

适用于 SIMATIC NET 的  
OPC 过程变量

用于 SIMATIC NET 的 OPC  
报警和事件服务器

## SIMATIC NET

# PC 软件 PG/PC 工业通信第 2 卷 - 接口

使用 OPC 服务器

SIMATIC 计算

示例程序

参考自动化接口

相关文献

## 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。

<b>△危险</b>
表示如果不采取相应的小心措施， <b>将会</b> 导致死亡或者严重的人身伤害。
<b>△警告</b>
表示如果不采取相应的小心措施， <b>可能</b> 导致死亡或者严重的人身伤害。
<b>△小心</b>
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。
<b>注意</b>
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：

<b>△警告</b>
<b>Siemens</b>
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 <b>Siemens</b>
推荐和允许。正确的运输、储存、组装、装配、安装、调试、操作和维护是产品安全、正常运行的前提。必须保证允许的环境条件。必须注意相关文件中的提示。

### 商标

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

### 责任免除

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

# 目录

1	前言 .....	19
1.1	如何使用本文档? .....	21
1.2	规定 .....	21
1.2.1	需要遵守哪些法律规定? .....	21
1.2.2	您需要知道哪些安全规定? .....	22
2	适用于 SIMATIC NET 的 OPC 过程变量 .....	23
2.1	存在哪些通信功能? .....	23
2.2	什么是过程变量? .....	23
2.3	过程变量的 ItemID 是如何构成的? .....	24
2.4	PROFIBUS DP .....	26
2.4.1	1 类 DP 主站 .....	27
2.4.1.1	适用于 PROFIBUS DP 协议的功能强大的 SIMATIC NET 进程内服务器 .....	27
2.4.1.2	适用于 PROFIBUS DP 协议的功能强大的 SIMATIC NET OPC 服务器 .....	28
2.4.1.3	DPC1 和 DPC2 服务 .....	30
2.4.1.4	1 类主站服务的过程变量 .....	31
2.4.1.5	1 类主站过程变量的语法 .....	32
2.4.1.6	协议 ID .....	33
2.4.1.7	经组态的 CP 名称 .....	34
2.4.1.8	1 类主站过程变量的示例 .....	34
2.4.1.9	DPC1 服务 .....	35
2.4.1.10	DPC1 服务的过程变量的语法 .....	35
2.4.1.11	DPC1 服务的过程变量的示例 .....	38
2.4.1.12	CP 5613/CP 5614 和 CP 5623/CP 5624 的快速逻辑 .....	38
2.4.1.13	快速逻辑的控制变量的语法 .....	39
2.4.1.14	DP 特定的信息变量 .....	41
2.4.1.15	DP 特定信息变量的语法 .....	41
2.4.1.16	DP 特定的信息变量的示例 .....	47
2.4.1.17	系统特定信息变量的语法 .....	47
2.4.2	2 类 DP 主站 .....	48
2.4.2.1	协议 ID .....	48
2.4.2.2	经组态的 CP 名称 .....	48
2.4.2.3	主站诊断的过程变量的语法 .....	49
2.4.2.4	从站诊断的过程变量的语法 .....	57
2.4.2.5	I/O 数据过程变量的语法 .....	64
2.4.2.6	数据记录过程变量的语法 .....	66
2.4.2.7	数据记录过程变量示例 .....	70
2.4.2.8	DP2 特定信息变量的语法 .....	71

---

2.4.2.9	系统特定信息变量的语法 .....	72
2.4.3	DP 从站 .....	73
2.4.3.1	用于访问本地从站数据的变量服务 .....	73
2.4.3.2	DP 从站过程变量的语法 .....	73
2.4.3.3	DP 从站过程变量示例 .....	75
2.4.3.4	DP 从站特定信息变量 .....	76
2.4.3.5	DP 从站特定信息变量的语法 .....	76
2.5	带 OPC UA 的 PROFIBUS DP .....	77
2.5.1	针对 DP 协议的 SIMATIC NET OPC UA 服务器 .....	78
2.5.2	支持带 OPC UA 的 DP 服务 .....	80
2.5.3	如何对 DP OPC UA 服务器进行寻址？ .....	82
2.5.4	DP OPC UA 服务器提供了哪些命名空间？ .....	85
2.5.5	Nodeld .....	86
2.5.6	DP 模块的板对象 .....	89
2.5.6.1	DP 模块的板对象概述 .....	89
2.5.6.2	板名称 .....	90
2.5.6.3	DP 主站板对象的类型定义 .....	91
2.5.6.4	DP 从站板对象的类型定义 .....	92
2.5.7	1 类 DP 主站 .....	93
2.5.7.1	1 类 DP 主站服务的过程变量 .....	93
2.5.7.2	1 类主站过程变量的语法 .....	93
2.5.7.3	1 类 DP 主站过程变量的示例 .....	96
2.5.7.4	DPC1 服务 .....	96
2.5.7.5	DPC1 服务的过程变量的语法 .....	97
2.5.7.6	DPC1 服务的过程变量的示例 .....	100
2.5.7.7	CP 5613/CP 5614/CP 5623/CP 5624 的快速逻辑（仅限主站） .....	100
2.5.7.8	快速逻辑的控制变量的语法 .....	101
2.5.7.9	控制帧同步和冻结 .....	103
2.5.7.10	同步和冻结控制变量的语法 .....	104
2.5.7.11	DP 特定的信息变量 .....	107
2.5.8	DP 从站 .....	121
2.5.8.1	用于访问本地 DP 从站数据的变量服务 .....	121
2.5.8.2	DP 从站过程变量的语法 .....	121
2.5.8.3	DP 从站的过程变量示例 .....	123
2.5.8.4	DP 从站的状态 .....	124
2.5.8.5	DP 从站的模式 .....	125
2.5.8.6	DP 从站特定信息变量 .....	125
2.5.8.7	DP 从站特定信息变量的语法 .....	125
2.5.9	DP OPC UA 模板数据变量 .....	129
2.5.9.1	DP OPC UA 模板数据变量 .....	129
2.6	S7 通信 .....	133
2.6.1	适用于 S7 协议的功能强大的 SIMATIC NET OPC 服务器 .....	134
2.6.2	协议 ID .....	136

2.6.3	连接名称.....	136
2.6.4	变量服务 .....	138
2.6.4.1	S7 变量服务的过程变量语法.....	138
2.6.4.2	S7 变量服务的过程变量的示例.....	144
2.6.4.3	具有最佳结构的数据项的示例.....	145
2.6.5	面向缓冲区的服务 .....	146
2.6.5.1	面向缓冲区的服务的过程变量语法.....	147
2.6.5.2	面向缓冲区的服务的过程变量示例.....	150
2.6.6	S7 特定的信息变量 .....	151
2.6.6.1	S7 特定的信息变量的语法 .....	151
2.6.6.2	S7 特定的诊断变量 .....	155
2.6.6.3	系统特定信息变量的语法 .....	161
2.6.7	块管理服务 .....	161
2.6.7.1	块管理服务的控制变量的语法.....	162
2.6.7.2	使用块管理服务的示例.....	166
2.6.8	密码 .....	168
2.6.8.1	密码的控制变量的语法.....	169
2.6.8.2	使用密码的示例.....	170
2.6.9	服务器服务 .....	170
2.6.9.1	使用服务器服务的示例.....	171
2.6.10	S7 模板数据变量 .....	173
2.6.10.1	命名空间中的模板数据变量.....	175
2.6.10.2	模板数据变量的语法 .....	175
2.6.11	未组态 S7 连接.....	177
2.6.12	COML S7 连接 .....	183
2.7	通过 OPC UA 进行的 S7 通信 .....	184
2.7.1	通过 OPC UA 进行的 S7 通信的属性 .....	184
2.7.2	S7 协议的 SIMATIC NET OPC UA 服务器.....	185
2.7.3	如何寻址 S7 OPC UA 服务器？ .....	188
2.7.4	S7 OPC UA 服务器提供了哪些命名空间？ .....	191
2.7.5	NodeId.....	192
2.7.6	连接对象 .....	195
2.7.6.1	连接名称 .....	196
2.7.7	生产 S7 连接对象的结构和函数 .....	197
2.7.7.1	S7 连接对象的类型定义 .....	197
2.7.7.2	S7 连接信息对象 .....	198
2.7.7.3	S7 特定的信息变量和返回值的示例 .....	199
2.7.7.4	S7 块服务的方法 .....	201
2.7.8	变量服务 .....	205
2.7.8.1	变量服务 .....	205
2.7.8.2	变量服务的语法 .....	205
2.7.8.3	S7 OPC UA 变量服务过程变量示例 .....	212
2.7.9	面向缓冲区的服务 .....	213
2.7.9.1	面向缓冲区的服务的语法 .....	213

---

2.7.9.2	面向缓冲区的服务的过程变量示例.....	216
2.7.10	S7 连接的块信息对象.....	218
2.7.10.1	长度信息.....	218
2.7.10.2	模板对象.....	219
2.7.10.3	诊断和组态信息.....	219
2.7.11	S7 OPC UA 模板数据变量.....	227
2.7.12	事件、条件和报警 .....	229
2.7.12.1	有哪些报警？ .....	229
2.7.12.2	什么是事件？ .....	229
2.7.12.3	S7 UA 报警服务器的事件有哪些？ .....	229
2.7.12.4	S7 UA 报警服务器的事件类型体系.....	230
2.7.13	标准事件类型 .....	233
2.7.13.1	显示名称为“BaseEventType”的标准事件类型 .....	233
2.7.13.2	显示名称为“ConditionType”的标准事件类型.....	240
2.7.13.3	显示名称为“AcknowledgeableConditionType”的标准事件类型 .....	243
2.7.13.4	显示名称为“AlarmConditionType”的标准事件类型 .....	243
2.7.13.5	显示名称为“ExclusiveLimitAlarmType”的标准事件类型 .....	245
2.7.13.6	显示名称为“ExclusiveLevelAlarmType”的标准事件类型 .....	246
2.7.13.7	显示名称为“ExclusiveDeviationAlarmType”的标准事件类型 .....	246
2.7.13.8	显示名称为“OffNormalAlarmType”的标准事件类型.....	246
2.7.14	S7 事件类型 .....	247
2.7.14.1	显示名称为“S7StatepathIAlarmType”的 S7 事件类型 .....	247
2.7.14.2	显示名称为“S7ExclusiveLevelAlarmType”的 S7 事件类型.....	248
2.7.14.3	显示名称为“S7ExclusiveDeviationAlarmType”的 S7 事件类型 .....	250
2.7.14.4	显示名称为“S7OffNormalAlarmType”的 S7 事件类型 .....	252
2.7.14.5	显示名称为“S7DiagnosisEventType”的 S7 事件类型 .....	254
2.7.15	区域树和源空间.....	255
2.7.16	接收事件.....	257
2.7.17	UA 报警方法.....	258
2.8	通过 OPC UA 进行的优化 S7 通信.....	258
2.8.1	通过 OPC UA 进行的优化 S7 通信的属性.....	258
2.8.2	支持优化 S7 协议的 SIMATIC NET OPC UA 服务器 .....	259
2.8.3	如何寻址 S7OPT OPC UA 服务器？ .....	262
2.8.4	S7OPT OPC UA 服务器提供了哪些命名空间？ .....	264
2.8.5	Nodeld .....	265
2.8.6	连接对象.....	267
2.8.7	连接名称.....	271
2.8.8	S7OPT 生产连接对象的结构和功能.....	272
2.8.9	S7OPT 连接对象的类型定义 .....	272
2.8.10	S7OPT 连接信息对象 .....	273
2.8.11	S7OPT 特定信息变量和返回值的示例.....	274
2.8.12	变量服务.....	275
2.8.12.1	标准访问.....	275
2.8.12.2	优化访问.....	284

2.8.13	S7 连接的块信息对象.....	290
2.8.13.1	长度信息.....	290
2.8.13.2	模板对象.....	291
2.8.13.3	诊断和组态信息.....	291
2.8.14	S7OPT OPC UA 模板数据变量.....	294
2.8.15	OPC UA 事件、条件和报警 .....	295
2.8.15.1	有哪些 OPC UA 报警? .....	295
2.8.15.2	什么是 OPC UA 事件? .....	296
2.8.15.3	S7OPT OPC UA 服务器支持哪些 S7OPT 事件类型? .....	296
2.8.15.4	S7OPT OPC UA 服务器的事件类型体系.....	297
2.8.16	标准事件类型及其属性的使用 .....	300
2.8.16.1	显示名称为“BaseEventType”的标准事件类型 .....	300
2.8.16.2	构成 SourceName、Message 和 Severity .....	302
2.8.16.3	显示名称为“ConditionType”的标准事件类型.....	305
2.8.16.4	ConditionName 的构成 .....	306
2.8.16.5	显示名称为“AcknowledgeableConditionType”的标准事件类型 .....	307
2.8.16.6	显示名称为“AlarmConditionType”的标准事件类型 .....	307
2.8.16.7	显示名称为“OffNormalAlarmType”的标准事件类型.....	308
2.8.17	S7OPT 事件类型 .....	309
2.8.17.1	显示名称为“S7OPTStatepathAlarmType”的 S7OPT 事件类型 .....	309
2.8.17.2	显示名称为“S7OPTConsistencyAlarmType”的 S7OPT 事件类型 .....	310
2.8.17.3	显示名称为“S7OPTOffNormalAlarmType”的 S7OPT 事件类型 .....	311
2.8.17.4	显示名称为“S7OPTInfoReportEventType”的 S7OPT 事件类型 .....	314
2.8.18	区域树和源空间.....	316
2.8.19	接收事件 .....	319
2.8.20	UA 报警方法.....	320
2.9	开放式通信服务 (SEND/RECEIVE) .....	321
2.9.1	通过工业以太网的开放式通信服务 (SEND/RECEIVE) .....	321
2.9.1.1	适用于 SR 协议的功能强大的 SIMATIC NET OPC 服务器 .....	321
2.9.1.2	协议 ID .....	323
2.9.1.3	连接名称 .....	323
2.9.1.4	变量服务 .....	324
2.9.1.5	面向缓冲区的服务 .....	327
2.9.1.6	SEND/RECEIVE 特定的信息变量 .....	331
2.9.1.7	系统特定信息变量的语法 .....	333
2.9.2	通过 PROFIBUS 的开放式通信服务 (SEND/RECEIVE) .....	333
2.9.2.1	协议 ID .....	334
2.9.2.2	连接名称 .....	334
2.9.2.3	面向缓冲区的服务 .....	334
2.9.2.4	FDL 特定的信息变量 .....	340
2.9.2.5	系统特定信息变量的语法 .....	343
2.10	通过工业以太网实现的 OPC UA 的开放式通信服务 (SEND/RECEIVE) .....	344
2.10.1	与 OPC UA 进行的 SR 通信的属性 .....	344

---

2.10.2	SR 协议的 SIMATIC NET OPC UA 服务器 .....	344
2.10.3	如何寻址 SR OPC UA 服务器？ .....	346
2.10.4	协议 ID .....	348
2.10.5	SR OPC UA 服务器提供了哪些命名空间？ .....	348
2.10.6	连接名称.....	349
2.10.7	Nodeld .....	350
2.10.8	S5 数据块和区域的数据变量（与 S5 兼容的通信） .....	352
2.10.9	面向缓冲区的服务 .....	357
2.10.10	SR 特定的信息变量.....	361
2.10.11	SR OPC UA 模板数据变量 .....	362
2.11	通过工业以太网实现的 SNMP 通信 .....	366
2.11.1	协议 ID .....	367
2.11.2	SNMP 协议的数据类型 .....	367
2.11.3	SNMP 变量服务的过程变量.....	368
2.11.4	SNMP 特定的信息变量 .....	369
2.11.5	SNMP 特定的陷阱 .....	372
2.12	通过工业以太网实现的 PROFINET IO 通信 .....	373
2.12.1	适用于 PROFINET IO 协议的功能强大的 SIMATIC NET OPC 服务器 .....	373
2.12.2	IO 数据的寻址方式是什么？ .....	375
2.12.3	如何浏览已组态的 PROFINET IO 命名空间？ .....	379
2.12.4	哪些 OPC 变量可用于 PROFINET IO？ .....	380
2.12.5	协议 ID .....	380
2.12.6	控制器名称 .....	380
2.12.7	PROFINET IO 过程变量 .....	380
2.12.8	PROFINET IO 数据状态 .....	386
2.12.9	PROFINET IO 数据记录 .....	390
2.12.10	系统特定信息变量的语法 .....	395
2.12.11	PROFINET IO 特定的信息变量 .....	395
2.13	PROFINET IO 通过工业以太网与 OPC UA 进行通信 .....	398
2.13.1	PROFINET IO 协议的 SIMATIC NET OPC UA 服务器 .....	398
2.13.2	如何对 PROFINET IO OPC UA 服务器进行寻址？ .....	401
2.13.3	PROFINET IO 与 OPC UA 之间的通信提供哪些命名空间？ .....	404
2.13.4	Nodeld .....	407
2.13.5	板对象和 PROFINET IO 控制器 .....	409
2.13.6	PROFINET IO 控制器的数据变量.....	411
2.13.7	设备对象 .....	412
2.13.8	设备对象的数据变量和方法 .....	414
2.13.9	PROFINET IO 模块对象 .....	415
2.13.9.1	PROFINET IO 模块的数据变量 .....	417
2.13.9.2	PROFINET IO 模块的 IO 数据变量.....	418
2.13.9.3	IOPS 和 IOCS 的数据变量.....	420
2.13.9.4	PROFINET IO 模块的数据记录数据变量 .....	425
2.13.10	PROFINET IO OPC UA 模板.....	428

2.13.10.1	模板数据变量 .....	428
2.13.10.2	模板数据变量的目录 .....	429
2.14	服务器诊断 .....	431
2.14.1	协议 ID .....	432
2.14.2	OPC DA 服务器诊断项 .....	432
2.15	具有 OPC 接口的面向缓冲区的服务 .....	437
2.15.1	使用缓冲区发送/接收服务 .....	437
2.15.2	面向缓冲区的通信的属性 .....	437
2.15.3	在 OPC 变量上映射数据缓冲区 .....	438
2.15.4	使用面向缓冲区的服务 .....	439
2.15.5	使用带有 TCP/IP 本地的面向缓冲区的服务时要注意的几个事项 .....	440
2.16	限制 OPC 变量的访问权限 .....	440
<b>3</b>	<b>用于 SIMATIC NET 的 OPC 报警和事件服务器 .....</b>	<b>445</b>
3.1	用于 S7 通信的事件服务器 .....	445
3.1.1	功能和报警类别 .....	445
3.1.2	事件参数 .....	449
3.1.3	事件属性 .....	452
3.1.4	模块诊断缓冲区中的条目属性 .....	465
3.1.5	指示连接中断的报警属性 .....	470
3.1.6	组态报警文本、源和区域 .....	472
3.1.7	如何在命名空间内浏览和过滤源信息、源和条件？ .....	478
3.1.8	将 STEP 7 组态值映射到 OPC S7 报警和事件参数 .....	480
3.1.9	S7 演示报警 .....	482
3.2	SNMP 通信简单事件服务器 .....	483
3.3	S7 和 SNMP 通信的报警和事件服务器 .....	487
3.3.1	SIMATIC NET 的报警和事件服务器 .....	487
3.3.2	服务器的名称 (ProgID) 是什么？ .....	488
3.3.3	如何检测报警源的协议？ .....	489
<b>4</b>	<b>使用 OPC 服务器 .....</b>	<b>493</b>
4.1	编程自动化接口 .....	493
4.1.1	对用于数据访问的自动化接口进行编程 .....	493
4.1.1.1	OPC 数据访问的对象模型都提供了什么？ .....	493
4.1.1.2	编程时需记住的要点 .....	494
4.1.1.3	用于数据访问的自动化接口的对象 .....	495
4.1.2	对用于报警和事件的自动化接口进行编程 .....	505
4.1.2.1	OPC 报警和事件的对象模型都提供了什么？ .....	505
4.1.2.2	编程时需记住的要点 .....	507
4.1.2.3	用于报警和事件的自动化接口的对象 .....	507
4.2	对自定义接口进行编程 .....	517
4.2.1	创建 COM 对象并查询 OPC 服务器的状态 .....	519

