORLDMAP_FAILED_WORLDMAP_LOCKED	0x8004700AL	无法复制全局映射。全局映射已锁定。
PCC_E_MODIFY_WORLDMAP_FAILED_WORLDMAP_LOCKED	0x8004700BL	无法更改全局映射。全局映射已锁定。
PCC_E_REMOVE_WORLDMAP_FAILED_WORLDMAP_LOCKED	0x8004700CL	无法移动全局映射。全局映射已锁定。
PCC_E_MODIFY_VIEW_FAILED_VIEW_LOCKED	0x8004700DL	无法更改视图。视图已锁定。
PCC_E_REMOVE_VIEW_FAILED_VIEW_LOCKED	0x8004700EL	无法移动视图。视图已锁定。
PCC_E_COPY_WORLDMAP_FAILED_VIEWS_LOCKED	0x8004700FL	无法复制全局映射。一个或多个视图已锁定。
PCC_E_MODIFY_WORLDMAP_FAILED_VIEWS_LOCKED	0x80047010L	无法更改全局映射。一个或多个视图已锁定。
PCC_E_REMOVE_WORLDMAP_FAILED_VIEWS_LOCKED	0x80047011L	无法删除全局映射。一个或多个视图已锁定。

错误消息	值	
PCC_E_COPY_WORLDMAP_FAILED_OBJECTS_LOCKED	0x80047012L	无法复制全局映射。一个或多个对象已锁定。
PCC_E_MODIFY_WORLDMAP_FAILED_OBJECTS_LOCKED	0x80047013L	无法更改全局映射。一个或多个对象已锁定。
PCC_E_REMOVE_WORLDMAP_FAILED_OBJECTS_LOCKED	0x80047014L	无法删除全局映射。一个或多个对象已锁定。
PCC_E_MODIFY_OBJECTS_FAILED_OBJECTS_LOCKED	0x80047015L	无法更改对象。一个或多个对象已锁定。
PCC_E_REMOVE_OBJECTS_FAILED_OBJECTS_LOCKED	0x80047016L	无法删除对象。一个或多个对象已锁定。
PCC_E_COPY_WORLDMAP_FAILED_NO_ACTIVATED_WORLDMAP	0x80047017L	全局映射复制作业失败。未激活任何全局映射。
PCC_E_REMOVE_WORLDMAP_FAILED_NO_ACTIVATED_WORLDMAP	0x80047018L	全局映射移动作业失败。未激活任何全局映射。
PCC_E_REMOVE_OBJECTS_FAILED_NO_ACTIVATED_OBJECTS	0x80047019L	对象移动作业失败。未激活任何对象。
PCC_E_MODIFY_OBJECTS_FAILED_NO_ACTIVATED_OBJECTS	0x8004701AL	对象更改作业失败。未激活任何对象。
PCC_E_REMOVE_VIEW_FAILED_NO_ACTIVATED_VIEW	0x8004701BL	视图更改作业失败。未激活任何视图。
PCC_E_MODIFY_VIEW_FAILED_NO_ACTIVATED_VIEW	0x8004701CL	全局映射更改作业失败。未激活任何作业。
PCC_E_MODIFY_WORLDMAP_FAILED_NO_ACTIVATED_JOB	0x8004701DL	全局映射更改作业失败。未激活任何作业。
PCC_E_COPY_WORLDMAP_FAILED_NOT_FOUND	0x8004701EL	无法复制全局映射。未找到全局映射。
PCC_E_WORLDMAP_NAME_IN_USE	0x8004701FL	全局映射的名称已经使用。
PCC_E_VERSION_NUMBER_INVALID	0x80047020L	版本号无效。
PCC_E_USER_DOES_NOT_HAVE_WRITE_PERMISSION	0x80047021L	用户不具备写入权限。
PCC_E_CREATE_AREALOCK_FAILED_OBJECTS_LOCKED	0x80047022L	无法创建区域锁定。存在已锁定的对象。
PCC_E_MODIFY_AREALOCK_FAILED_OBJECTS_LOCKED	0x80047023L	无法更改区域锁定。存在已锁定的对象。

错误消息	值	
PCC_E_JOB_NOT_FOUND	0x80047024L	未找到作业。
PCC_E_JOB_MUST_EXIST_MORE_THAN_ONCE	0x80047025L	作业必须存在多次。
PCC_E_CANNOT_MODIFY_ROOT_CTX	0x80047026L	无法更改根上下文。
PCC_E_CANNOT_MODIFY_JOB	0x80047027L	无法更改作业。
PCC_E_CANNOT_MODIFY_JOB_RESERVED_BY_OTHER_USER	0x80047028L	无法更改作业。该作业由另一个用户保留。
WINCC_DBPLUGIN_E_ROW_EXISTS	0x80047100L	行已存在。
WINCC_DBPLUGIN_E_CONSTRAINT_CONFLICT	0x80047101L	已发生约束冲突。
WINCC_DBPLUGIN_E_INVALID_OBJECT_NAME	0x80047102L	对象名称无效。
WINCC_DBPLUGIN_E_INVALID_COLUMN_NAME	0x80047103L	列名称无效。
WINCC_DBPLUGIN_E_DATA_CONVERSION	0x80047104L	数据转换过程中出错。
WINCC_DBPLUGIN_E_DATA_CONVERSION_DATETIME	0x80047105L	无法将字符串转换为日期/时间格式。
WINCC_DBPLUGIN_E_STRING_TOO_LONG	0x80047106L	字符串已缩短。
WINCC_DBPLUGIN_E_ARITHMETIC_OVERFLOW	0x80047107L	数据类型“溢出数量”无法保存该值。
WINCC_DBPLUGIN_RECORD_LOCKED	0x80047108L	数据记录已锁定。
WINCC_DBPLUGIN_TABLE_LOCKED	0x80047109L	表格已锁定。
WINCC_DBPLUGIN_CANNOTOPENDB	0x8004710AL	无法打开数据库。
PCC_E_CANNOT_DO_MAKE_UNIQUE_NAME	0x8004710BL	无法为对象分配唯一名称。
PCC_E_NO_DB_FOUND_ON_RTS	0x8004710CL	无法访问 RTS 主站上的数据库，或者该数据库处于备用模式。
PCC_E_OPERATION_NOT_ALLOWED_ON_RTS	0x8004710DL	RTS 数据库中不允许此过程。此过程必须在 PSOS 上的工程数据库中执行。
PCC_E_NO_DB_FOUND_ON_PSOS	0x8004710EL	无法访问 PSOS 上的数据库。

索引

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

安

按

保

报

备

编

变

查

从

错

打

导

登

对

发

方

数

停

通

统

外

系

显

线

向

校

增

置

终

注

转