---

4.2.2	用于数据访问的自定义接口的对象.....	519
4.2.2.1	OPCServer 对象 .....	519
4.2.2.2	OPCGroup 对象.....	525
4.2.3	用于报警和事件的自定义接口的对象.....	535
4.2.3.1	OPCEventServer 对象.....	535
4.2.3.2	OPCEventSubscription 对象.....	538
4.2.3.3	OPCEventAreaBrowser 对象.....	541
4.2.4	报警和事件客户端的接口 .....	541
4.2.4.1	IOPCEventSink 接口.....	541
4.2.4.2	IOPCShutdown 接口 .....	542
4.2.5	OPC DA 过程变量的错误消息 .....	543
4.3	对 XML 接口进行编程 .....	548
4.3.1	元素的描述 .....	550
4.3.2	基本模式.....	551
4.3.2.1	ItemProperty .....	552
4.3.2.2	ItemValue.....	553
4.3.2.3	OPCError .....	554
4.3.2.4	ReplyBase.....	555
4.3.2.5	RequestOptions .....	556
4.3.3	读取和写入作业.....	558
4.3.3.1	读取 .....	558
4.3.3.2	ReadResponse .....	560
4.3.3.3	写入 .....	563
4.3.3.4	WriteResponse .....	566
4.3.4	使用订阅.....	567
4.3.4.1	订阅 .....	569
4.3.4.2	SubscribeResponse.....	571
4.3.4.3	SubscriptionPolledRefresh .....	573
4.3.4.4	SubscriptionPolledRefreshResponse .....	574
4.3.4.5	SubscriptionCancel.....	575
4.3.4.6	SubscriptionCancelResponse .....	576
4.3.5	更多查询.....	577
4.3.5.1	Browse .....	577
4.3.5.2	BrowseResponse.....	579
4.3.5.3	GetProperties .....	581
4.3.5.4	GetPropertiesResponse .....	582
4.3.5.5	GetStatus .....	583
4.3.5.6	GetStatusResponse.....	585
4.4	对 OPC UA 接口进行编程.....	586
4.4.1	对 OPC UA 服务器进行组态 .....	587
4.4.1.1	验证 .....	587
4.4.1.2	端点安全.....	588
4.4.2	OPC UA 服务器的证书管理 .....	589

4.4.3	OPC UA 客户端“OPC Scout V10”中的证书管理 .....	592
4.4.4	使用图形用户界面进行证书管理 .....	597
4.4.5	OPC UA 服务 .....	599
4.4.6	创建 OPC UA 客户端 .....	603
4.4.6.1	OPC UA 下的接口 .....	603
4.4.6.2	OPC UA 的 C 接口 .....	603
4.4.6.3	OPC UA 的 .NET 接口 .....	604
4.4.7	OPC UA 服务器的消息 .....	605
4.4.8	从 OPC 数据访问/报警和事件移植到 OPC UA .....	607
4.5	对 OPC UA 冗余接口进行编程 .....	608
4.5.1	安装和组态“网络负载平衡”功能 .....	612
4.5.2	簇组态 .....	618
4.5.3	使用发现服务器定位服务器端点 .....	627
4.5.4	OPC UA 重新连接步骤 .....	627
4.5.5	OPC UA 客户端支持的冗余 .....	629
4.5.6	什么时候会发生服务器故障切换？ .....	630
4.5.7	服务器故障切换的详细信息 .....	631
4.5.8	SIMATIC NET OPC UA S7 A&C 对冗余故障切换的响应 .....	633
4.5.9	有关 OPC UA S7 通信的常规建议和注意事项 .....	635
4.5.10	由 OPC UA 客户端读出簇的状态 .....	636
5	<b>SIMATIC 计算 .....</b>	<b>641</b>
5.1	.NET OPC 客户端控件 .....	642
5.1.1	概述 .....	642
5.1.2	步骤 1 - 安装 .....	643
5.1.2.1	引用控件 .....	643
5.1.2.2	设置示例程序 .....	646
5.1.3	步骤 2 - WINDOWS FORM 中的控件 .....	648
5.1.3.1	添加 Windows 控件 .....	651
5.1.3.2	添加 .NET OPC 客户端控件 .....	653
5.1.4	步骤 3 - 组态数据链接 .....	654
5.1.4.1	浏览 OPC 数据项 .....	655
5.1.4.2	浏览 Windows 控件属性 .....	658
5.1.4.3	创建链接列表 .....	659
5.1.4.4	用于写入的触发器 .....	661
5.1.4.5	OPC 质量和错误 .....	665
5.1.5	步骤 4 - 启动示例程序 .....	668
5.1.5.1	调试和发布 .....	668
5.1.5.2	分发应用程序 .....	668
5.1.6	故障排除 .....	669
5.1.7	为 .NET OPC 数据控件创建 OPC UA 证书 .....	671
5.1.8	连接至 OPC UA 服务器时的 OPC UA 证书 .....	671
5.2	.NET OPC 客户端 API .....	673
5.2.1	SIMATICNET.OPCDAClient 命名空间 .....	674

5.2.1.1	数据模型的类别.....	674
5.2.1.2	DaServerMgt 对象的接口.....	689
5.2.1.3	故障排除.....	710
5.2.2	命名空间 SIMATICNET.OPCDACLIENT 的示例.....	711
5.2.3	SIMATICNET.OPCCMN 命名空间.....	715
5.2.3.1	OPCServerEnum 对象的接口.....	715
5.2.3.2	ServerIdentifier 类.....	718
5.2.3.3	ServerCategory 枚举器.....	718
5.2.3.4	EndpointIdentifier 类.....	719
5.2.3.5	PkiCertificate 类.....	719
5.2.3.6	创建新证书.....	720
5.2.3.7	toDER 方法.....	721
5.2.3.8	fromDER 方法.....	721
5.2.3.9	toWindowsStore 方法.....	722
5.2.3.10	toWindowsStoreWithPrivateKey 方法.....	722
5.2.3.11	fromWindowsStore 方法.....	723
5.2.3.12	fromWindowsStoreWithPrivateKey 方法.....	723
5.2.3.13	fromWindowsStoreWithPrivateKey 方法.....	724
5.2.3.14	属性.....	725
5.2.3.15	枚举器 WinStoreLocation .....	725
6	示例程序.....	729
6.1	VB.NET 中的 OPC 自动化接口（同步通信）.....	729
6.1.1	激活仿真连接 .....	729
6.1.2	使用示例程序 .....	730
6.1.3	如何运行程序？ .....	732
6.1.4	将 OPC 自动化接口与 .NET-Framework 配合使用 .....	732
6.2	C++ 中的 OPC 自定义接口（同步通信）.....	735
6.2.1	激活仿真连接 .....	736
6.2.2	使用示例程序 .....	736
6.2.3	如何运行程序？ .....	738
6.2.4	OPCDA_SyncDlg.cpp 程序的说明.....	741
6.2.4.1	OnInitDialog .....	742
6.2.4.2	OnStart.....	743
6.2.4.3	OnRead.....	753
6.2.4.4	OnWrite.....	755
6.2.4.5	OnStop.....	757
6.2.4.6	DestroyWindow.....	760
6.2.4.7	GetQualityText.....	760
6.2.5	有关编写您自己的程序的注意事项.....	761
6.3	C++ 中的 OPC 自定义接口（异步通信）.....	762
6.3.1	激活仿真连接 .....	762
6.3.2	使用示例程序 .....	763
6.3.3	启动程序.....	763

6.3.4	读取和写入数值.....	763
6.3.5	激活组 .....	764
6.3.6	停止程序.....	765
6.3.7	如何运行程序？ .....	765
6.3.8	程序说明.....	772
6.3.9	“OPCDA_AsyncDlg.cpp”文件.....	773
6.3.10	Callback.cpp 和 Callback.h.....	798
6.3.11	有关编写您自己的程序的注意事项.....	804
6.4	以 VB.NET 编写的 OPC 自定义接口（异步通信） .....	805
6.4.1	使用示例程序 .....	805
6.4.2	程序说明.....	806
6.5	以 C# 编写的 OPC 自定义接口（同步通信） .....	812
6.5.1	使用示例程序 .....	813
6.5.2	程序说明.....	814
6.6	以 C# 编写的 OPC 自定义接口（异步通信） .....	817
6.6.1	使用示例程序 .....	818
6.6.2	程序说明.....	819
6.7	以 C# 编写的 OPC XML 接口 .....	824
6.7.1	使用示例程序 .....	825
6.7.2	向项目添加 Web 服务 .....	827
6.7.3	MainForm 类 .....	828
6.7.4	MainForm 的构造函数和 Dispose 方法.....	830
6.7.5	创建对话框元素.....	831
6.7.6	Main 方法 .....	837
6.7.7	Button_Start_Sample_Click 方法.....	838
6.7.8	Button_Stop_Sample_Click 方法 .....	841
6.7.9	Button_Read_Value_Click 方法 .....	841
6.7.10	Button_Write_Value_Click 方法 .....	844
6.8	以 C++ 编写的 OPC 报警和事件自定义接口 .....	846
6.9	以 C 编写的 OPC UA 接口.....	849
6.9.1	激活仿真连接 .....	850
6.9.2	导入客户端证书.....	850
6.9.3	使用示例程序 .....	851
6.9.4	启动程序.....	851
6.9.5	读取和写入值 .....	851
6.9.6	监控变量.....	852
6.9.7	停止程序.....	852
6.9.8	程序顺序的说明.....	853
6.9.8.1	连接建立.....	853
6.9.8.2	读取和写入变量.....	854
6.9.8.3	监控变量和报警.....	855
6.9.8.4	终止连接.....	855

---

6.9.9	有关转换为实际变量的注意事项 .....	856
6.10	C# 中的 OPC UA 接口（异步通信） .....	856
7	参考自动化接口 .....	857
7.1	常规信息 .....	857
7.1.1	什么是接口？ .....	858
7.1.2	OPC 的两种接口类型 .....	858
7.1.3	COM/OLE 对象 .....	859
7.1.4	集合对象 .....	859
7.1.5	对象模型 .....	860
7.1.6	数据同步 .....	861
7.1.7	例外 .....	861
7.1.8	事件 .....	862
7.1.9	数组 .....	862
7.1.10	参数 .....	862
7.1.11	类型库 .....	862
7.2	OPCServer 对象 .....	863
7.2.1	OPCServer 对象的属性 .....	864
7.2.1.1	StartTime .....	864
7.2.1.2	CurrentTime .....	865
7.2.1.3	LastUpdateTime .....	865
7.2.1.4	MajorVersion .....	866
7.2.1.5	MinorVersion .....	866
7.2.1.6	BuildNumber .....	867
7.2.1.7	VendorInfo .....	867
7.2.1.8	ServerState .....	868
7.2.1.9	LocaleID .....	869
7.2.1.10	Bandwidth .....	870
7.2.1.11	OPCGroups .....	870
7.2.1.12	PublicGroupNames .....	871
7.2.1.13	ServerName .....	871
7.2.1.14	ServerNode .....	872
7.2.1.15	ClientName .....	873
7.2.2	OPCServer 对象的方法 .....	873
7.2.2.1	GetOPCServers .....	874
7.2.2.2	Connect .....	874
7.2.2.3	Disconnect .....	876
7.2.2.4	CreateBrowser .....	877
7.2.2.5	GetErrorString .....	877
7.2.2.6	QueryAvailableLocaleIDs .....	878
7.2.2.7	QueryAvailableProperties .....	879
7.2.2.8	GetItemProperties .....	880
7.2.2.9	LookupItemIDs .....	881
7.2.3	OPCServer 对象的事件 .....	883

7.2.3.1	ServerShutDown.....	883
7.3	集合对象 OPCGroups.....	884
7.3.1	OPCGroups 对象的属性.....	885
7.3.1.1	Parent.....	885
7.3.1.2	DefaultGroupIsActive.....	885
7.3.1.3	DefaultGroupUpdateRate .....	886
7.3.1.4	DefaultGroupDeadband.....	886
7.3.1.5	DefaultGroupLocaleID .....	887
7.3.1.6	DefaultGroupTimeBias .....	888
7.3.1.7	Count.....	888
7.3.2	OPCGroups 对象的方法.....	889
7.3.2.1	Item .....	889
7.3.2.2	Add.....	890
7.3.2.3	GetOPCGroup .....	891
7.3.2.4	Remove.....	891
7.3.2.5	RemoveAll.....	892
7.3.2.6	ConnectPublicGroup.....	893
7.3.2.7	RemovePublicGroup.....	893
7.3.3	OPCGroups 对象的事件.....	894
7.3.3.1	GlobalDataChange .....	894
7.4	OPCGroup 对象 .....	896
7.4.1	OPCGroup 对象的属性.....	898
7.4.1.1	Parent.....	898
7.4.1.2	Name.....	898
7.4.1.3	IsPublic.....	899
7.4.1.4	IsActive.....	900
7.4.1.5	IsSubscribed .....	900
7.4.1.6	ClientHandle .....	901
7.4.1.7	ServerHandle .....	902
7.4.1.8	LocaleID .....	903
7.4.1.9	TimeBias .....	904
7.4.1.10	DeadBand .....	904
7.4.1.11	UpdateRate .....	905
7.4.1.12	OPCItems .....	906
7.4.2	OPCGroup 对象的方法.....	907
7.4.2.1	SyncRead .....	907
7.4.2.2	SyncWrite.....	909
7.4.2.3	AsyncRead.....	911
7.4.2.4	AsyncWrite.....	913
7.4.2.5	AsyncRefresh.....	915
7.4.2.6	AsyncCancel .....	917
7.4.3	OPCGroup 对象的事件.....	918
7.4.3.1	DataChange .....	918

---

7.4.3.2	AsyncReadComplete .....	919
7.4.3.3	AsyncWriteComplete .....	920
7.4.3.4	AsyncCancelComplete .....	921
7.5	集合对象 OPCItems .....	922
7.5.1	OPCItems 集合对象的属性 .....	923
7.5.1.1	Parent .....	923
7.5.1.2	DefaultRequestedDataType .....	923
7.5.1.3	DefaultAccessPath .....	924
7.5.1.4	DefaultIsActive .....	925
7.5.1.5	Count .....	925
7.5.2	OPCItems 集合对象的方法 .....	926
7.5.2.1	Item .....	926
7.5.2.2	GetOPCItem .....	926
7.5.2.3	AddItem .....	927
7.5.2.4	AddItems .....	928
7.5.2.5	Remove .....	929
7.5.2.6	Validate .....	930
7.5.2.7	SetActive .....	931
7.5.2.8	SetClientHandles .....	932
7.5.2.9	SetDataTypes .....	933
7.6	OPCItem 对象 .....	934
7.6.1	OPCItem 对象的属性 .....	935
7.6.1.1	Parent .....	935
7.6.1.2	ClientHandle .....	935
7.6.1.3	ServerHandle .....	936
7.6.1.4	AccessPath .....	937
7.6.1.5	AccessRights .....	937
7.6.1.6	ItemID .....	938
7.6.1.7	IsActive .....	938
7.6.1.8	RequestedDataType .....	939
7.6.1.9	Value .....	940
7.6.1.10	Quality .....	940
7.6.1.11	TimeStamp .....	941
7.6.1.12	CanonicalDataType .....	942
7.6.1.13	EUType .....	942
7.6.1.14	EUIInfo .....	943
7.6.2	OPCItem 对象的方法 .....	944
7.6.2.1	Read .....	944
7.6.2.2	Write .....	945
7.7	定义 .....	946
7.7.1	服务器状态 .....	946
7.7.2	错误消息 .....	947
7.8	自动化接口参考附录 .....	949

8	相关文献 .....	951
8.1	OPC 规范 .....	953



# 前言

## SIMATIC NET OPC 服务器 V13 中的新增功能

在 OPC 统一架构版本 1.01 中，先前的功能已得到增强和扩展。

SIMATIC NET OPC UA 服务器还支持：

- S7OPT UA 报警

本手册中的新增功能

- 在“优化的 S7 通信”部分添加了有关 S7OPT UA 报警的说明。

## SIMATIC NET - 开创书面形式的成功解决方案之先河

本文档可作为成功应用 SIMATIC NET 的指南。

其中全面介绍了各个主题，还说明了如何安装和组态各个组件以及如何基于 OPC 创建程序。您将看到 SIMATIC NET

工业通信将为您、您的自动化解决方案带来契机；最重要的是，将为您的公司开启成功之门。

## SIMATIC NET - 正确的决定

您了解了分布式自动化系统的优点并希望优化使用工业通信。

您期待与实力雄厚的合作伙伴合作并生产出可靠的创新产品。SIMATIC NET 是您正确的选择。

本文档根据您所掌握的基础知识，帮助您利用各位专家的专有技术获得利润。

## 如果您是初学者

利用本手册可系统性地熟悉相关内容。从第 1 卷工业通信简介开始入手。第 1 卷中提供了有关 SIMATIC NET OPC 服务器的通信原理和功能范围的所有必要信息。熟读 OPC 接口的基础知识，熟悉 SIMATIC NET 通信协议及其优势和功能。

## 1.1 如何使用本文档?

### 如果您是专业人员

利用本手册可立即开始工作。第 2 卷提供了使用 SIMATIC NET 所需的所有信息。

### 您是否发现示例很实用？

可灵活利用所提供的示例程序将您自己的想法付诸实践。

### SIMATIC NET 词汇表

在 SIMATIC NET 词汇表部分针对本文档中所用的专业术语进行了解释。

用户可在以下位置找到 SIMATIC NET 词汇表：

- SIMATIC NET 手册集

该 DVD 随一些 SIMATIC NET 产品一起提供。

- 请参见 Internet 上的以下条目 ID:

50305045 (<http://support.automation.siemens.com/WW/view/zh/50305045>)

### 安全消息

#### 说明

Siemens 为其自动化和驱动器产品系列提供 IT

安全机制，从而支持设备/机器的安全运行。根据 IT

安全准则，我们将对这些产品进行进一步的研发。

因而，我们建议您定期检查我们产品的更新情况，确保只使用最新的版本。可以在以下链接中找到相关信息：

(<http://support.automation.siemens.com/WW/lisapi.dll?func=cslib.csinfo2&aktprim=99&lang=zh>)

在此，您可以通过注册获取产品特定的新闻快递。

不过，为了确保设备/机器的安全运行，还需将该自动化组件集成到整个设备/机器的全面的 IT 安全方案中，此方案基于最新的 IT 技术。可以在以下链接中找到相关信息：

(<http://www.siemens.com/industrialsecurity>)

也必须考虑要使用的其他制造商的产品。

## 1.1 如何使用本文档？

为帮助您使用本文档，本文档中使用了各种约定和符号。

这些约定和符号旨在突出显示不同类型的信息，便于您迅速获取所需信息。

### 约定

本文档使用各种约定突出显示不同类型的信息，让您能够立即找到特定类型的信息。

下表显示了所使用的约定。

约定	表示：
斜体	突出显示的表达式或项目、变量、程序、对话框等的名称。
<尖括号中的文本>	必须用当前相关术语替换的<可变术语>。
在文本列开始处以粗体显示的术语	下文中介绍的语法元素。
以 Courier New 字体显示的文本	此为程序代码。
{大括号中的语法文本}	语法说明中的可选信息
"引号中的语法文本"	语法说明中的可变术语

## 1.2 规定

必须遵守以下与 SIMATIC NET 产品相关的规定：

### 1.2.1 需要遵守哪些法律规定？

我们必须指出，本文档的内容不能成为任何以前或现有协议、承诺或法律关系的一部分，也不能对以前或现有部分进行修改。采购协议包含了 Siemens 的完整专属义务。

本文档中所包含的任何陈述都不会创建新的担保或限制现有担保。

出于简明原因，我们还要进一步指出，本文档不能处理因使用 OPC 服务器而产生的所有可能出现的问题。

如果您还需要更多信息，或者所产生的特殊问题在本文档中不能得到充分解决，请与当地 Siemens 代表取得联系。

## 免责声明

我们已检查过本文档中的内容与所描述的硬件和软件相符。  
由于差错在所难免，我们不能保证完全一致。  
但我们会定期审查本文档中的内容，并在后续版本中进行必要的更正。  
欢迎提出改进建议。  
技术数据如有改动，恕不另行通知。  
未经明确的书面许可，不得复制、传播或使用本手册或所含内容。  
违者应对造成的损失承担责任。  
保留所有权利，包括实用新型或外观设计的专利许可权及注册权。

Siemens AG  
IIA SC CI  
Postfach 4848  
D-90026 Nürnberg

### 1.2.2 您需要知道哪些安全规定？

#### 合格人员

只允许合格人员安装和使用本产品。  
合格人员是指被授权按照既定安全惯例和标准，对线路、设备和系统进行调试、接地和标记的人员。

#### 正确使用



该设备及其部件只能用于相关产品目录或技术说明书中所描述的应用领域，并且只能与由 **Siemens** 认可或推荐的其他制造商提供的设备或部件一起使用。

只有正确地运输、存储、设置和安装本产品，并且按照推荐的方式操作和维护，产品才能正常、安全地运行。

使用本文档中介绍的产品，可以轻松访问和修改过程数据。

修改过程数据会导致无法预料的过程反应，从而导致人员死亡、重伤和/或设备损坏。

操作时要多加小心，确保不会访问可能造成受控设备和装置产生意外反应的数据。

# 适用于 SIMATIC NET 的 OPC 过程变量

本章说明过程变量名称的语法。这些名称指定在 OPC 接口上对哪些过程值进行寻址。编程或组态 OPC 客户端时指定变量名称。

## 2.1 存在哪些通信功能？

OPC 服务器提供了对 SIMATIC NET 工业通信网络的标准化访问。

SIMATIC NET OPC 服务器支持应用程序与任何通过 PROFIBUS 或工业以太网联网的自动化组件进行交互。它提供了以下通信功能：

- S7 通信
- 开放式通信服务 (SEND/RECEIVE)
- PROFIBUS DP 和 FDL
- SNMP
- PROFINET IO

针对各种要求对通信功能进行了特殊优化。用于数据访问和统一架构的 OPC 服务器可同时支持若干个通信功能。S7 通信、PROFINET IO 通信、SEND/RECEIVE 通信和 DP 通信也可用于 OPC UA。

## 2.2 什么是过程变量？

过程变量是过程 I/O

的可写入和/或可读取数据项，如用作可编程控制器输入值的罐温度。

### OPC COM (数据访问)

在 OPC 数据访问类模型中，过程变量通过 OPC 数据项类表示。只有此类元素才表示 OPC 中过程的实数值。

## 2.3 过程变量的 **ItemID** 是如何构成的？

### **ItemID**

**ItemID** 是唯一标识过程变量的字符串。它会告诉服务器哪些过程变量分配给了 OPC 数据项。然后可通过 OPC 数据项访问过程值。

SIMATIC NET 的 OPC 服务器使用 OPC 数据项映射 SIMATIC NET 通信协议的各种通信服务，方法是将 **ItemID** 的各部分用作调用通信功能的参数。

### OPC UA

在 OPC UA

对象模型中，过程的所有元素均以合适的对象表示。例如，过程变量表示为数据变量或属性，报警表示为报警实例，通信服务表示为方法。各对象之间可相互关联。

### **Nodeld**

**Nodeld** 由命名空间索引和标识符组成（字符串、数字值、字节字符串或 GUID）。

**Nodeld** 可用于标识 OPC UA 服务器上过程中的对象。

SIMATIC NET 的 OPC UA 服务器可通过访问对象来映射协议的各种通信服务。

## 2.3 过程变量的 **ItemID** 是如何构成的？

在 OPC 接口上，过程变量通过唯一名称 **ItemID** 进行标识。**ItemID** 由以下几部分组成：

### 语法

```
<protocolID>: [<connectionname>] <variablename>
```

### 说明

#### **<protocolID>**

指定用于访问过程变量的协议。

存在以下几种协议 ID：

DP	DP 协议，包括 DP 主站、DP 从站和 DPC1
S7	通过 PROFIBUS 和工业以太网实现的 S7 功能
SR	通过工业以太网实现的开放式通信服务
FDL	通过 PROFIBUS 实现的开放式通信服务

SNMP	通过工业以太网实现的 SNMP 通信
PN IO	通过工业以太网实现的 PROFINET IO 通信
DP2	2 类 DP 主站和 DPC2

**<connectionname>**

连接名称标识访问伙伴（例如 PLC、其它 PC 站或 DP 从站）时使用的连接或通信模块。网络组态期间在 STEP 7/SIMATIC NCM PC 中指定此项。

所指定的连接名称取决于正在使用的协议。连接名称在协议中必须唯一。

**说明****连接名称允许使用的字符**

连接名称允许使用字符集中的以下字符：

A-Z、a-z、0-9、\_、-  
、^、!、#、\$、%、&、'、(、)、<、>、=、?、~、+、\*、'、'、:、@、{}、"  
以及空格（连接名称的开头和结尾不能使用空格）

连接名称不允许使用字符集中的以下字符：

- 句点“.”
- 管道符“|”
- 反斜线“\”
- 斜线“/”
- 方括号“[”和“]”

此外，连接名称的开头和结尾不允许使用空格。

**<variablename>**

要寻址的变量。

对于用连接名称指定的连接，变量名称必须唯一。

变量名称的结构取决于所选择的协议。

**说明**

*ItemID* 不区分大小写。

**说明**

有关 OPC UA NodeId 的语法介绍，请参见“对 OPC UA 接口进行编程 (页 586)”部分。

---

### 说明

大括号表示可选语法元素。

---

## 2.4 PROFIBUS DP

### DP 主站的过程变量

DP 主站模式的 SIMATIC NET 的 OPC 服务器提供了用于以下服务的过程变量：

- 1 类主站的服务
  - 访问和监视 DP 输入和输出
- DPC1 服务
  - 非周期性传输数据块
- 快速逻辑
  - CP 5613/CP 5613 A3 和 CP 5614/CP 5614 A3 (仅限 DP 主站)
    - 自动监视从站数据
  - CP 5623 和 CP 5624 (仅限 DP 主站)
    - 自动监视从站数据
- 诊断变量
  - 评估静态诊断信息

### 2 类 DP 主站的过程变量

- 组态服务
  - 读取从站的输入和输出数据
  - 读取从站组态
  - 写入从站的数据记录
- 在线诊断
  - 读取从站的诊断数据
  - 读取 1 类主站诊断数据
- 写入和读取从站的数据记录 (DPC2)

## DP 从站的过程变量

DP 从站模式的 SIMATIC NET 的 OPC 服务器提供了用于以下服务的过程变量：

- 用于访问本地从站数据的变量服务
  - 访问从站的输入和输出
- 诊断变量
  - 评估从站的静态诊断信息

### 2.4.1 1 类 DP 主站

#### 2.4.1.1 适用于 PROFIBUS DP 协议的功能强大的 SIMATIC NET 进程内服务器

PROFIBUS DP 协议包含计算机通信处理器上输入和输出数据的映像。

由于对此过程数据的访问在本地计算机内进行，因此访问速度极快，特别是使用 SIMATIC NET 模块 CP 5613/CP 5614 和 CP 5623/CP 5624 通过“DP Base”接口进行访问时。

在某些情况下，例如使用基于 PC 的控制器时，必须对过程数据进行极其快速的访问。对于速度极快的 DP 协议，SIMATIC NET 提供了进程内服务器，而该服务器实际上也为 OPC 客户端提供了 DP 协议的全部性能。

将 OPC 用作基于 COM 的客户端服务器架构会涉及某些内部执行时间，这取决于 OPC 服务器的执行情况。

由于过程切换而使用本地服务器（也称为“进程外服务器”；具有其自己的进程空间的“\*.exe”文件），并从客户端向服务器传送函数参数（封送）时，这些事件会导致主要问题。

如果 OPC 服务器被作为进程内服务器实施，则可避免用于更改进程和封送的时间，因为 OPC 服务器会采用动态链接库 (DLL) 的形式，并在客户端的进程中运行。

但在使用进程内服务器时，必须考虑以下几点：

- 任何时候都只能有 1 个客户端使用服务器。

如果多个客户端同时使用进程内 OPC

服务器，则将意味着会在不同的进程空间中多次生成服务器，并会导致对相同硬件的同时但不协调的访问。因此，只有首次启动的客户端才有权访问过程数据，其它客户端的访问都会被拒绝。

- OPC 服务器的稳定性取决于客户端。

例如，如果 OPC 客户端以不受控制的方式运行并造成访问违例，则 OPC 服务器也将受到影响。结果是，OPC 服务器将无法根据需要复位通信模块。

此外，也会无法使用组态程序来显式关闭 OPC 服务器。

## 调用 DP 的进程内服务器

使用单独的 ProgID 寻址 DP 的进程内服务器。ProgID 称为“OPC.SimaticNET.DP”。

通过函数调用或在 OPC 客户端中指定 ProgID 来选择服务器。

### 函数调用示例：

#### Visual Basic:

```
ServerObj.Connect ("OPC.SimaticNET.DP")
```

#### Visual C++:

```
r1 = CLSIDFromProgID(L"OPC.SimaticNET.DP", &clsid);
r1 = CoCreateInstance (clsid, NULL, CLSCTX_INPROC_SERVER,
IID_IOPCServer, (void**) &m_pIOPCServer);
```

## 2.4.1.2 适用于 PROFIBUS DP 协议的功能强大的 SIMATIC NET OPC 服务器

### 即使供几个客户端使用也能提高性能

如上所述，用于 PROFIBUS DP 的高性能进程内服务器只能供一个客户端使用。

要允许具有相同性能要求的两个或多个客户端的相同利用，第二个组态变型应可用。

要使用此变型，底层 DP 协议库以及作为进程内服务器的 COM 服务器均应在进程外 OPC 服务器上进行加载。协议在 OPC

服务器的进程中进行处理，这就避免了在进程与多协议模式之间进行切换所需的额外执行时间。不过，OPC 客户端与 OPC 服务器之间的进程切换仍然必不可少。

## 组态

通过在“通信设置”组态程序中选择“DP”协议作为唯一协议，隐式启用此高速变型（如果选择了其它协议或 OPC UA 接口，则会失去下面介绍的性能优点）：



图 2-1 “通信设置”组态程序中用于选择 DP 协议的窗口

也可选择“符号”(Symbols)。

### 说明

使用符号编辑器创建符号时，允许使用以下字符：A-Z、a-z、0-9、\_、-、^、!、#、\$、%、&、'、/、(、)、<、>、=、?、~、+、\*、'、`、|、@、[、]、{、}、"。使用 STEP 7 创建符号时，还应记住，如果符号文件中的符号同时具有 <symbolname> 和 <symbolname>[<索引>] 形式，则解析数组时会出现问题。

## 优点/缺点

使用功能强大 DP OPC 服务器具有以下缺点：只能采用单协议 (DP) 模式。

但另一方面，则具有以下优点：

- 与多协议模式相比，性能更高。
- 组态简单。
- 多个客户端可同时使用服务器。
- OPC 服务器的稳定性不取决于客户端。

### 2.4.1.3 DPC1 和 DPC2 服务

#### 1类 DP 主站

1类 DP 主站与 DP 从站循环通信。通过 DPC1，进行循环操作的主站能够处理附加的非周期性数据通信量（读取数据记录、写入数据记录），并对从站报警做出响应。为了能够使用 DPC1，DP 从站必须支持这些 DP-V1 附加功能。

通信包括如下核心功能：

- 组态参数值并将其分配给从站
- 与 DP 从站进行周期性数据传送
- 监视 DP 从站
- 提供诊断信息

#### 2类 DP 主站

利用 DPC2，2类 DP 主站可与 DP 从站建立更多通信关系。DP 从站必须支持 DPV1 附加功能。最重要的 DPC2 附加功能如下：

- 连接建立
- 终止连接
- 读取数据记录
- 写入数据记录

#### 具有 DP-V1 附加功能的从站

具有 DPV1 附加功能的从站可与 1类和 2类 DP 主站通信。

## DP 协议概述

下图显示了 DP 或 DPV1 协议的组成部分：

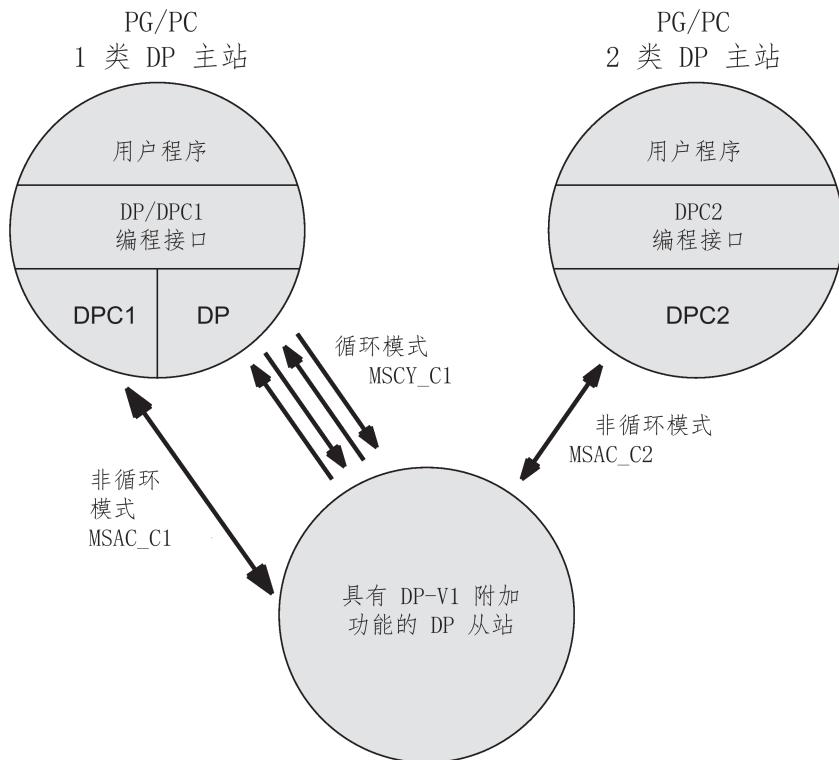


图 2-2 DP 或 DP-V1 协议的组成部分

MSCY_C1	主站-从站, 周期性 C1 模式
MSAC_C1	主站-从站, 非周期性 C1 模式
MSAC_C2	主站-从站, 非周期性 C2 模式

### 2.4.1.4 1 类主站服务的过程变量

利用访问周期性数据的服务，可以访问、监视和控制从站的输入和输出。

通过以下方式进行访问：

- 从站编号。  
此编号与 PROFIBUS 地址相对应。
- 子模块编号。  
DP 从站可包括若干个具有不同输入/输出区域的子模块。
- 输入/输出区域。

### 2.4.1.5 1 类主站过程变量的语法

#### 语法

输入:

```
DP: [<connectionname>] slave<address>{M<number>}_I{<format>
<offset>{.<bit>}{},<quantity>}}
```

输出:

```
DP: [<connectionname>] slave<address>{M<number>}_Q{<format>
<offset>{.<bit>}{},<number>}}
```

#### 说明

##### DP

用于访问过程变量的 DP 协议。

##### <connectionname>

协议特定的连接名称。连接名称在组态中指定。

在 DP 协议中，连接名称是通信模块的已组态名称。

##### slave

表示对 DP 从站进行寻址。

##### <address>

从站的 PROFIBUS 地址。

范围: 0 ... 126。

##### M

包含输入或输出区域的子模块编号的标识符。

##### <number>

包含输入或输出区域的子模块的编号。

##### \_I

输入的标识符。输入为只读输入。

##### \_Q

输出的标识符。可读取和写入输出。

##### <format>

所传送数据的格式。

指定格式即指定数据类型。

原则上可通过 OPC 的自动化和自定义接口读取所有列出的 OLE 数据类型。  
但是，某些开发工具（如 Visual Basic）只提供有限数量的数据类型。

下表列出了可用来表示变量值的相应 Visual Basic 类型。

格式 标识符	说明	OLE 数据类型	Visual Basic 类型
X	位	VT_BOOL	布尔
BYTE 或 B	字节（无符号 8）	VT_UI1	字节
CHAR	字符（有符号 8）	VT_I1	整型
WORD 或 W	字（无符号 16）	VT_UI2	长整型
INT	整型（有符号 16）	VT_I2	整型
DWORD 或 D	双字（无符号 32 位）	VT_UI4	双精度
DINT	双精度整数（无符号 32）	VT_I4	长整型
REAL	浮点数（IEEE 4 字节）	VT_R4	单精度

#### <offset>

要寻址元素所在从站地址空间中的字节偏移。  
如果指定了子模块，则在子模块内应用偏移。  
如果未指定子模块，则偏移适用于从站的整个输入/输出区域。

#### <bit>

已寻址字节中的位编号。  
仅允许使用 X 格式标识符指定位，其范围介于 0 至 7 之间。

#### <quantity>

元素的数量。  
变量的数据类型 (VT\_ARRAY) 是一个具有指定格式的元素的数组。如果省略了 quantity，则数量假定为 1，且变量的数据类型不是数组。

#### 说明

如果未指定子模块，则变量会返回覆盖所有子模块的整个输入或输出区域。  
也可以通过指定格式、偏移和数量进行结构化访问。

### 2.4.1.6 协议 ID

DP 协议的协议 ID 是“DP”。

#### 2.4.1.7 经组态的 CP 名称

连接名称通过指定连接到 PROFIBUS 网络的通信处理器来指定通信访问。

在 DP 协议中，CP 名称是通信模块的已组态名称。

#### 语法

DP: [<configuredCPname>]

#### 说明

<configuredCPname>

用 SIMATIC STEP 7/NCM 指定经组态的 CP 名称。

#### “经组态的 CP 名称”示例

典型的经组态的 CP 名称：

CP 5611 A2

CP 5621

CP 5613

CP 5614

CP 5623

CP 5624

#### 2.4.1.8 1 类主站过程变量的示例

下面几个示例说明了 DP 变量的变量名称的语法。

#### 输入

DP:[CP 5623]Slave005M003\_IB0

Slave005M003\_IB0

从站 5 的子模块 3 的输入字节 0

DP:[CP 5623]Slave005M003\_IB1,3

Slave005M003\_IB1,3

具有 3 个字节的数组，起始于从站 5 的子模块 3 的输入字节 1

*DP:[CP 5623]Slave005M003\_ID2*  
*Slave005M003\_ID2*  
 双字，起始于从站 5 的子模块 3 的输入字节 2

*DP:[CP 5623]Slave004M003\_IReal0*  
*Slave004M003\_IReal0*  
 浮点数，位于从站 4 子模块 3 的输入区域中

*DP:[CP 5623]Slave004\_IB0,8*  
*Slave004\_IB0,8*  
 覆盖所有子模块的从站 4 的整个输入区域的前 8 个字节

## 输出

*DP:[CP 5623]Slave005M007\_QB1*  
*Slave005M007\_QB1*  
 子模块 7 的输出字节 1

*DP:[CP 5623]Slave005M007\_QX2.5*  
*Slave005M007\_QX2.5*  
 从站 5 子模块 7 的输出字节 2 中的位 5

*DP:[CP 5623]Slave004\_QW0,8*  
*Slave004\_QW0,8*  
 具有 8 个字的数组，来自覆盖所有子模块的从站 4 的输出区域

### 2.4.1.9 DPC1 服务

可使用 DPC1 服务来访问 DPC1 从站的数据记录。数据记录非周期性传送。

数据记录的含义由从站的供应商指定。例如，数据记录可用于驱动器的组态数据。

DPC1 变量注册完毕后，OPC 服务器仅能检查语法是否正确，而不会根据 DPC1 从站的组态检查变量在伙伴设备上是否有效以及数据记录大小是否足够。

### 2.4.1.10 DPC1 服务的过程变量的语法

## 语法

```
DP: [<connectionname>] slave<address>S<slot>data<index>
{,<length>} {,<subarea>}
```

## 说明

### DP

用于访问过程变量的 DP 协议。

### <connectionname>

协议特定的连接名称。连接名称在组态中指定。

### slave

用于通过 DP 协议访问从站的标识符。

### <address>

从站的 PROFIBUS 地址。

范围: 0 ... 126。

### S

从站插槽的标识符，通常是子模块。

### <slot>

从站扩展存储区中的插槽，用于非周期性服务。插槽和索引可用于标识数据记录。

### data

用于访问数据记录的标识符。

### <index>

从站扩展存储区中插槽内的索引，用于非周期性服务。

插槽和索引可用于标识数据记录。

### <length>

记录的长度。范围介于 1 至 240 之间。

### <subarea>

子区域由以下几部分组成:

<format><offset>{.<bit>}{,<quantity>}

### <format>

传送数据时所采用的格式。

如果未指定格式，则使用字节格式。

格式 标识符	说明	OLE 数据类型	Visual Basic 类型
X	位	VT_BOOL	布尔
BYTE 或 B	字节 (无符号 8)	VT_UI1	字节
CHAR	字符 (有符号 8)	VT_I1	整型

格式 标识符	说明	OLE 数据类型	Visual Basic 类型
WORD 或 W	字（无符号 16）	VT_UI2	长整型
INT	整型（有符号 16）	VT_I2	整型
DWORD 或 D	双字（无符号 32 位）	VT_UI4	双精度
DINT	双精度整数（无符号 32）	VT_I4	长整型
REAL	浮点数	VT_R4	单精度

**<offset>**

要寻址元素在数据记录中的字节地址。

**<bit>**

已寻址字节中的位编号。

只能使用格式标识符 X 指定位。

**<quantity>**

元素数量

变量的数据类型是一个具有指定格式的元素的数组 (VT\_ARRAY)。如果省略了 **<quantity>**，则假设数量为 1。当数量为 1 时，数据类型是变量而不是数组。

**数据记录的值**

返回数据记录项的值

如果未指定子区域，则数据类型为 VT\_ARRAY | VT\_UI1。

此时的值为只读值；如果指定了数据记录长度，还可以写入值。

**说明**

通过参数 *quantity* 和 *format* 得出的数据长度不得超过从站上数据记录的大小。

数据记录的大小取决于特定的从站，因此无法通过 OPC 服务器对其进行检查。

如果定义了子区域，请记住以下要点：

如果读取了数据记录，则伙伴设备最初将会始终读取整个数据记录。

之后才能对子区域进行评估。

写入时，也会将整个数据记录发送到伙伴设备。

如果使用组合调用写入了若干个子区域，则首先将所有子区域输入到数据记录中，然后再发送数据记录。

因此，应该将所有 OPC

数据项以及对数据记录的部分访问权限放入一个组中，然后写入整个组。

---

### 说明

#### 将数据记录写入伙伴设备

在写入数据记录时，应确保无重叠和间隙，因为无法预测将会写入哪个值。

---

### 2.4.1.11 DPC1 服务的过程变量的示例

下面几个示例说明了 DPC1 服务的变量名称的语法。

#### DPC1 的变量名称

*DP:[CP 5623]Slave005S003Data2,120,DWORD7*

*Slave005S003Data2,120,DWORD7*

访问起始于数据记录中的偏移 7 的双字，数据记录长度为 120 个字节，位于从站 5 的索引 2 的插槽 3 中。

*DP:[CP 5623]Slave005S003Data2,120,B8,4*

*Slave005S003Data2,120,B8,4*

访问具有 4 个字节的数组，该数组起始于数据记录中的偏移 8，数据记录长度为 120 个字节，位于从站 5 的索引 2 的插槽 3 中。

### 2.4.1.12 CP 5613/CP 5614 和 CP 5623/CP 5624 的快速逻辑

CP 5613/CP 5623 以及 CP 5614/CP 5624 的 DP

主站部分支持快速逻辑属性。也就是说可以设置 CP

参数，以便在与某一从站进行数据交换时可以将值写入同一从站或其它从站。进行数据交换时也会通知用户应用程序。

CP 5613/CP 5614 和 CP 5623/CP 5624 提供了 4 个快速逻辑触发器，可使用 OPC 控制变量对其进行配置和评估。

## 快速逻辑的优点

使用快速逻辑具有以下优点：

- OPC 服务器和 OPC 客户端上的负载较小。
- 数据传输速度更快，因为数据传输不依靠在 CP 上运行的软件，而是直接在 CP 的硬件中进行。

---

### 说明

快速逻辑触发器在触发后会再次自动禁用。因此必须使用 FLActivate 变量再次激活该触发器。

只有在 DP 主站处于 OPERATE 模式且相关从站处于 READY 状态时，快速逻辑功能才能正常工作。因此，只有在用户程序将 DP 主站带入 OPERATE 模式后，才能通过 DP 应用程序激活快速逻辑触发器。

只要快速逻辑触发器处于激活状态，DP 用户程序就不能通过快速逻辑写入连接到输入字节的输出字节。

---

### 2.4.1.13 快速逻辑的控制变量的语法

#### 语法

DP: [<connectionname>] FL<parameter><N>

#### 说明

##### DP

用于访问过程变量的 DP 协议。

##### <connectionname>

协议特定的连接名称。连接名称在组态中指定。

##### FL

快速逻辑的标识符。

##### <parameter>

可能值如下：

##### State

返回快速逻辑状态。

返回值：

***CLEARED***

触发器 *N* 未激活。

***ACTIVATED***

触发器 *N* 已激活。

***TRIGGERED***

触发器 *N* 已执行监视。

OLE 数据类型	Visual Basic 类型
VT_BSTR	字符串

**Activate**

只能写入激活。

写入参数字段就是为在 **ItemID** 中指定的触发器指定快速逻辑属性。

所写入的参数字段是一个具有 8 个字节的数组，具有以下结构：

***slave\_addr\_in\_byte***

已针对触发器选择的输入所属从站的地址

***index\_in\_byte***

触发器的输入字节的偏移

***cmp\_value\_in\_byte***

输入字节的比较值

***mask\_in\_byte***

您可以屏蔽输入字节中的单个位，以便在比较时将它们忽略。

通过将相应位中的值置“1”来屏蔽位，也就是说如果设置 **mask\_in\_byte==0x00**，则

**cmp\_value\_in\_byte** 的所有位都用于比较。触发器会在所选输入字节中的所有未屏蔽位与 **cmp\_value\_in\_byte** 中的相应位相匹配时触发。

***slave\_addr\_out\_byte***

选择出现触发条件时将更改的输出字节所属的从站

***index\_out\_byte***

输出字节的偏移量

***value\_out\_byte***

将写入的输出字节的值

***mask\_out\_byte***

可以单独屏蔽输出字节中的位，以便在达到触发条件时不修改这些位。

可通过将相应位中的值置“1”来屏蔽位，这意味着，如果设置了 *mask\_out\_byte==0x00*，则会将 *value\_out\_byte* 中的所有位写入所选输出字节。

OLE 数据类型	Visual Basic 类型
VT_ARRAY of VT_UI1	Byte()

#### Clear

只能写入清除。

如果写入布尔值 TRUE，则会禁用在 ItemID 中指定的快速逻辑触发器。

OLE 数据类型	Visual Basic 类型
VT_BOOL	布尔

#### <N>

正在使用的快速逻辑触发器编号。值介于 1 至 4 之间。

### 2.4.1.14 DP 特定的信息变量

1 类主站的 DP 特定信息变量可用于查询有关 DP 主站及与其相连接的 DP 从站的信息。

可查询以下信息：

- DP 主站的模式
- DP 主站的事件消息
- CP 的生命迹象监视
- DP 从站的模式
- DP 从站的类型
- 用于更详细诊断的其它信息

### 2.4.1.15 DP 特定信息变量的语法

#### 语法

DP: [<connectionname>]<informationparameter>

## 说明

### DP

用于访问过程变量的 DP 协议。

### <连接名称>

协议特定的连接名称。连接名称在组态中指定。

### <信息参数>

可能值如下：

- Masterstate
- EvAutoclear
- EvWatchdog
- EvClass2Master
- WatchdogTimeout
- SlaveNSlvState
- SlaveNSlvDiag
- SlaveNSlvRestart
- SlaveNMiscSlvType
- 其它从站信息

### Masterstate

DP 主站的当前模式。

可写入也可读取当前模式。只有在 DP

应用程序环境的上下文中，才能通过写入如下所示的其中一个值来设置模式。

输入和返回值：

### OFFLINE

主站与从站之间无通信。

### STOP

除诊断数据外，主站和从站之间无通信

### CLEAR

参数分配和组态阶段

### AUTOCLEAR

自动清除阶段，DP 主站无法再访问所有从站

*OPERATE*

生产阶段

OLE 数据类型	Visual Basic 类型
VT_BSTR	字符串

**EvAutoclear**

与 DP 从站通信期间发出错误信号。DP 系统自动关闭并切换到 CLEAR 模式。

返回值：

*True*

与 DP 从站通信时出现错误，系统已关闭并切换到 CLEAR 模式

*False*

未出现错误

OLE 数据类型	Visual Basic 类型
VT_BOOL	布尔

**说明**

如果在组态期间设置了 AUTOCLEAR，则只能使用 EvAutoclear。

如果此信息参数是输出，则不需要任何响应。

**EvWatchdog**

指示模块上的作业监视已超时。OPC 服务器在监视时间内未调用 DP 功能。OPC 客户端或 OPC 服务器可能不再可用。

在组态中设置此监视时间。

返回值：

*True*

模块上的监视已超时

*False*

模块上的监视未超时

OLE 数据类型	Visual Basic 类型
VT_BOOL	布尔

**EvClass2Master**

指示通过 2 类 DP 主站进行访问。

返回值:

*True*

2 类 DP 主站参与数据通信，并访问 1 类 DP 主站的内部诊断列表

*False*

2 类 DP 主站未参与数据通信

OLE 数据类型	Visual Basic 类型
VT_BOOL	布尔

**说明**

不需要用户程序对事件作出响应。

**WatchdogTimeout**

CP 上的看门狗。如果写入了值，则可对看门狗进行设置。在组态中设置默认值。

返回值:

*0*

监视关闭

*400 - 102000*

以 ms 为单位的任意值。该值为 400 的整数倍。

*6000*

默认值

OLE 数据类型	Visual Basic 类型
VT_UI4	双精度

**Slave[nSlvState]**

地址为 *n* 的 DP 从站的当前状态。

返回值:

*OFFLINE*

主站和从站之间无通信

*NOT\_ACTIVE*

DP 从站未激活

*READY*

DP 从站处于数据传送阶段

*READY\_DIAG*

DP 从站处于数据传送阶段；同时存在诊断数据

*NOT\_READY*

DP 从站未处于数据传送阶段

*NOT\_READY\_DIAG*

DP 从站未处于数据传送阶段；同时存在诊断数据

OLE 数据类型	Visual Basic 类型
VT_BSTR	字符串

**Slave/nSlvDiag**

地址为 *n* 的 DP 从站的上次诊断数据。

诊断数据的结构特定于各个从站，因此请参见相应从站的文档。

OLE 数据类型	Visual Basic 类型
VT_ARRAY   VT_UI1	Byte()

**Slave/nSlvRestart**

重新启动 DP 从站的条目，即通过某个条目取消激活再立即重新激活 DP 从站。

在相应已组态的循环时间间隔内调用这两个内部功能。

OLE 数据类型	Visual Basic 类型
VT_BOOL	布尔

返回值：

*True*

重新启动从站 *n*

*False*

无功能

***Slave/nMiscSlvType***

地址为  $n$  的 DP 从站的类型。

返回值：

*NO\_SLV*

无 DP 从站

*NORM*

标准 DP 从站

*ET200\_U*

非标准从站： ET 200 U

*ET200K\_B*

非标准从站： ET 200 K/B

*ET200\_SPM*

非标准从站： 常规 SPM 站

*UNDEFINED*

未知 DP 从站

OLE 数据类型	Visual Basic 类型
<i>VT_BSTR</i>	字符串

**其它从站信息**

下列从站特定的信息参数不用于生产通信。

但如果出现问题，则该参数在评估下文中列出的变量时还是很有用的。

所有变量通常都会直接分配给低级别的 DP 服务。

这些变量的读取方法与其它信息变量相同。 调用结果是特定低级别 DP 服务的字节转储。

***Slave/nMiscReadSlvParCfgData***

地址为  $n$  的从站的组态数据，用于描述输入和输出数据区域以及数据的一致性

***Slave/nMiscReadSlvParUserData***

从站  $n$  的应用程序数据（参数文件的一部分）

***Slave/nMiscReadSlvParPrmData***

从站  $n$  的参数分配数据

***Slave/nMiscReadSlvParType***

从站  $n$  的 SI 标志和类型（与从站相关的标志、从站类型及其它保留数据）

*SlaveMiscSlvDiag*从站 *n* 的诊断数据**2.4.1.16 DP 特定的信息变量的示例**

下面几个示例说明了 DP 特定的信息变量的返回值。

**DP 主站模式***DP:[CP 5623]MasterState***MasterState**

例如，可返回以下值：

*OPERATE*

DP 主站处于生产阶段。

**从站的当前状态***DP:[CP 5623]Slave3SlvState***Slave3SlvState**

例如，可返回以下值：

*READY*

地址为 3 的 DP 从站处于数据传送阶段。

**从站类型***DP:[CP 5623]Slave3MiscSlvType***Slave3MiscSlvType**

例如，可返回以下值：

*ET200\_U*

从站 3 是 ET 200 U。

**2.4.1.17 系统特定信息变量的语法****DP: [SYSTEM] &version()****&version()**

返回 DP OPC 服务器的版本 ID。例如，此处返回字符串

*SIMATIC NET Core Server DP V 7.xxxx.yyyy.zzzz Copyright 2012*

数据类型: VT\_BSTR

访问权限: 只读

## 2.4.2 2类 DP 主站

### 2类 DP 主站

2类 DP 主站可组态和执行在线诊断作业。2类 DP 主站的基本功能:

- 主站诊断: 读取 1类主站诊断数据
- 从站诊断: 读取从站的诊断数据和从站组态
- I/O 数据: 读取从站的输入和输出数据
- 数据记录: 写入和读取从站的数据记录

利用可访问 I/O

数据的非周期性服务, 您可以读取从站的输入和输出, 还可以访问和监视从站的诊断数据。  
。从站必须支持 DPV1 附加功能。您还可以访问 1类主站的数据。

访问方法:

- 通过从站的 PROFIBUS 地址  
从站设备无法启动通信。它们不接收令牌且仅响应来自于主站的轮询。  
因此, 从站也被称为被动节点。此编号与 PROFIBUS 地址相同。
- 通过主站的 PROFIBUS 地址  
从 2类主站访问 1类主站的诊断数据。

#### 2.4.2.1 协议 ID

所有数据项均以前缀“DP2”开头。

#### 2.4.2.2 经组态的 CP 名称

经组态的 CP 名称可指定所使用的通信处理器 (CP)。

### 语法

DP2: [<configuredCPname>]

## 说明

### <configuredCPname>

用 SIMATIC STEP 7/NCM 指定经组态的 CP 名称。

## 示例

*DP2:[CP 5623]*

### 2.4.2.3 主站诊断的过程变量的语法

## 简介

以下部分说明了访问 1 类 DP 主站的诊断数据时所使用的各种语法形式。

## 语法

DP2 : [<configuredCPname>]master<masteraddress>MstDiag

### 总线节点的 DP2 项

#### master

通过 DP 协议访问主站的标识符。

#### <masteraddress>

总线节点的 PROFIBUS 站地址。

范围：0 到 126

## 说明

### 总线节点的 DP2 数据项 - 主站诊断数据项

#### MstDiag

DP 主站的上次系统诊断数据。

数据类型： VT\_ARRAY

访问权限：只读

OLE 数据类型： VT\_BOOL

说明： 126 个元素

数组中的每个元素都指示已分配的从站是否已发送诊断数据；数组索引与 PROFIBUS 地址相对应。

值	含义
FALSE	从站尚未发送诊断数据
TRUE	从站已发送诊断数据

## 语法

```
DP2 : [<configuredCPname>]master<masteraddress>MstState
```

## 说明

### 总线节点的 DP2 数据项 - 主站诊断数据项

#### MstState

DP 主站的状态。包含当前状态和一些版本信息。

数据类型： VT\_ARRAY

访问权限：只读

OLE 数据类型： VT\_UI1

说明： 16 个元素

字节	说明
1	运行状态 0x40 – STOP 0x80 – CLEAR 0xC0 - OPERATE
2 和 3	标识号
4	硬件的版本 (DDLM/用户界面)
5	软件的版本 (DDLM/用户界面)
6	硬件的版本
7	软件的版本
8 到 16	保留

## 语法

```
DP2 : [<configuredCPname>]master<masteraddress>
DataTransferList
```

## 说明

### 总线节点的 DP2 数据项 - 主站诊断数据项

#### **DataTransferList**

DP 主站的数据传送列表。

数据类型: VT\_ARRAY

访问权限: 只读

OLE 数据类型: VT\_BOOL

说明: 126 个元素

数组中的每个元素可表示所分配的从站是否处于生产阶段并已发送数据; 数组索引对应于 PROFIBUS 地址。

值	说明
FALSE	从站不处于数据传送阶段
TRUE	从站处于数据传送阶段

## 语法

```
DP2 : [<configuredCPname>]master<masteraddress>slave
<address>SlvDiag
```

## 说明

### 总线节点的 DP2 数据项 - 主站诊断数据项

#### **<address>**

总线节点的 PROFIBUS 站地址。无需将总线节点连接至 PROFIBUS 便可创建和使用项（可能出现访问错误）。

范围: 0 到 126

---

### 说明

从站服务成功执行后, SIMATIC NET OPC

服务器只能识别已寻址的总线节点实际上是否与 DP 兼容。

由于关键总线节点可能会对类似服务做出异常响应, 因此用户应负责创建相应项。

---

### *SlvDiag*

存储在 DP 主站上的 DP 从站的上次诊断数据。

数据类型: VT\_ARRAY

访问权限: 只读

OLE 数据类型: VT\_UI1

下列各附加数据项针对 *SlvDiag* 字段的各个元素的特定内容和描述而存在:

### 语法

```
DP2:[<configuredCPname>]master<masteraddress>slave
<address>SlvDiagMasterLock
```

如果另一个主站已为该 DP

从站分配了参数, 则此位置位。换句话说, 其所拥有的主站当前无权访问该从站。

*SlvDiag* 字段的第一个字节 (站状态) 的位 7。

数据类型: VT\_BOOL

访问权限: 只读

### 语法

```
DP2:[<configuredCPname>]master<masteraddress>slave
<address>SlvDiagPrmFault
```

如果从站的最后一个参数分配帧错误 (例如, 长度错误、标识号错误或参数无效), 则

DP 从站将对此位进行置位。

*SlvDiag* 字段的第一个字节 (站状态) 的位 6。

数据类型: VT\_BOOL

访问权限: 只读

## 语法

```
DP2:[<configuredCPname>]master<masteraddress>slave
<address>SlvDiagInvalidSlaveResponse
```

如果从已寻址的 DP 从站接收了不合理的响应，则会对此位进行置位。  
*SlvDiag* 字段的第一个字节（站状态）的位 5。

数据类型： VT\_BOOL

访问权限：只读

## 语法

```
DP2:[<configuredCPname>]master<masteraddress>slave
<address>SlvDiagNotSupported
```

如果此从站不支持所请求的功能（例如：请求 SYNC 模式，但此从站不支持），则会对此位进行置位。  
*SlvDiag* 字段的第一个字节（站状态）的位 4。

数据类型： VT\_BOOL

访问权限：只读

## 语法

```
DP2:[<configuredCPname>]master<masteraddress>slave
<address>SlvDiagExtDiag
```

此位由 DP 从站置位。此位置位后，在特定于从站的诊断区域 (Ext\_Diag\_Data) 中必定有一个诊断条目。

如果此位未置位，则在特定于从站的诊断区域 (Ext\_Diag\_Data) 中可能有一条状态消息。必须为特定应用程序定义此状态消息的含义。

*SlvDiag* 字段的第一个字节（站状态）的位 3。

数据类型： VT\_BOOL

访问权限：只读

## 语法

```
DP2:[<configuredCPname>]master<masteraddress>slave
<address>SlvDiagCfgFault
```

如果 DP 主站上次发送的组态数据与 DP

从站检测到的数据不匹配，则会对此位进行置位。这意味着存在组态错误。

*SlvDiag* 字段的第一个站状态字节的位 2。

数据类型： VT\_BOOL

访问权限： 只读

## 语法

```
DP2:[<configuredCPname>]master<masteraddress>slave
<address>SlvDiagStationNotReady
```

如果 DP 从站尚未准备好进行生产数据交换，则会对此位进行置位。

*SlvDiag* 字段的第一个字节（站状态）的位 1。

数据类型： VT\_BOOL

访问权限： 只读

## 语法

```
DP2:[<configuredCPname>]master<masteraddress>slave
<address>SlvDiagStationNonExistent
```

如果 DP 主站无法通过总线连接到 DP

从站，则该主站将对此位进行置位。如果此位置位，则诊断位包含上次诊断消息的状态或初始值。DP 从站将此位永久清零。

*SlvDiag* 字段的第一个字节（站状态）的位 0。

数据类型： VT\_BOOL

访问权限： 只读

## 语法

```
DP2:[<configuredCPname>]master<masteraddress>slave
<address>SlvDiagDeactivated
```

如果 DP

从站在本地参数集中指示为未激活，并且已排除在外，不参与周期性处理，则会对此位进行置位。

*SlvDiag* 字段的第二个字节（站状态）的位 7。

保留 *SlvDiag* 字段的第二个字节（站状态）的位 6。

数据类型: VT\_BOOL

访问权限: 只读

## 语法

```
DP2:[<configuredCPname>]master<masteraddress>slave
<address>SlvDiagSyncMode
```

如果 DP 从站已收到 Sync 控制命令，则 DP 从站将对此位进行置位。

*SlvDiag* 字段的第二个字节（站状态）的位 5。

数据类型: VT\_BOOL

访问权限: 只读

## 语法

```
DP2:[<configuredCPname>]master<masteraddress>slave
<address>SlvDiagFreezeMode
```

如果 DP 从站已收到 Freeze 控制命令，则 DP 从站将对此位进行置位。

*SlvDiag* 字段的第二个字节（站状态）的位 4。

数据类型: VT\_BOOL

访问权限: 只读

## 语法

```
DP2:[<configuredCPname>]master<masteraddress>slave
<address>SlvDiagWDOn
```

此位由 DP 从站置位。如果将此位置 1，则会在 DP 从站上激活看门狗。

*SlvDiag* 字段的第二个字节（站状态）的位 3。

*SlvDiag* 字段的第二个字节（站状态）的位 2 永久置 1。

数据类型: VT\_BOOL

访问权限: 只读

## 语法

```
DP2:[<configuredCPname>]master<masteraddress>slave
<address>SlvDiagStatDiag
```

如果 DP 从站将此位置位，则 DP 主站将提取诊断信息，直至此位再次清零。

例如，如果 DP 从站不能提供任何的有效用户数据，则会将此位置位。

*SlvDiag* 字段的第二个字节（站状态）的位 1。

数据类型： VT\_BOOL

访问权限：只读

## 语法

```
DP2:[<configuredCPname>]master<masteraddress>slave
<address>SlvDiagPrmReq
```

如果 DP 从站将此位置位，则必须为从站重新分配参数和重新组态。

在重新分配参数之前，此位保持置位状态。

*SlvDiag* 字段的第二个字节（站状态）的位 0。

**注意：**

位 1 (*SlvDiagStatDiag*) 和位 0 (*SlvDiagPrmReq*) 同时置位时，位 0 的优先级较高。

数据类型： VT\_BOOL

访问权限：只读

## 语法

```
DP2:[<configuredCPname>]master<masteraddress>slave
<address>SlvDiagExtDiagOverflow
```

此位置位后，可用诊断信息量比 *SlvDiagExtDiagData* 中指出的信息量更大。

如果可用通道诊断信息量比 DP 从站能够输入到发送缓冲区的信息量大，则 DP 从站将会对此位进行置位。或者，如果 DP 从站发送的诊断数据量比 DP 主站在诊断缓冲区中处理的数据量大，则 DP 主站将会对此位进行置位。

*SlvDiag* 字段的第三个字节（站状态）的位 7。

保留 *SlvDiag* 字段的第三个字节（站状态）的位 6 到 0。

数据类型： VT\_BOOL

访问权限：只读

## 语法

```
DP2:[<configuredCPname>]master<masteraddress>slave
<address>SlvDiagMasterAddr
```

将为此 DP 从站分配参数的 DP 主站的地址输入到 *SlvDiag* 字段的字节 4 (*Diag.MasterAdd* 字节)。如果还没有 DP 主站为该 DP 从站分配参数，则该 DP 从站会在此字节中将地址设置为 255。

*SlvDiag* 字段的字节 4 (*Diag.MasterAdd* 字节)。

数据类型： VT\_UI1

访问权限：只读

## 语法

```
DP2:[<configuredCPname>]master<masteraddress>slave
<address>SlvDiagIdentNumber
```

用此字表示 DP 从站类型的制造商 ID。此 ID 可用于测试和精确标识。

*SlvDiag* 字段的第 5 个和第 6 个字节（标识号）。

数据类型： VT\_UI2

访问权限：只读

## 语法

```
DP2:[<configuredCPname>]master<masteraddress>slave
<address>SlvDiagExtDiagData
```

DP 从站可在此字节字段中输入其特定的诊断数据，(*ExtDiagData* 字节)最大长度为 26 个字节。指定具有设备相关诊断数据和 ID 相关诊断数据单独标题字节的块结构。

*SlvDiag* 字段的第 7 个至第 32 个字节 (*ExtDiagData*)。

数据类型： VT\_ARRAY

访问权限：只读

OLE 数据类型： VT\_UI1

### 2.4.2.4 从站诊断的过程变量的语法

## 简介

以下部分说明了访问从站的诊断数据时所使用的各种语法形式。

## 语法

```
DP2:[<configuredCPname>]slave<address>SlvDiag
```

## 说明

### 总线节点的 DPMCL2 数据项 - 从站诊断数据项

#### **SlvDiag**

DP 从站的上次诊断数据。

数据类型: VT\_ARRAY

访问权限: 只读

OLE 数据类型: VT\_UI1

下列各附加数据项针对 *SlvDiag* 字段的各个元素的特定内容和描述而存在:

## 语法

DP2:[<configuredCPname>] slave<address>SlvDiagMasterLock

如果另一个主站已为该 DP

从站分配了参数，则此位置位。换句话说，其所拥有的主站当前无权访问该从站。

*SlvDiag* 字段的第一个字节（站状态）的位 7。

数据类型: VT\_BOOL

访问权限: 只读

## 语法

DP2:[<configuredCPname>] slave<address>SlvDiagPrmFault

如果从站的最后一个参数分配帧错误（例如，长度错误、标识号错误或参数无效），则 DP 从站将对此位进行置位。

*SlvDiag* 字段的第一个字节（站状态）的位 6。

数据类型: VT\_BOOL

访问权限: 只读

## 语法

DP2:[<configuredCPname>] slave<address>SlvDiagInvalidSlaveResponse

如果从已寻址的 DP 从站接收了不合理的响应，则会对此位进行置位。

*SlvDiag* 字段的第一个字节（站状态）的位 5。

数据类型: VT\_BOOL

访问权限：只读

## 语法

DP2:[<configuredCPname>]slave<address>SlvDiagNotSupported

如果此从站不支持所请求的功能（例如：请求 SYNC 模式，但此从站不支持），则会对此位进行置位。

*SlvDiag* 字段的第一个字节（站状态）的位 4。

数据类型： VT\_BOOL

访问权限：只读

## 语法

DP2:[<configuredCPname>]slave<address>SlvDiagExtDiag

此位由 DP 从站置位。此位置位后，在特定于从站的诊断区域 (*Ext\_Diag\_Data*) 中必定有一个诊断条目。

如果此位未置位，则在特定于从站的诊断区域 (*Ext\_Diag\_Data*) 中可能有一条状态消息。必须为特定应用程序定义此状态消息的含义。

*SlvDiag* 字段的第一个字节（站状态）的位 3。

数据类型： VT\_BOOL

访问权限：只读

## 语法

DP2:[<configuredCPname>]slave<address>SlvDiagCfgFault

如果 DP 主站上次发送的组态数据与 DP 从站检测到的数据不匹配，则会对此位进行置位。这意味着存在组态错误。*SlvDiag* 字段的第一个站状态字节的位 2。

数据类型： VT\_BOOL

访问权限：只读

## 语法

DP2:[<configuredCPname>]slave<address>SlvDiagStationNotReady

如果 DP 从站尚未准备好进行生产数据交换，则会对此位进行置位。

*SlvDiag* 字段的第一个字节（站状态）的位 1。

数据类型： VT\_BOOL

访问权限： 只读

## 语法

```
DP2:[<configuredCPname>] slave<address>SlvDiagStationNonExistent
```

如果 DP 主站无法通过总线连接到 DP

从站，则该主站将对此位进行置位。如果此位置位，则诊断位包含上次诊断消息的状态或初始值。DP 从站将此位永久清零。

*SlvDiag* 字段的第一个字节（站状态）的位 0。

数据类型： VT\_BOOL

访问权限： 只读

## 语法

```
DP2:[<configuredCPname>] slave<address>SlvDiagDeactivated
```

如果 DP

从站在本地参数集中指示为未激活，并且已排除在外，不参与周期性处理，则会对此位进行置位。

*SlvDiag* 字段的第二个字节（站状态）的位 7。

保留 *SlvDiag* 字段的第二个字节（站状态）的位 6。

数据类型： VT\_BOOL

访问权限： 只读

## 语法

```
DP2:[<configuredCPname>] slave<address>SlvDiagSyncMode
```

如果 DP 从站已收到 Sync 控制命令，则 DP 从站将对此位进行置位。

*SlvDiag* 字段的第二个字节（站状态）的位 5。

数据类型： VT\_BOOL

访问权限： 只读

## 语法

DP2:[<configuredCPname>]slave<address>SlvDiagFreezeMode

如果 DP 从站已收到 **Freeze** 控制命令，则 DP 从站将对此位进行置位。

*SlvDiag* 字段的第二个字节（站状态）的位 4。

数据类型： VT\_BOOL

访问权限：只读

## 语法

DP2:[<configuredCPname>]slave<address>SlvDiagWDOn

此位由 DP 从站置位。如果将此位置 1，则会在 DP 从站上激活看门狗。

*SlvDiag* 字段的第二个字节（站状态）的位 3。

*SlvDiag* 字段的第二个字节（站状态）的位 2 永久置 1。

数据类型： VT\_BOOL

访问权限：只读

## 语法

DP2:[<configuredCPname>]slave<address>SlvDiagStatDiag

如果 DP 从站将此位置位，则 DP 主站将提取诊断信息，直至此位再次清零。

例如，如果 DP 从站不能提供任何的有效用户数据，则会将此位置位。

*SlvDiag* 字段的第二个字节（站状态）的位 1。

数据类型： VT\_BOOL

访问权限：只读

## 语法

DP2:[<configuredCPname>]slave<address>SlvDiagPrmReq

如果 DP 从站将此位置位，则必须为从站重新分配参数和重新组态。

在重新分配参数之前，此位保持置位状态。

*SlvDiag* 字段的第二个字节（站状态）的位 0。

注意：

位 1 (*SlvDiagStatDiag*) 和位 0 (*SlvDiagPrmReq*) 同时置位时，位 0 的优先级较高。

数据类型： VT\_BOOL

访问权限：只读

## 语法

DP2 : [<configuredCPname>] slave<address>SlvDiagExtDiagOverflow

此位置位后，可用诊断信息量比 *SlvDiagExtDiagData* 中指出的信息量更大。

如果可用通道诊断信息量比 DP 从站能够输入到发送缓冲区的信息量大，则 DP 从站将会对此位进行置位。或者，如果 DP 从站发送的诊断数据量比 DP 主站能够在诊断缓冲区中处理的数据量大，则 DP 主站将会对此位进行置位。

*SlvDiag* 字段的第三个字节（站状态）的位 7。

保留 *SlvDiag* 字段的第三个字节（站状态）的位 6。

数据类型： VT\_BOOL

访问权限：只读

## 语法

DP2 : [<configuredCPname>] slave<address>SlvDiagMasterAddr

将为此 DP 从站分配参数的 DP 主站的地址输入到 *SlvDiag* 字段的字节

4 (*Diag.MasterAdd* 字节)。如果还没有 DP 主站为该 DP 从站分配参数，则该 DP 从站会在此字节中将地址设置为 255。

*SlvDiag* 字段的字节 4 (*Diag.MasterAdd* 字节)。

数据类型： VT\_UI1

访问权限：只读

## 语法

DP2 : [<configuredCPname>] slave<address>SlvDiagIdentNumber

用此字表示 DP 从站类型的制造商 ID。此 ID 可用于测试和精确标识。

*SlvDiag* 字段的第 5 个和第 6 个字节（标识号）。

数据类型： VT\_UI2

访问权限：只读

## 语法

DP2 : [<configuredCPname>] slave<address>SlvDiagExtDiagData

DP 从站可在此字节字段中输入其特定的诊断数据，(*ExtDiagData* 字节) 最大长度为 26 个字节。指定具有设备相关诊断数据和 ID 相关诊断数据单独标题字节的块结构。

*SlvDiag* 字段的第 7 个至第 32 个字节 (*ExtDiagData*)。

数据类型: VT\_ARRAY

访问权限: 只读

OLE 数据类型: VT\_UI1

## 语法

```
DP2:[<configuredCPname>]slave<address>SlvCFGData
```

## 说明

### 总线节点的 DPMCL2 数据项 - 从站诊断数据项

#### SlvCFGData

存储在 DP 主站上的 DP 从站的组态数据。

数据类型: VT\_ARRAY

访问权限: 只读

OLE 数据类型: VT\_UI1

## 语法

```
DP2:[<configuredCPname>]slave<address>SetSlaveAddress
```

## 说明

### 总线节点的 DPMCL2 数据项 - 从站诊断数据项

#### SetSlaveAddress

设置从站的新 PROFIBUS 地址

数据类型: VT\_ARRAY

访问权限: 只写

OLE 数据类型: VT\_VARIANT

写入该数据项时，执行相应的 DPMCL2 服务。

要写入的值还包含执行服务时所需的由参数组成的数组。

数组元素	数据类型	含义
1	VT_UI1	要分配的新从站地址
2	VT_BOOL	指示以后是否能够再次更改 DP 从站地址的标志。
3	VT_I4	节点的设备类型 (PROFIBUS 标识号)
4	VT_ARRAY   VT_UI1	用户特定数据。可传送空数组。

成功写入新从站地址后，可能需要为从站创建新数据项。

包含旧从站地址的数据项（例如 DPMCL2\_I/O 数据项）继续访问旧地址处的从站，因此无法再对其进行成功读取。

#### 2.4.2.5 I/O 数据过程变量的语法

英语缩写 I/O (输入/输出) 与德语缩写 E/A (Eingang/Ausgang) 相对应。

##### 语法

访问从站 I/O 区域中的输入时，从站变量必须采用以下语法：

```
DP2:[<configuredCPname>] slave<address>_E{<format>
<offset{.bit}>{,quantity}}
```

或 (英语) :

```
DP2:[<configuredCPname>] slave<address>_I{<format>
<offset{.bit}>{,quantity}}
```

##### 对总线节点

##### DPMCL2 项的说明 - 从站 I 项

<address>\_E 或英语 <address>\_I  
 <address>\_A 或英语 <address>\_Q

只能用 SIMATIC NET OPC 服务器的 OPC 客户端读取输入和输出。

2 类 DP 主站将忽略在已定义组态和一致性分配的子模块和数据尺寸 (字节或字)。

这样做会将其与 1 类 DP 主站区分开。尽管 2 类 DP

主站可以在运行期间查询从站的组态，但从站和相应的 1 类 DP

主站仍可以在运行期间变更其组态，而无需通知 2 类 DP 主站。

这样做会造成组态数据不一致，即使执行耗费时间的组态数据轮询也是如此（记住网络上有相当数量的负荷）。

通过指定从站数量 **<address>** 和 I/O 区域访问 I/O 区域，其中 **\_I** (或 **\_E**, 德语) 用于标识输入区域。

如果忽略可选格式信息，数据将采用字节数组的形式随输入的整个长度一同返回。

数据类型: **VT\_ARRAY**

访问权限: 只读

OLE 数据类型: **VT\_UI1**

#### **<format>**

格式元素用于指定发送数据时将使用的格式。如果未指定格式，则使用字节格式。指定格式也会指定数据类型。

格式标识符	说明	OLE 数据类型	Visual Basic 数据类型
X	位	VT_BOOL	布尔
BYTE 或 B	字节 (无符号 8)	VT_UI1	字节
CHAR	字符 (有符号 8)	VT_I1	Integer
WORD 或 W	字 (无符号 16)	VT_UI2	Long
INT	整型 (有符号 16)	VT_I2	Integer
DWORD 或 D	双字 (无符号 32 位)	VT_UI4	双精度
DINT	双精度整数 (无符号 32)	VT_I4	Long
REAL	浮点数	VT_R4	Single

#### **<offset{.bit}>**

要访问的元素以字节计的偏移量 (偏移量寻址)。只能用类型 X 指定位。

以字节为单位指定偏移量。

示例: **X2.3** 返回第 2 个字节的第 3 位。

#### **<quantity>**

元素的数量。变量的数据类型是带有指定格式元素的数组 (数据类型 **VT\_ARRAY**)。无法指定格式 X 的数量。

如果忽略此部分名称或指定数字 1，将假定数量为 1，变量的数据类型不是数组。

## 语法

访问从站 I/O 区域中的输出时，从站变量必须采用以下语法：

```
DP2:[<configuredCPname>] slave<address>_A{<format>
<offset{.bit}>{,quantity}}
```

或（英语）：

```
DP2:[<configuredCPname>] slave<address>_Q{<format>
<offset{.bit}>{,quantity}}
```

## 对总线节点

### DPMCL2 项的说明 - 从站 Q 项

通过指定从站数量 <address> 和 I/O 区域访问 I/O 区域，其中 \_IQ (或 \_A, 德语) 用于标识输出区域。

如果忽略可选格式信息，数据将采用字节数组的形式随输出的整个长度一同返回。

数据类型： VT\_ARRAY

访问权限： 只读

OLE 数据类型： VT\_UI1

采用与从站的 I 项相同的格式。

### 2.4.2.6 数据记录过程变量的语法

## 语法

```
DP2:[<configuredCPname>] slave<address>S<slot>data<index>
{,<length>} {,<subarea>}
where subarea = <subarea> = <format><offset>{.<bit>} {,<quantity>}
```

## 总线节点的 DP2 项

### slave

用于通过 DP 协议访问从站的标识符。

### address

总线节点的 PROFIBUS 站地址。无需将总线节点连接至 PROFIBUS 便可创建和使用项（可能出现访问错误）。

范围： 0 到 126

---

### 说明

从站服务成功执行后，SIMATIC NET OPC 服务器只能识别已寻址的总线节点实际上是否与 DP 兼容。

由于关键总线节点可能会对类似服务做出异常响应，因此用户应负责创建相应项。

---

## 对总线节点

### DPC2 项的说明 - DPC2 读/写

#### **S**

从站插槽的标识符，通常是子模块。

#### **<slot>**

从站扩展存储区中的插槽。插槽和索引可用于标识数据记录。

范围： 0 到 255（在 C 接口上： 0 到 255）

#### **data**

数据记录标识符。

#### **<index>**

从站插槽中的数据索引。

范围： 0 到 255

#### **<length>**

记录的长度。

范围： 1 到 240

#### **<subarea>**

子区域的标识符。

#### **<format>**

格式元素用于指定发送数据时将使用的格式。指定格式也会指定数据类型。

格式标识符	说明	OLE 数据类型	Visual Basic 数据类型
X	位	VT_BOOL	布尔
BYTE 或 B	字节（无符号 8）	VT_UI1	字节
CHAR	字符（有符号 8）	VT_I1	Integer
WORD 或 W	字（无符号 16）	VT_UI2	Long
INT	整型（有符号 16）	VT_I2	Integer

格式标识符	说明	OLE 数据类型	Visual Basic 数据类型
DWORD 或 D	双字（无符号 32 位）	VT_UI4	双精度
DINT	双精度整数（无符号 32）	VT_I4	Long
REAL	浮点数	VT_R4	Single

**<offset{.bit}>**

要被访问的元素以字节计的偏移量。只能用类型 X 指定位。

示例：X2.3 返回第 2 个字节的第 3 位。

**<quantity>**

元素的数量（不允许采用 X 格式）。

变量的数据类型是带有指定格式元素的数组（数据类型 VT\_ARRAY）。

如果忽略此部分名称或指定数字 1，则假定数字为 1，变量的数据类型不是数组。

**数据记录的值**

返回数据记录项的值

如果未指定子区域，则数据类型为 VT\_ARRAY | VT\_UI1。

只读；如果指定了数据记录长度，则也可以将其写入。

**说明**

通过参数 quantity 和 format 得出的数据长度不得超过从站上数据记录的大小。

数据记录的大小取决于特定的从站，因此无法通过 OPC 服务器对其进行检查。

读取数据记录的子区域时，伙伴设备首先读取整个数据记录，然后评估相关的子区域。

写入子区域时，整个数据记录也将被传送到伙伴设备。

如果在写入作业中写入数据记录的几个子区域，则在更新过数据记录的所有子区域后，将数据记录只写入伙伴设备。

**使用 OPC**

服务器时，将对特定的数据记录具有部分访问权限的所有项一同放入一个组以及对整个组进行写操作是很有意义的事情。

**说明****将数据记录写入伙伴设备**

在写入数据记录时，应确保无重叠和间隙，因为无法预测将会写入哪个值。

---

### 说明

无法使用相同项或具有相同地址、插槽和索引的项通过相同的方式读取 OPC DPC2 数据记录项的写入值。DPC2

数据记录的写入值和读取值可能不同，这取决于具体的设备。

---

### 语法

```
DP2:[<configuredCPname>] slave<address>DTS<slot>data
<index>{,<length>} {,<subarea>}
where <subarea> = <format><offset>{.<bit>} {,<quantity>}
```

### 对总线节点

#### DPC2 项的说明 - DPC2 数据传送

指定长度后，DPC2 数据传送项的访问权限为读/写（否则为只读）。

如果未指定子区域，则采用以下数据类型：

数据类型： VT\_ARRAY

OLE 数据类型： VT\_UI1

此项在协议中执行数据交换。

这些项的读访问适用于本地，可用于返回通过最后的写访问获得的数据记录。

#### DTS

数据传送的标识符。

#### <slot>

从站扩展存储区中的插槽。插槽和索引用于标识数据记录。

范围：0 到 255

#### <offset>{.<bit>}

要被访问的元素以字节计的偏移量。只能用类型 X 指定位。

示例：X2.3 返回第 2 个字节的第 3 位。

**<quantity>**

元素的数量（不允许采用 X 格式）。变量的数据类型是具有指定格式元素（数据类型：VT\_ARRAY）的数组。如果忽略此部分名称或指定数字 1，则假定数字为 1，变量的数据类型不是数组。

**说明**

读取数据记录的子区域时，评估从伙伴设备收到的最后数据记录的子区域。

写入子区域时，整个数据记录也将被传送到伙伴设备。

如果在写入作业中写入数据记录的几个子区域，则在更新过数据记录的所有子区域后，将数据记录只写入伙伴设备。

**使用 OPC**

服务器时，将对特定的数据记录具有部分访问权限的所有项一同放入一个组以及对整个组进行写操作是很有意义的事情。

**说明****将数据记录写入伙伴设备**

在写入数据记录时，应确保无重叠和间隙，因为无法预测将会写入哪个值。

**2.4.2.7 数据记录过程变量示例**

下面几个示例说明了 DPC2 数据记录的变量名称的语法。

**DPC2 的变量名称**

*DP2:[CP 5623]Slave005S003Data2,120,DWORD7*

访问起始于数据记录中的偏移 7 的双字，数据记录长度为 120 个字节，位于从站 5 的索引 2 的插槽 3 中。

*DP2:[CP 5623]Slave005S003Data2,120,B8,4*

访问具有 4 个字节的数组，该数组起始于数据记录中的偏移 8，数据记录长度为 120 个字节，位于从站 5 的索引 2 的插槽 3 中。

*DP2:[CP 5623]Slave005DTS006Data2,10,DWORD2*

数据传送访问起始于数据记录中的偏移 2 的双字，数据记录长度为 10 个字节，位于从站 5 的索引 2 的插槽 6 中。

### 2.4.2.8 DP2 特定信息变量的语法

#### 语法

DP2 : [<configuredCPname>] &identify()

#### 说明

##### **&identify()**

将指定设备的站标识符以包含 4 个字符串的数组的形式返回：

数据类型： VT\_ARRAY

说明： 4 个数组元素 (0-3)

访问权限： 只读

OLE 数据类型： VT\_BSTR

返回值的元素：

- 供应商
- 控制器
- 硬件版本
- 软件版本

示例： {Siemens AG|FW-CP 5613A2 EL (E2)|1.0|V 6.2.1.3175 20.08.2004}

**OLE 数据类型 Visual Basic 类型**

VT\_ARRAY | VT\_BSTR String()

#### 语法

DP2 : [<configuredCPname>] lifelist()

#### 说明

##### **lifelist()**

常规项

返回一个包含所有已连接总线节点的列表。

数据类型： VT\_ARRAY

说明： 127 个数组元素 (0-126)

访问权限：只读

OLE 数据类型： VT\_UI1

每个数组元素代表一个 PROFIBUS 站地址。 数组元素的值有以下含义：

元素	含义
0x00	STATION_PASSIVE
0x10	STATION_NON_EXISTENT
0x20	STATION_ACTIVE_READY (准备进入逻辑环)
0x30	STATION_ACTIVE (已在逻辑环中)

#### 说明

将 lifelist 用作 OPC 项（例如 DP2:[CP5614A2]lifelist()）时，请记住以下内容：

“Lifelist”FDL 服务只能通过一个应用程序调用。如果将其设置为 OPC 项，则其它应用程序将不再使用该服务。特别是 lifelist 将不再在“通信设置”组态程序中显示。

由于 lifelist 也会对总线造成一定量的负载，而这种负载与 OPC 一同循环发生，因此应该尽量避免发生这一情况。

#### 2.4.2.9 系统特定信息变量的语法

DP2:[SYSTEM]&version()

##### &version()

返回 DP2 OPC 服务器的版本 ID。例如，此处返回字符串

*SIMATIC NET Core Server DP2 V 7.xxxx.yyyy.zzzz Copyright 2012*

数据类型： VT\_BSTR

访问权限：只读

## 2.4.3 DP 从站

---

### 说明

将 PROFIBUS 通信处理器作为 DP 从站时，只有将 OPC 作为 DP-V0 从站才能运行。由于指定了 HARDNET 模块，因此不能通过 OPC 读取或写入任何数据记录。

---

### 2.4.3.1 用于访问本地从站数据的变量服务

SIMATIC NET 的 OPC 服务器在 PROFIBUS DP 网络中不但可作为 DP 主站运行，还可扮演 DP 从站的角色。OPC 服务器管理该 DP 从站的输入和输出存储区，并将它们映射到 OPC 变量。一个 OPC 客户端可以读取主站设置的输出并在下一个循环中会被主站获取的输入中设置值。

DP 从站具有模块化结构。一个 DP

从站可以包括几个带有不同输入/输出区域的子模块。在组态期间分配子模块。

变量名称用于标识从站子模块中的输入或输出区域。

通过指定子模块编号和输入或输出区域访问从站的输入和输出。

也可以通过变量名称查询从站和 DP 主站的状态信息。

---

### 说明

只有在组态期间设置了“DP 基站”模式，带有 CP 5614、CP 5614 A2 或 CP 5614 FODP 模块的 DP 主站和 DP 从站的并行操作才有可能实现。您也可以选择使用 CP 5624。

---

### 2.4.3.2 DP 从站过程变量的语法

#### 语法

输入：

```
DP:[<connectionname>] slave{M<number>}_I{<format><offset>
{.<bit>} {,<quantity>}}
```

输出：

```
DP:[<connectionname>] slave{M<number>}_Q{<format><offset>
{.<bit>} {,<quantity>}}
```

## 说明

### DP

用于访问过程变量的协议。

### <connectionname>

在组态中指定的通信模块的名称。

### slave

用于通过 DP 协议访问从站的标识符。

### M

子模块编号的标识符。

### <number>

包含输入或输出区域的子模块的编号。

### I

输入的标识符。

### \_Q

输出的标识符。

### <format>

传送数据时所采用的格式。

指定格式也会指定数据类型。

所有指定的 OLE 数据类型都可通过 OPC 的自动化接口进行读取。

但是，某些开发工具（例如 Visual Basic）只提供有限数量的数据类型。

因此，下表列出了可用于表示变量值的相应 Visual Basic 类型。

格式标识符	说明	OLE 数据类型	Visual Basic 类型
X	位	VT_BOOL	布尔
BYTE 或 B	字节（无符号 8）	VT_UI1	字节
CHAR	字符（有符号 8）	VT_I1	Integer
WORD 或 W	字（无符号 16）	VT_UI2	Long
INT	整型（有符号 16）	VT_I2	Integer
DWORD 或 D	双字（无符号 32 位）	VT_UI4	Double
DINT	双精度整数（无符号 32）	VT_I4	Long
REAL	浮点数	VT_R4	Single

**<offset>**

要被寻址的元素所在从站的地址空间中的字节地址。

**<bit>**

已寻址字节中的位编号。范围在 0 和 7 之间。

一个仅能用格式标识符 *X* 指定的位。

**<quantity>**

元素的数量。

变量的数据类型 (VT\_ARRAY) 是一个具有指定格式的元素的数组。

如果忽略数量，则假定数量为 1，并且变量的数据类型不是数组。

请勿使用带 *X* 格式标识符的数量参数。

### 2.4.3.3 DP 从站过程变量示例

下面几个示例说明了 DP 从站变量的变量名称的语法。

#### 输入

*DP:[CP 5611]SlaveM003\_IB0*

SlaveM003\_IB0

DP 从站子模块 3 的输入字节 0 (偏移 0)。

*DP:[CP 5611]SlaveM003\_IB1,3*

SlaveM003\_IB1,3

一个包含 3 个字节的数组，从 DP 从站子模块 3 的输入字节 1 (偏移 1) 处开始。

*DP:[CP 5624]SlaveM003\_IX0.0*

SlaveM003\_IX0.0

DP 从站子模块 3 的字节 0 中的输入位 0

*DP:[CP 5614]SlaveM003\_IB3,8*

SlaveM003\_IB3,8

一个包含 8 个输入字节的数组，从 DP 从站子模块 3 的偏移 3 开始。

#### 输出

*DP:[CP 5611]SlaveM003\_QW3*

SlaveM004\_QW3

位于 DP 从站子模块 4 中地址 3 的输出字。

*DP:[CP 5611]SlaveM003\_QDWORD2*

*SlaveM003\_QDWORD2*

位于 DP 从站子模块 3 中地址 2 的输出双字。

*DP:[CP 5624]SlaveM003\_QX3.7*

*Slave\_QX3.7*

DP 从站字节 3 中的输出位 7。

*DP:[CP 5624]SlaveM001\_QW0,4*

*SlaveM001\_QW0,4*

DP 从站子模块 1 中包含 4 个输出字的数组。

#### 2.4.3.4 DP 从站特定信息变量

对于 DP 从站诊断，有几个预定义的信息变量。

#### 2.4.3.5 DP 从站特定信息变量的语法

##### 语法

有两个选项：

*DP: [<connectionname>]<diagnosticitem>*

*DP: [<connectionname>]<parameteritem>*

##### 说明

###### DP

用于访问过程变量的协议。

###### <connectionname>

在组态中指定的通信模块的名称。

###### <diagnosticitem>

预定义项。

有以下选项：

###### devicestate

DP 从站所处模块的状态。

可能具有以下状态:

- ONLINE
- OFFLINE

#### <parameteritem>

预定义参数项。

可用选项如下:

- *SlaveMiscReadSlvParCfgData*  
从站的组态数据
- *SlaveSlvState*  
从站的状态。

可能具有以下状态:

- DATA\_EXCHANGE
- NO\_DATA\_EXCHANGE

## 2.5 带 OPC UA 的 PROFIBUS DP

### 带 OPC UA 的 DP 主站的过程变量

通过 OPC UA 的 DP 主站模式的 SIMATIC NET 的 OPC 服务器提供针对以下服务的过程变量:

- 1 类主站的服务  
访问和监视 DP 输入和输出
- 同步/冻结  
向从站组非周期性发送控制帧
- 快速逻辑
  - CP 5613 A2 和 CP 5614 A2 (仅限 DP 主站)  
自动监视从站数据
  - CP 5623 和 CP 5624 (仅限 DP 主站)  
自动监视从站数据
- 诊断变量  
评估静态诊断信息

## DP 从站的过程变量

通过 OPC UA 的 DP 从站模式的 SIMATIC NET 的 OPC 服务器提供针对以下服务的过程变量：

- 用于访问本地从站数据的变量服务
  - 访问从站的输入和输出
- 诊断变量
  - 评估从站的静态诊断信息

### 2.5.1 针对 DP 协议的 SIMATIC NET OPC UA 服务器

#### 简介

以下部分描述了一个 DP 协议的组态变量，此变量在支持 OPC UA 的同时也支持 DP COM OPC 数据访问服务器。要做到这一点，需要将 DP COM OPC 访问服务器设置为进程外 OPC 服务器。

由于 PROFIBUS DP 协议在 PC 中通信处理器的 DP RAM 中保持了一个输入和输出数据的映像，因此对过程数据的访问将仅在 PC 内部本地发生。特别是在 DP Base 接口上使用 SIMATIC NET 模块 CP 5613 A2、CP 5613 A3、CP 5614 A2、CP 5614 A3、CP 5623 和 CP 5624 时，这样做会使访问变得极其快速。

在某些情况下，例如在使用基于 PC 的控制器时，必须在极短时间内访问过程数据。

## 组态

通过在“通信设置”组态程序的“OPC 协议选择”(OPC protocol selection) 中选择“DP”和“OPC UA”激活 DP OPC UA 服务器:



图 2-3 “通信设置”组态程序中用于选择 DP 协议的 OPC UA 的窗口

## 优点/缺点

使用 DP OPC UA 服务器时，只能使用 DP OPC 服务器的进程外模式。必须启动 DP OPC UA 服务器进程才能维护 UA 准备接收的状态。即使所有 OPC UA 客户端都已注销，也不会退出 DP OPC UA 服务器。如果停止 DP OPC UA 服务器进程，则 UA 功能将不再可用。

与 DP COM 服务器相比，有如下优点：

- 不再需要 COM/DCOM 组态。
- 高速、安全的通信

## 2.5.2 支持带 OPC UA 的 DP 服务

### 1 类 DP 主站

DP OPC UA 服务器支持 1 类 DP 主站。1 类 DP 主站用于处理与 DP 从站的周期性通信。通信包括如下核心功能：

- 组态参数值并将其分配给从站
- 与 DP 从站进行周期性数据传送
- 监视 DP 从站
- 提供诊断信息

通过 DPC1，一个循环运行的主站也可以处理非周期性数据通信量。

### OPC UA 中 1 类 DP 主站概述

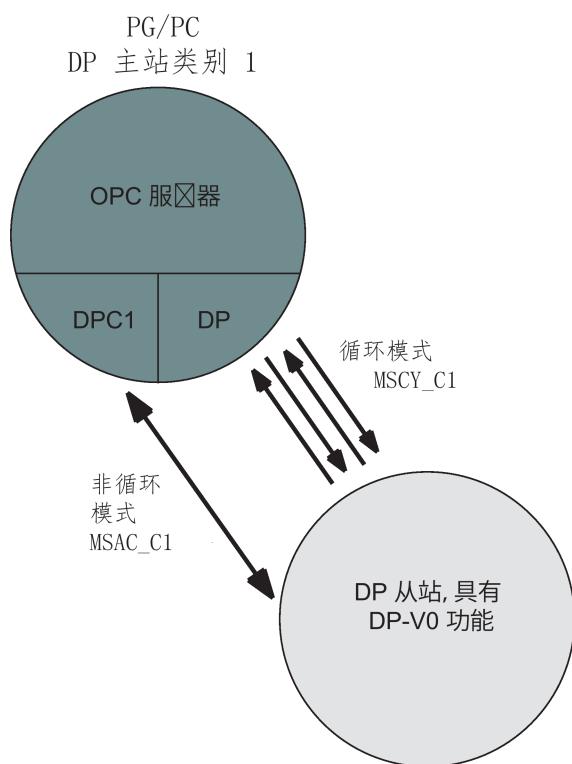


图 2-4 1 类 DP 主站的协议部分

- MSCY\_C1 主站-从站，周期性 C1 模式  
 MSAC\_C1 主站-从站，非周期性 C1 模式

### DP 从站 (DP-V0)

DP OPC UA 服务器支持 DP 从站功能 DP-V0。通信包括如下核心功能：

- 1 类 DP 主站的参数分配和组态
- 对 1 类 DP 主站的周期性数据传送
- 1 类 DP 主站的监视
- 提供 1 类 DP 主站的诊断信息

通过 DPC1，一个循环运行的主站也可以处理非周期性数据通信量。

### DP OPC UA 从站 (V0) 概述

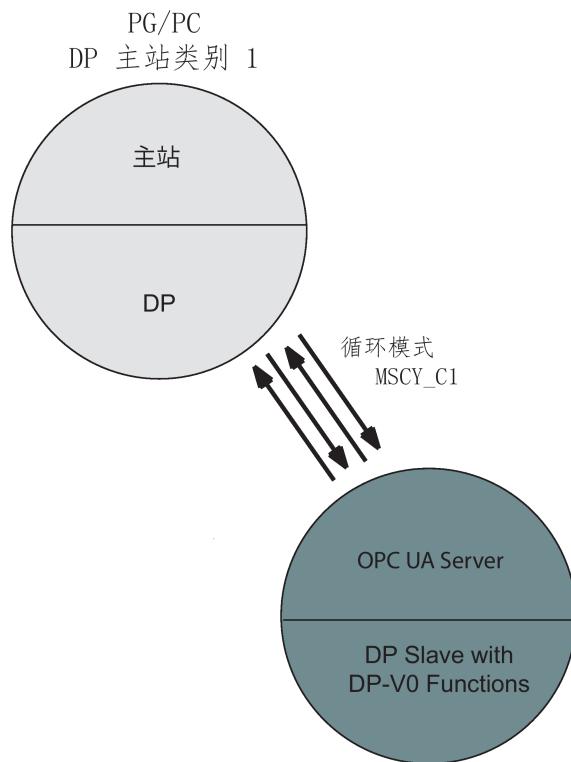


图 2-5 DP 从站 (V0) 的协议部分

MSCY\_C1 主站-从站，周期性 C1 模式

MSAC\_C1 主站-从站，非周期性 C1 模式

### 2.5.3 如何对 DP OPC UA 服务器进行寻址？

#### 服务器 URL

对于本地二进制 TCP 协议，OPC 客户端有两种对服务器进行寻址的方法：

- 直接寻址：

– opc.tcp://<hostname>:55103

或

– opc.tcp://<IP-Adresse>:55103

或

– opc.tcp://localhost:55103

DP OPC UA 服务器使用端口 55103。

- “发现”服务是一个工具，用于列出所有可用的服务器 URL。

出于此目的，请使用以下服务器 URL：

– opc.tcp://<hostname>:4840

或

– opc.tcp://<IP-Adresse>:4840

或

– http://<hostname>:52601/UADiscovery/

或

– http://<IP-Adresse>:52601/UADiscovery/

发现服务器使用端口 4840（用于 TCP 连接）和端口 52601（用于 HTTP 连接）。

#### IPv6 地址

也可通过 IPv6 寻址实现 OPC 客户端应用程序和 OPC 服务器之间的数据交换。

地址必须位于括号中，例如 [fe80:e499:b710:5975:73d8:14]

## 端点和安全模式

SIMATIC NET DP OPC UA 服务器支持通过 TCP

协议实现的通信，该协议由加密和签名确保安全。

应该在生产操作中首选这些安全模式。

已寻址主机上的“发现”服务将用信号通知服务器的端点，换句话说就是服务器的安全要求和协议支持。

DP OPC UA 服务器的服务器 URL "opc.tcp://<hostname>:55103" 提供以下端点：

- 在“签名和加密”安全模式下的端点：

必须进行签名和加密才能与服务器进行通信。通过交换证书和输入密码来保护通信。

除了安全模式，还显示了安全策略 Basic128Rsa15。

- 在“无”安全模式下的端点：

在此模式下，服务器不需要使用安全功能（安全策略“无”），并且这一情况适用于测试和调试。

有关安全功能的详细信息，请参见“对 OPC UA 接口进行编程 (页 586)”部分。

安全策略“Basic128Rsa18”和“无”在 OPC 基金会的 UA 规范中，其 Internet 地址如下所示：

[> "Specifications" > "Part 7"](http://opcfoundation.org/UA)

有关更多的详细信息，请参见以下 Internet 页面：

OPC 基金会 ([> "Security Category" > "Facets" > "Security Policy"](http://www.opcfoundation.org/profilereporting/index.htm))

## OPC Scout V10 的 OPC UA 发现

OPC Scout V10 允许您使用“发现”服务进入 OPC Scout V10 导航区域中的 UA 端点（服务器 URL）。

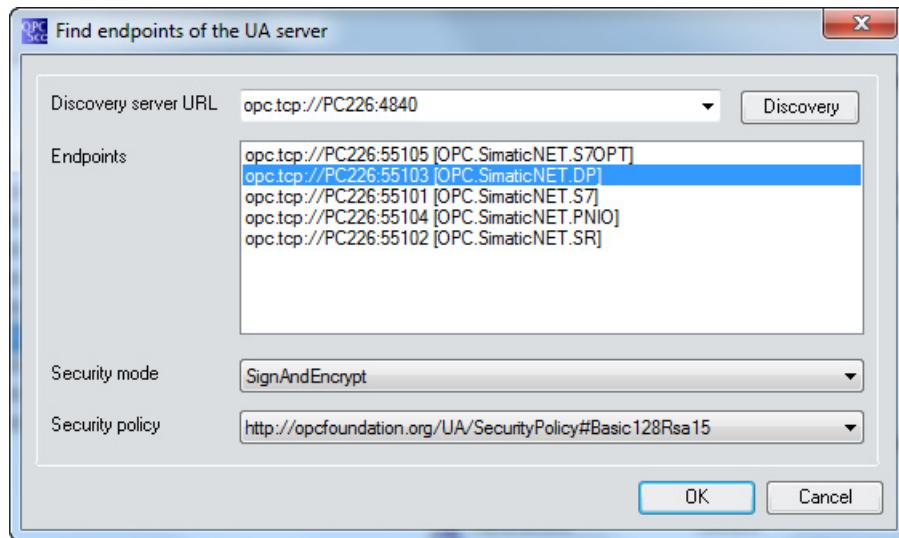


图 2-6 OPC Scout V10 的“查找 UA 服务器端点”对话框

可以使用 OPC UA 发现服务列出所有可用的 OPC UA 服务器（包括 DP OPC UA 服务器）。

OPC Scout V10 可识别 SIMATIC NET 支持的所有 OPC UA 端点。

已寻址主机上的“发现”服务随即针对这些端点及其端口和安全模式用信号通知已注册的 DP OPC UA 服务器。

更多详细信息，请参见 OPC Scout V10 的在线帮助。

## 2.5.4 DP OPC UA 服务器提供了哪些命名空间？

DP OPC UA 服务器提供了以下命名空间：

表格 2-1 DP OPC UA 的命名空间：

命名空间索引	“标识符”（命名空间 URI）/注释
0	“ <a href="http://opcfoundation.org/UA/">http://opcfoundation.org/UA/</a> ” (由 OPC 基金会指定)
1	"urn:Siemens.Automation.SimaticNET.DP:(GUID)" DP OPC UA 服务器的唯一标识符。
2	"DPTYPES:" DP 特定对象类型的定义。
3	"DP:" DP OPC UA 服务器的标识符，使用新简化的语法（可浏览且可与 OPC UA 一起使用）
4	"DPCOM:" 服务器的标识符，使用旧语法，与 DP OPC DA 兼容（可与 OPC UA 一起使用，但不能浏览）

命名空间索引 0 和 1 保留，其意义由 OPC 基金会指定。

将剩余命名空间索引分配至标识符（命名空间  
URI），这一操作必须由指定该标识符的客户端通过 OPC UA  
会话开始处的数据变量“NamespaceArray”来完成。标识符“DPTYPES:”、“DP:”、  
和“DPCOM”始终与 DP OPC UA 服务器一起存在。

## 2.5 带 OPC UA 的 PROFIBUS DP

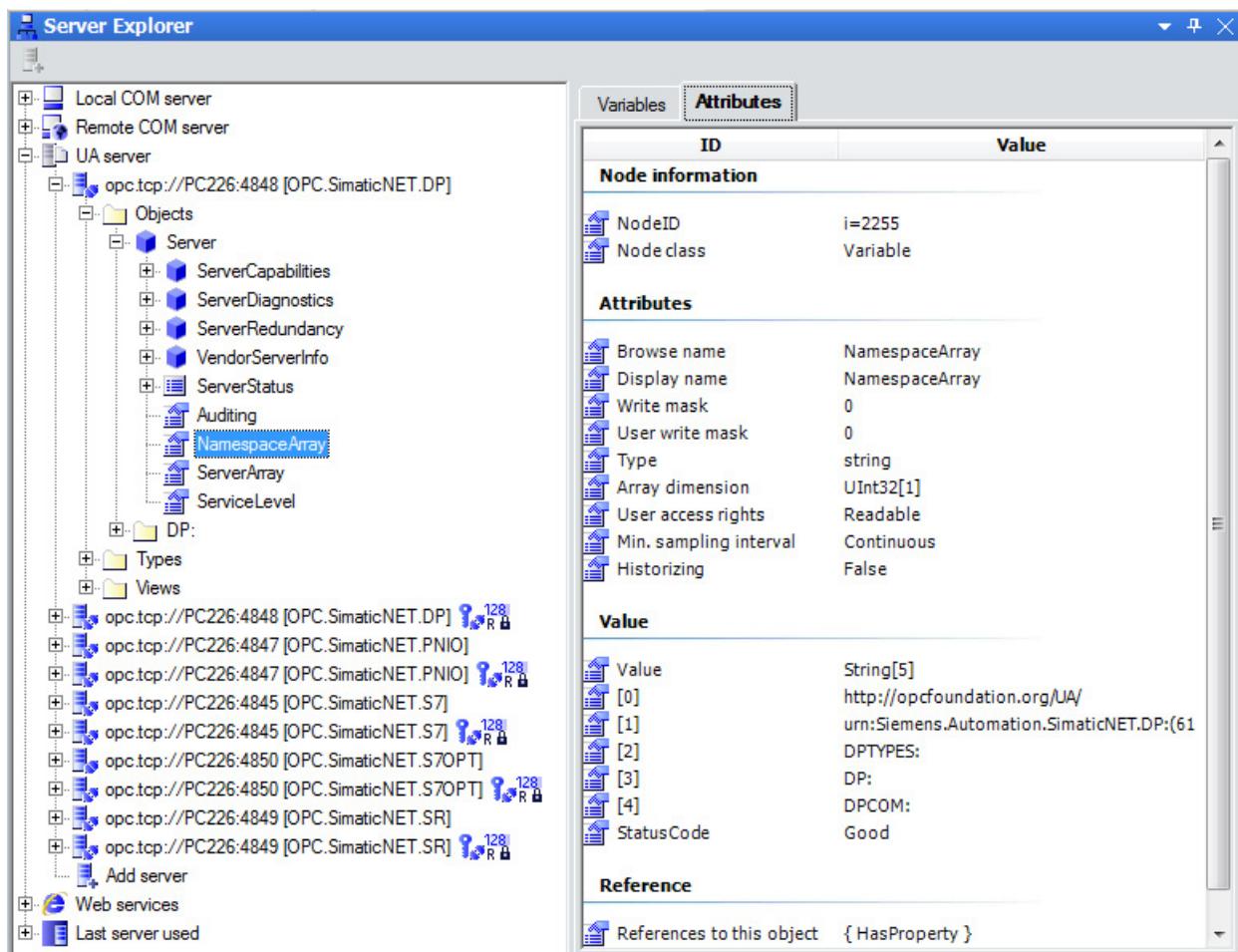


图 2-7 使用 OPC Scout V10 的浏览功能显示 DP OPC UA 的命名空间

### 2.5.5 Nodeld

#### DP 过程变量的标识

Nodeld 在运行期间借助下列元组对 DP 过程变量进行唯一标识：

- 命名空间索引
- 标识符（字符串、数字值）

## 示例

- NodId:
  - 命名空间 URI:  
 $DP:$   
 (= 命名空间索引 3) , 适用于 Siemens.Automation.SimaticNET.DP
  - 标识符:  
 $DPMasterName.slv17.q0.244$
  
- NodId:
  - 命名空间 URI:  
 $DPCOM:$   
 (= 命名空间索引 4) , 适用于 OPC.SimaticNET; 语法与 DP OPC DA 兼容
  - 标识符:  
 $DPMasterName.slv17.q0.244$

## 新的命名空间是如何适应 OPC UA 运作的?

用于读写过程变量的 COM 服务器的“OPC 数据访问”项是一个独立的领域。  
报警领域与其并存但又相互独立。

另一方面，自动化对象的 OPC UA 视图也与对象的各种属性相关。OPC UA 不再单独访问各项，但还会访问对象及其子对象。

- 例如，从站的变量和方法是 DP 主站对象的子对象。  
特性和属性可更详细地定义对象。
- 面向从站访问的 OPC 数据访问项与 OPC UA 数据变量相对应。
- 面向快速逻辑的 OPC 数据访问项与 OPC UA 数据方法相对应。

合格的 NodId 标识符在 OPC UA 中比在“OPC  
数据访问”中更有意义。每次单独访问对象、子对象、特性及属性均使用其 NodId。

OPC UA 提供显示名称等内容以支持本地语言。

也就是说，可通过不同方式浏览相同对象（例如在 OPC UA  
客户端指定的各种语言环境下），每次都显示同一个 NodId。显示名称的选择与相关  
NodId 类似。整个命名空间的文本均采用英语。

## DP OPC UA 数据对象的语法

OPC UA 定义一个用于访问单个对象的优化的语法。所有 OPC UA 对象的 **NodeId** 均具有以下结构：

**<boardobject>. <object>."<subobject>".<property>**

子对象可包含其它子对象。

对无法解释的 **NodeId** 的访问将被拒绝，并显示错误消息。任何项中的字母“**A-Z**”都不区分大小写。

## 符号对象表示法

OPC UA 规范建议对地址空间的层级描述使用统一的符号表示法。

本文档中使用以下符号：

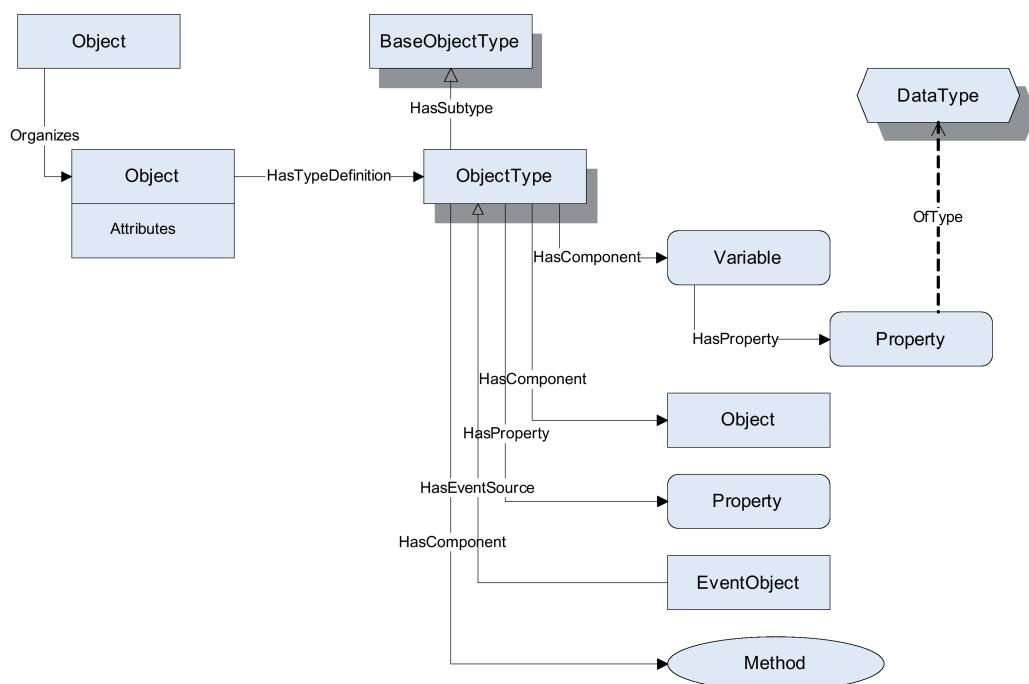


图 2-8 OPC UA 地址空间的符号

## 2.5.6 DP 模块的板对象

### 2.5.6.1 DP 模块的板对象概述

#### DP 板的类型

始终将所有协议特定的对象分配至一个板，对于 DP，这个板通常是 DP 主站模块或 DP 从站模块。也存在异常情况，即所谓的系统板和 DP 演示模块。

- DP 主站模块  
DP 主站模块用于与 DP 从站模块交换数据，通常使用 STEP 7 进行组态。
- DP 从站模块  
DP 从站模块用于与 DP 主站模块交换数据，通常使用 STEP 7 进行组态。
- 演示模块  
名称为“DEMO”的演示模块用于模拟带有几个简单从站的 DP 主站。

#### 演示模块

`<boardobject>:= "DEMO"`

对于名称为“DEMO”的演示模块，有的对象的命名空间与 DP 主站模块的命名空间类似。演示模块旨在使您熟悉 SIMATIC NET OPC 系统，可以通过组态激活该模块。

`<boardobject>:= "DEMO_S"`

对于名称为“DEMO\_S”的演示模块，有的对象的命名空间与 DP 从站模块的命名空间类似。演示模块旨在使您熟悉 SIMATIC NET OPC 系统，可以通过组态激活该模块。

---

#### 说明

名称为“DEMO”和“DEMO\_S”的演示模块一定不能与名称相同的 DP 主站模块或 DP 从站模块同时使用。如果一个演示模块被添加至组态，则将忽略用此名称组态的 DP 主站模块或 DP 从站模块。

#### DP 板对象是什么？

始终将所有生产协议特定的对象分配到板。在 DP 中，这些就是 DP 主站模块或 DP 从站模块（板）。演示板和 DEMO\_S 板除外。

## 2.5.6.2 板名称

### DP 模块的板名称

板名称是在 STEP 7 或 TIA 门户中组态的用于标识模块的 DP 名称。在 STEP 7 中，此名称称为“本地 ID”。在 OPC 服务器内，本地 ID 是唯一的。

#### 板类型

OPC 服务器支持以下板类型：

- DP 主站模块
- DP 从站模块

#### DP 板名称允许使用哪些字符？

对于 **<boardname>**，可使用数字“0-9”、大小写字母字符“A-z”和特殊字符“\_-+()”。板名称的长度最多可为 24 个字符。名称不区分大小写。

---

#### 说明

在 SIMATIC NET 中，通常如下例中所示使用模块名称。

---

板名称“SYSTEM”、“DEMO”和“DEMO\_S”已保留，不得使用。

#### 板名称示例

典型示例包括：

- CP5613A2
- CP5624Slave
- DPMaster

### 2.5.6.3 DP 主站板对象的类型定义

#### DP 主站板对象的类型定义

对于可以通过生产 DP 主站板使用的对象和功能，将定义特定的 OPC UA 对象类型：

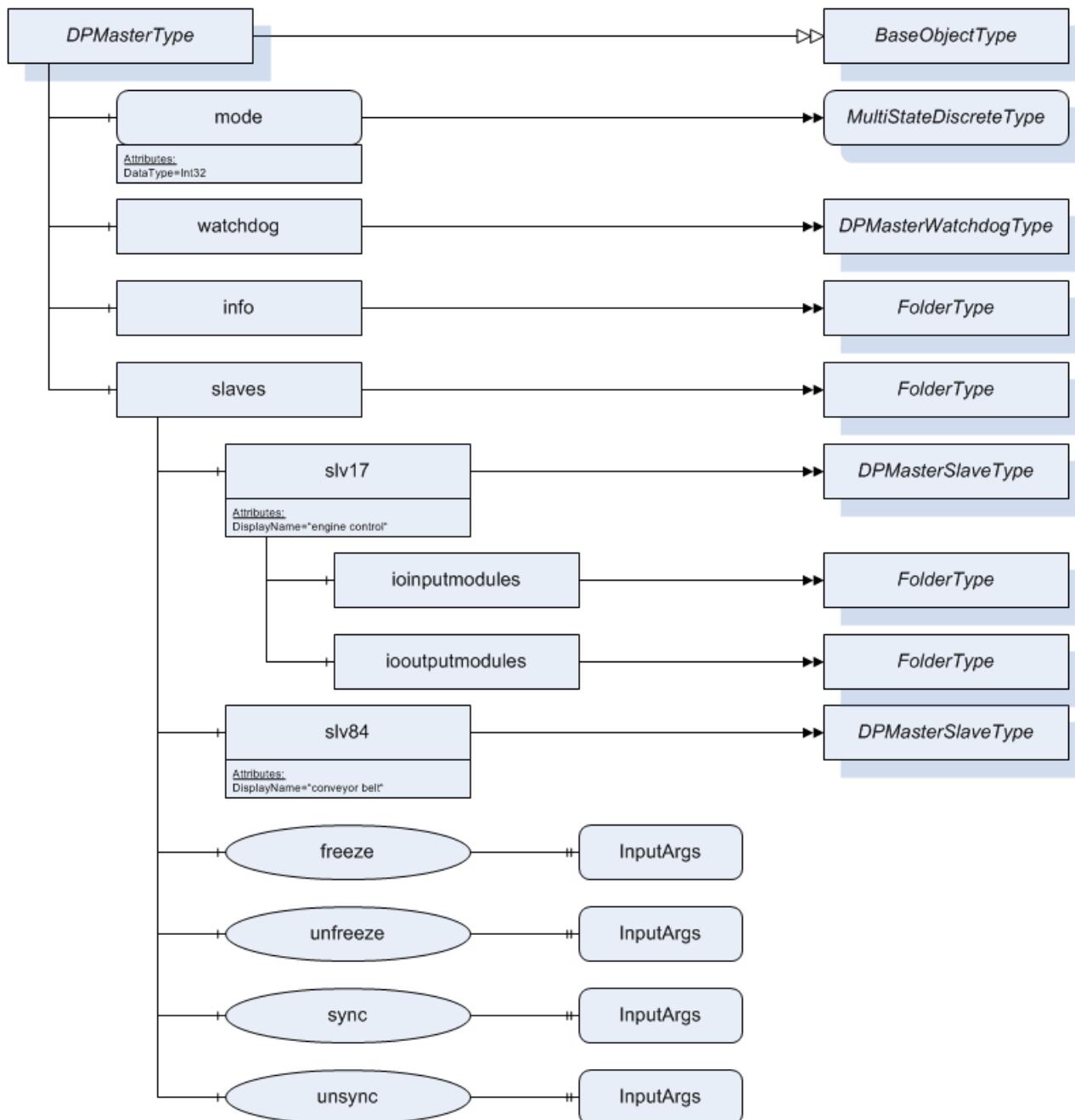


图 2-9 OPC UA 命名空间内 DP 主站板对象的类型

针对对象的 OPC UA 命名空间内显示了此类型的实例。  
能在“类型”下构造化读取类型本身。

#### 2.5.6.4 DP 从站板对象的类型定义

##### DP 从站板对象的类型定义

对于可以通过生产 DP 从站板使用的对象和功能，将定义特定的 OPC UA 对象类型：

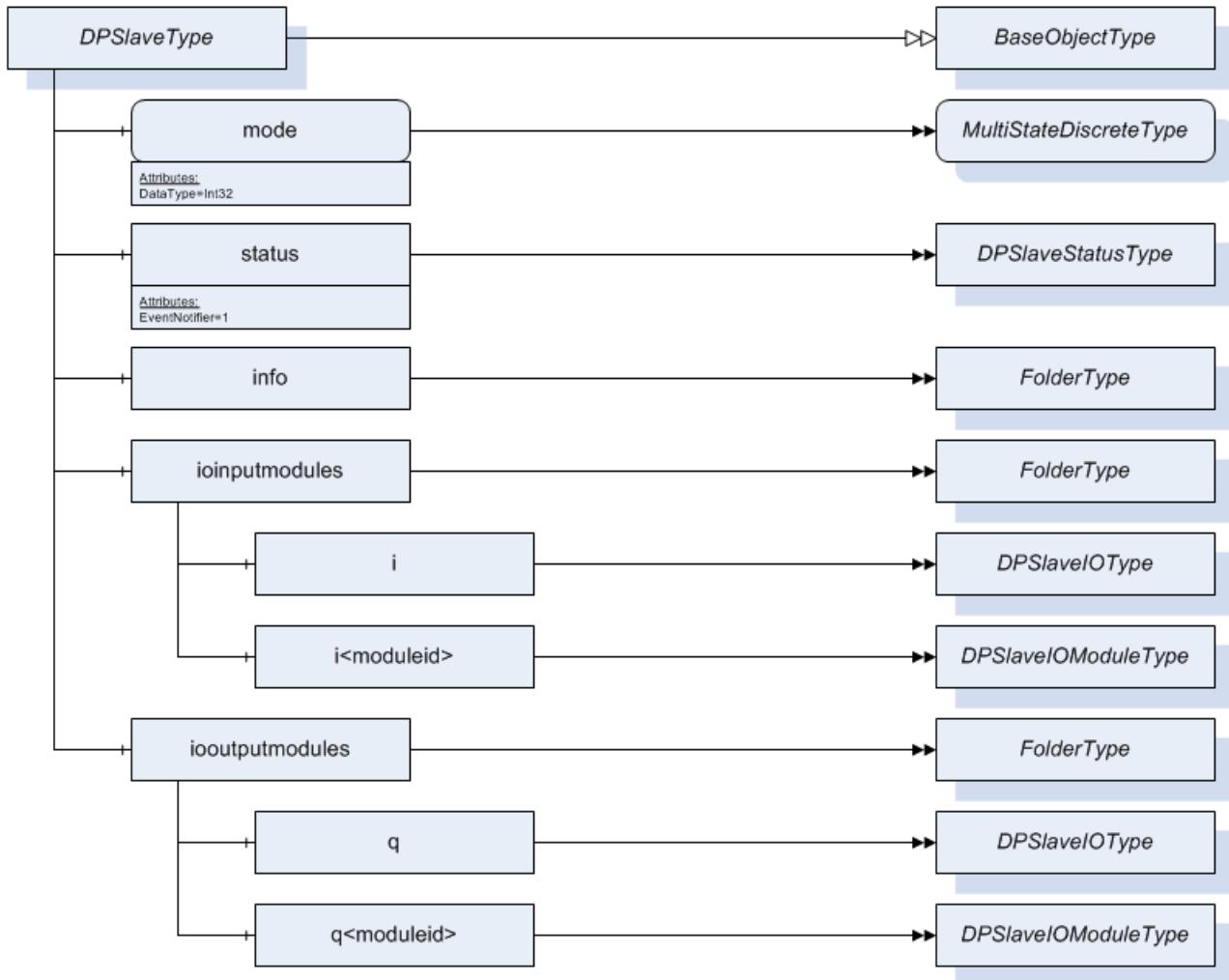


图 2-10 OPC UA 命名空间内 DP 从站板对象的类型

针对对象的 OPC UA 命名空间内显示了此类型的实例。  
能在“类型”下构造化读取类型本身。

## 2.5.7 1类 DP 主站

### 2.5.7.1 1类 DP 主站服务的过程变量

利用访问周期性数据的服务，可以访问从站的输入和输出，并对其进行监视和控制。

通过以下方式进行访问：

- 从站编号  
此编号与 PROFIBUS 地址相对应。
- 模块编号  
DP 从站可以包括带有不同输入/输出区域的几个子模块。
- 输入/输出区域

### 2.5.7.2 1类主站过程变量的语法

#### 过程变量的语法

DP OPC UA NodeID 的过程变量的优化语法（用于读取和写入数据变量）：

命名空间 URI: DP: (命名空间索引: 3)

#### 语法

可用选项如下：

- <DPmaster>.<slave>.i{<number>} { .<offset>, <DPtype>{ , <quantity> } }
- <DPmaster>.<slave>.q{<number>} { .<offset>, <DPtype>{ , <quantity> } }

#### 说明

##### <DPmaster>

协议特定的连接名称。连接名称在组态中指定。在 DP 协议中，连接名称是通信模块的已组态名称。

#### <slave>

从站符号名称，用作访问 DP 从站的标识符。可在 STEP 7/NCM PC 中分配从站的符号名称（必须组态 V8.2 或更高版本的 OPC 服务器）。

#### 说明

如果在 STEP 7/NCM PC 中组态 V8.2 以下版本的 OPC 服务器且未分配从站的符号名称，则会采用标识符“slv”后接组态的从站地址形式作为从站的符号名称。

如果在 STEP 7/NCM PC 中组态 V8.2 及以上版本的 OPC 服务器且未分配从站的符号名称，则会将从站的标识符用作从站名称。在 STEP 7/NCM PC 中执行一致性检查，确保使用唯一从站名称且从站名称不为空。

#### 说明

##### 从站名称允许使用的字符

从站名称允许使用字符集中的以下字符：

A-Z、a-z、0-9、\_、-、^、!、#、\$、%、'、(、)、=、~、+、'、'、@、{、}

从站名称不允许使用字符集中的以下字符：

- 句点“.”
- 冒号“:”
- 管道符“|”
- 反斜线“\”
- 方括号“[”和“]”
- 双引号“”
- 连字符“&”
- 问号“?”
- 尖括号“<”和“>”
- 星号“\*”

从站名称的开头和结尾不能使用空格。

#### q

输出的标识符。可读取和写入输出。

**i**

输入的标识符。输入为只读。

**<number>**

包含输入或输出区域的子模块的编号。

**<offset> (基于零的偏移量 - 第一个元素是零)**

定位要被寻址的元素所处从站中地址空间的字节偏移量。

如果指定了子模块，则在子模块内应用偏移。

如果未指定子模块，则偏移适用于从站的整个输入/输出区域。字节偏移量以零为基础。

**<DPtype>**

DP 数据类型转换为 OPC UA 服务器上相应的 OPC UA 数据类型。

下表列出了类型标识符以及可用于表示变量值的对应 OPC UA 数据类型。

数据类型	OPC UA 数据类型	注释
x<bitaddress>	布尔	位（布尔） 除区域内的字节偏移量外，还必须指定相关字节中的 <bitaddress>。 值范围是 0 到 7
b	字节	字节（无符号）
	字节字符串	如果未指定 <DPtype>，则用作默认值。OPC UA 不识别“Byte[]”，但对此使用标量数据类型“ByteString”。
w	UInt16	字（无符号）
dw	UInt32	双字（无符号）
lw	UInt64	长字（无符号）
c	SByte	字节（有符号）
i	Int16	字（有符号）
di	Int32	双字（有符号）
li	Int64	长字（有符号）
r	浮点	浮点（4 字节）
lr	双精度	浮点（8 字节）
s<stringlength>	字符串	必须指定为字符串保留的 <stringlength>。 值范围是 1 到 254 写入时还可写入较短的字符串，使传送的数据长度始终为保留的字符串长度（字节）。不必要的字节用值 0 填充。

**<quantity>**

元素的数量。变量的数据类型是具有指定格式元素的数组。指定多个数组元素将导致形成相应类型的数组，即使只对一个数组元素进行寻址也是如此。

**2.5.7.3 1类 DP 主站过程变量的示例**

下面几个示例说明了 DP 变量的变量名称的语法。

**输入**

命名空间 URI: DP: (命名空间索引: 3)

- CP 5623.conveyorbelt.i3.0  
标识“conveyorbelt”从站的子模块 3 的输入字节 0
- CP 5623.motorcontrol.i3.1,b,3  
标识从“motorcontrol”从站的子模块 3 的输入字节 1 开始的 3 字节数组
- CP 5623.weldingrobot.i3.2,dw  
标识从“weldingrobot”从站的子模块 3 的输入字节 2 开始的双字
- CP 5623.slv4.i3.0,r  
标识从从站 4 子模块 3 的输入字节 0 开始的浮点数
- CP 5623.slv4.i.0,b,8  
标识从站 4 所有子模块的整个输入区域的前 8 个字节

**输出**

命名空间 URI: DP: (命名空间索引: 3)

- CP 5623.conveyorbelt.q7.1  
标识“conveyorbelt”从站的子模块 7 的输出字节 1
- CP 5623.motorcontrol.q.7,2,x5  
标识“motorcontrol”从站子模块 7 的输出字节 2 中第 5 位
- CP 5623.slv4.q.0,w,8  
标识 8 字数组，来自覆盖所有子模块的从站 4 的输出区域

**2.5.7.4 DPC1 服务**

可使用 DPC1 服务来访问 DPC1 从站的数据记录。数据记录非周期性传送。

数据记录的含义由从站的供应商指定。例如，数据记录可用于驱动器的组态数据。

DPC1 变量注册完毕后，OPC 服务器仅能检查语法是否正确，而不会根据 DPC1 从站的组态检查变量在伙伴设备上是否有效以及数据记录大小是否足够。

### 2.5.7.5 DPC1 服务的过程变量的语法

#### 过程变量的语法

数据记录数据变量的 DP OPC UA Node ID 的过程变量优化语法：

命名空间 URI: DP: (命名空间索引: 3)

#### 典型语法

可读和可写数据记录（没有可写数据记录长度信息）：

- <DPmaster>.<slave>.s<slot>.dr<index>{,<length>} {<offset>{,<DPtype>{,<quantity>}}}

可读数据记录：

- <DPmaster>.<slave>.s<slot>.dr<index>{<offset>{,<DPtype>{,<quantity>}}}

#### 说明

##### **<DPmaster>**

协议特定的连接名称。连接名称在组态中指定。在 DP 协议中，连接名称是通信模块的已组态名称。

##### **<slave>**

从站符号名称，用作访问 DP 从站的标识符。可在 STEP 7/NCM PC 中分配从站的符号名称（必须组态 V8.2 或更高版本的 OPC 服务器）。

---

#### 说明

如果在 STEP 7/NCM PC 中组态 V8.2 以下版本的 OPC 服务器且未分配从站的符号名称，则会采用标识符“slv”后接组态的从站地址形式作为从站的符号名称。

如果在 STEP 7/NCM PC 中组态 V8.2 及以上版本的 OPC 服务器且未分配从站的符号名称，则会将从站的标识符用作从站名称。在 STEP 7/NCM PC 中执行一致性检查，确保使用唯一从站名称且从站名称不为空。

## 说明

### 从站名称允许使用的字符

从站名称允许使用字符集中的以下字符：

A-Z、a-z、0-9、\_、-、^、!、#、\$、%、'、(、)、=、~、+、'、'、@、{、}

从站名称不允许使用字符集中的以下字符：

- 句点“.”
- 冒号“:”
- 管道符“|”
- 反斜线“\”
- 方括号“[”和“]”
- 双引号“”
- 连字符“&”
- 问号“?”
- 尖括号“<”和“>”
- 星号“\*”

从站名称的开头和结尾不能使用空格。

---

## s

从站插槽标识符

### <slot>

从站扩展存储区中的插槽，用于非周期性服务。插槽和索引用于标识数据记录。

### dr

用于标识访问数据记录的标识符。

### <index>

从站扩展存储区中插槽内的索引，用于非周期性服务。插槽和索引用于标识数据记录。

### <length>

记录的长度。范围介于 1 至 240 之间。

### <offset>

要寻址元素在数据记录中的字节地址。

**<DPtype>**

数据类型。

数据类型转换为 OPC UA 服务器上对应的 OLE 数据类型。

数据类型	OPC UA 数据类型	注释
x<bitaddress>	布尔	位 (布尔) 除区域内的字节偏移量外, 还必须指定相关字节中的 <bitaddress>。 值范围是 0 到 7
b	字节	字节 (无符号)
	字节字符串	如果未指定 <DPtype>, 则用作默认值。OPC UA 不识别“Byte[]”, 但对此使用标量 数据类型“ByteString”。
w	UInt16	字 (无符号)
dw	UInt32	双字 (无符号)
lw	UInt64	长字 (无符号)
c	SByte	字节 (有符号)
i	Int16	字 (有符号)
di	Int32	双字 (有符号)
li	Int64	长字 (有符号)
r	浮点	浮点 (4 字节)
lr	双精度	浮点 (8 字节)
s<stringlength>	字符串	必须指定为字符串保留的 <stringlength>。 值范围是 1 到 254 写入时还可写入较短的字符串, 使 传送的数据长度始终为保留的字符 串长度 (字节)。不必要的字节用 值 0 填充。

#### <quantity>

元素的数量。变量的数据类型是具有指定格式元素的数组。指定多个数组元素将导致形成相应类型的数组，即使只对一个数组元素进行寻址也是如此。

#### 2.5.7.6 DPC1 服务的过程变量的示例

下面几个示例说明了 DPC1 服务的变量名称的语法。

#### DPC1 的变量名称

CP 5613.motorcontrol.s3.dr2,120.7,dw

标识访问起始于数据记录中的偏移 7 的双字，数据记录长度为 120 个字节，位于从站“motorcontrol”的插槽 3 索引 2。

CP 5613.slv5.s3.dr2,120.8,b,4

访问具有 4 个字节的数组，该数组起始于数据记录中的偏移 8，数据记录长度为 120 个字节，位于从站 5 的插槽 3 索引 2。

#### 2.5.7.7 CP 5613/CP 5614/CP 5623/CP 5624 的快速逻辑（仅限主站）

CP 5613/CP 5623 以及 CP 5614/CP 5624 的 DP

主站部分支持快速逻辑属性。也就是说可以设置 CP 参数，以便在与某一从站进行数据交换时可以将值写入同一从站或其它从站。进行数据交换时也会通知用户应用程序。

CP 5613/CP 5614 和 CP 5623/CP 5624 提供 6 个快速逻辑触发器，可使用 OPC UA 数据变量和方法对其进行配置和评估。

## 快速逻辑的优点

使用快速逻辑具有以下优点：

- OPC 服务器和 OPC 客户端上的负载较小。
- 数据传输速度更快，因为数据传输不依靠在 CP 上运行的软件，而是直接在 CP 的硬件中进行。

---

### 说明

快速逻辑触发器在触发后会再次自动禁用。

然后需要使用“fl<fastlogicid>.on”方法再次激活触发器。

只有在 DP 主站处于 OPERATE 模式且涉及的从站处于 READY 状态时，快速逻辑功能才能正常工作。因此，只有在用户程序将 DP 主站带入“OPERATE”模式且涉及的从站处于“READY”状态时，才应该通过 DP 应用程序激活快速逻辑触发器。

只要快速逻辑触发器处于激活状态，DP 用户程序就不能通过快速逻辑写入连接到输入字节的输出字节。

---

## 2.5.7.8 快速逻辑的控制变量的语法

### OPC UA 数据变量的语法

命名空间 URI: DP: (命名空间索引: 3)

```
<DPmaster>.fl<fastlogicid>.state  
<DPmaster>.fl<fastlogicid>.fastlogicid
```

### OPC UA 方法的语法

命名空间 URI: DP: (命名空间索引: 3)

```
<DPmaster>.fl<fastlogicid>.activate  
<DPmaster>.fl<fastlogicid>.clear
```

### 说明

#### **<DPmaster>**

协议特定的连接名称。连接名称在组态中指定。在 DP 协议中，连接名称是通信模块的已组态名称。

**f**

快速逻辑标识符。

**<fastlogicid>**

正在使用的快速逻辑触发器编号。值介于 1 至 4 之间。

**state**

返回快速逻辑状态。

返回值：

- CLEARED (0)

触发器 <fastlogicid> 未激活。

- ACTIVATED (1)

触发器 <fastlogicid> 已激活。

- TRIGGERED (2)

触发器 <fastlogicid> 已执行监视。

**fastlogicid**

返回正在使用的快速逻辑触发器 ID。

**activate**

激活并调用快速逻辑触发器。

“activate”OPC UA 方法作为参数随调用一起发送。

如果发送参数块，则为 OPC UA 方法中指定的触发器 <fastlogicid> 设置快速逻辑属性。

发送的参数块是一个 8 字节数组，结构如下：

- **slave\_addr\_in\_byte**

已针对触发器选择的输入所属从站的地址

- **index\_in\_byte**

触发器的输入字节偏移量

- **cmp\_value\_in\_byte**

输入字节的比较值

- **mask\_in\_byte**

您可以屏蔽输入字节中的单个位，以便在比较时将它们忽略。

通过将相应位中的值置“1”来屏蔽位，也就是说如果设置 **mask\_in\_byte==0x00**，则 **cmp\_value\_in\_byte** 的所有位都用于比较。所选输入字节中的所有未屏蔽位都与 **cmp\_value\_in\_byte** 中的对应位匹配时，触发器即会触发。

- **slave\_addr\_out\_byte**

选择出现触发条件时将更改的输出字节所属的从站

- **index\_out\_byte**  
输出字节的偏移量
- **value\_out\_byte**  
将写入的输出字节的值
- **mask\_out\_byte**  
可以单独屏蔽输出字节中的位，以在满足触发条件时不修改这些位。  
通过将相应位中的值置“1”来屏蔽位，也就是说如果设置 **mask\_out\_byte==0x00**，则 **value\_out\_byte** 中的所有位都写入所选输出字节。

**clear**

禁用快速逻辑。

“off”OPC UA 方法只能写入。 它没有任何参数。

## 示例

OPC UA 变量: CP 5614.fl4.state

OPC UA 方法: CP 5623.fl3.clear

### 2.5.7.9 控制帧同步和冻结

对于特定的应用程序,有四条控制命令可用于发送控制帧。 标准应用不需要这些命令。

控制帧是主站发送到一个从站、一组、多组或所有从站的帧。 从站不确认这些帧。

控制帧用于将控制命令（称为全局控制）传送到所选从站以允许同步。

控制命令包括三个组成部分:

- 标识符，指示是否在对一个或多个 DP 从站进行寻址
- 从站组标识
- 控制命令

## 创建组

组态期间可将组标识符分配给从站；也就是说可在同一个组中包括多个从站。

创建数据库时指定哪些从站属于一个组。 在此阶段为每个 DP 从站分配一个组号。

在参数分配阶段向 DP 从站通知这个组号。 最多可指定八个组。

### 2.5.7.10 同步和冻结控制变量的语法

#### OPC UA 方法的语法

命名空间 URI: DP: (命名空间索引: 3)

```
<DPmaster>.q.sync()  
  
<DPmaster>.q.unsync()  
  
<DPmaster>.i.freeze()  
  
<DPmaster>.i.unfreeze()  
  
<DPmaster>.<slave>.q.sync()  
  
<DPmaster>.<slave>.q.unsync()  
  
<DPmaster>.<slave>.i.freeze()  
  
<DPmaster>.<slave>.i.unfreeze()  
  
<DPmaster>.<slave>.state.<status>
```

#### 说明

##### **<DPmaster>**

协议特定的连接名称。连接名称在组态中指定。在 DP 协议中，连接名称是通信模块的已组态名称。

##### **<slave>**

从站符号名称，用作访问 DP 从站的标识符。可在 STEP 7/NCM PC 中分配从站的符号名称（必须组态 V8.2 或更高版本的 OPC 服务器）。

---

#### 说明

如果在 STEP 7/NCM PC 中组态 V8.2 以下版本的 OPC 服务器且未分配从站的符号名称，则会采用标识符“slv”后接组态的从站地址形式作为从站的符号名称。

如果在 STEP 7/NCM PC 中组态 V8.2 及以上版本的 OPC 服务器且未分配从站的符号名称，则会将从站的标识符用作从站名称。在 STEP 7/NCM PC 中执行一致性检查，确保使用唯一从站名称且从站名称不为空。

---

---

## 说明

### 从站名称允许使用的字符

从站名称允许使用字符集中的以下字符：

A-Z、a-z、0-9、\_、-、^、!、#、\$、%、'、(、)、=、~、+、'、'、@、{、}

从站名称不允许使用字符集中的以下字符：

- 句点“.”
- 冒号“:”
- 管道符“|”
- 反斜线“\”
- 方括号“[”和“]”
- 双引号“”
- 连字符“&”
- 问号“?”
- 尖括号“<”和“>”
- 星号“\*”

从站名称的开头和结尾不能使用空格。

---

## q

输出的标识符。

### sync()

使用“sync”OPC UA 方法调用，所选组中所有 DP 从站的当前输出状态都将冻结。此后 DP 主站发送的所有输出数据最初不被 DP 从站接受。

写入的参数块是一个字节，结构如下：

- **slavegroups**

字节的每一位代表 8 个可组态从站组中的一个。位 0 代表第一个从站组。

如果重复执行 sync 调用，主站的当前输出数据传送到从站的输出且输出再次冻结。

**unsync()**

“unsync”OPC UA 方法调用取消 DP 从站输出冻结。收到 unsync 调用后，DP 从站重新接受 DP 主站周期性发送的所有输出数据。

写入的参数块是一个字节，结构如下：

- **slavegroups**

字节的每一位代表 8 个可组态从站组中的一个。位 0 代表第一个从站组。

**i**

输入的标识符。

**freeze()**

使用“freeze”OPC UA 方法调用，所选组的所有 DP 从站输入的当前状态都被冻结。在随后的读取周期中，DP 主站接收冻结的输入数据。

发送的参数块是一个字节，结构如下：

- **slavegroups**

字节的每一位代表 8 个可组态从站组中的一个。位 0 代表第一个从站组。

如果重复发送 freeze 调用，主站的当前输入数据传送到从站的输入且输入再次冻结。

**unfreeze()**

“unfreeze”OPC UA 方法调用取消 DP 从站输入冻结。收到 unfreeze 调用后，DP 从站重新为 DP 主站提供当前输入数据。

发送的参数块是一个字节，结构如下：

- **slavegroups**

字节的每一位代表 8 个可组态从站组中的一个。位 0 代表第一个从站组。

**state**

从站状态方法或变量的标识符。

**<status>**

状态	说明
activate()	激活从站
deactivate()	禁用从站
restart()	重新启动从站
state	表示从站状态的变量 (EnumString)。

**示例**

CP 5614.q.sync()

CP 5614.motorcontrol.state.state

### 2.5.7.11 DP 特定的信息变量

#### DP 特定数据信息变量

可使用 DP 主站特定数据变量获取 DP 主站的状态、属性或类型信息。

可获取以下信息：

- DP 主站模式 (mode)
- DP 主站的模块信息 (info)
- DP 主站的看门狗 (watchdog)
- DP 主站通过从站了解的从站参数。

#### DP 特定信息变量的语法

Nodeld:

命名空间索引： 3

*<masterobject>. <informationparameter>*

#### DP 主站的模式

#### DP 主站模式 (mode) 的语法

命名空间 URI: DP: (命名空间索引: 3) 3)

*<DPmaster>. <informationparameter>*

#### 说明

##### **<DPmaster**

协议特定的连接名称。连接名称在组态中指定。在 DP 协议中，连接名称是通信模块的已组态名称。

## &lt;informationparameter&gt;

模块信息	说明
mode	DP 主站的当前模式。当前模式可读可写。只有在 DP 应用程序环境的上下文中，才能通过写入如下所示的其中一个值来设置模式。 OPC UA 类型“MultistateDiscreteType”的数据变量，读写 “mode”可以返回如下值：
OFFLINE (0)	主站与从站之间没有通信。
STOP (1)	除诊断数据外，主站与从站间之间没有通信。
CLEAR (2)	参数分配和组态阶段
AUTOCLEAR (3)	自动清除阶段，DP 主站不能再访问所有从站
OPERATE (4)	生产阶段

## 示例：

CP 5614.mode

## DP 主站的模块信息

## DP 主站的模块信息 (info) 的语法

命名空间 URI: DP: (命名空间索引: 3)

&lt;DPmaster&gt;.&lt;moduleinformation&gt;

## 说明

## &lt;DP 主站&gt;

协议特定的连接名称。连接名称在组态中指定。在 DP 协议中，连接名称是通信模块的组态名称。

## &lt;moduleinformation&gt;

模块信息	说明
标识号	认证标识号 OPC UA 类型“UInt16”的数据变量，只读

hardware	硬件 UA 类型“MultistateDiscreteType”的数据变量，只读。	
	0	SOFTNET
	1	CP 5613 (电气 PROFIBUS 端口)
	2	CP 5613 FO (光学 PROFIBUS 端口)
	3	CP 5614 (电气 PROFIBUS 端口)
	4	CP 5614 FO (光学 PROFIBUS 端口)
	5	CP 5614 FO (光学 PROFIBUS 端口) 连接外部电源
	6	CP 5613 FO (光学 PROFIBUS 端口) 连接外部电源
	7	CP 5613 A2 (电气 PROFIBUS 端口)
	8	CP 5614 A2 (电气 PROFIBUS 端口)
	9	CP 5623 (电气 PROFIBUS 端口)
	10	CP 5624 (电气 PROFIBUS 端口)
	11	CP 5613 A3 (电气 PROFIBUS 端口)
	12	CP 5614 A3 (电气 PROFIBUS 端口)
hardwareversion	硬件版本 OPC UA 的数据变量类型为“字节”，只读	
firmwareversion	固件版本 OPC UA 的数据变量类型为“UInt16”，只读	
buspar	读取主站的总线参数 OPC UA 的数据变量类型为“ByteString”，只读	

示例：

CP 5614.hardware

**DP 主站的看门狗**

**DP 主站的看门狗的语法**

命名空间 URI: DP: (命名空间索引: 3)

&lt;DPmaster&gt;.watchdog.&lt;watchdogvariable&gt;

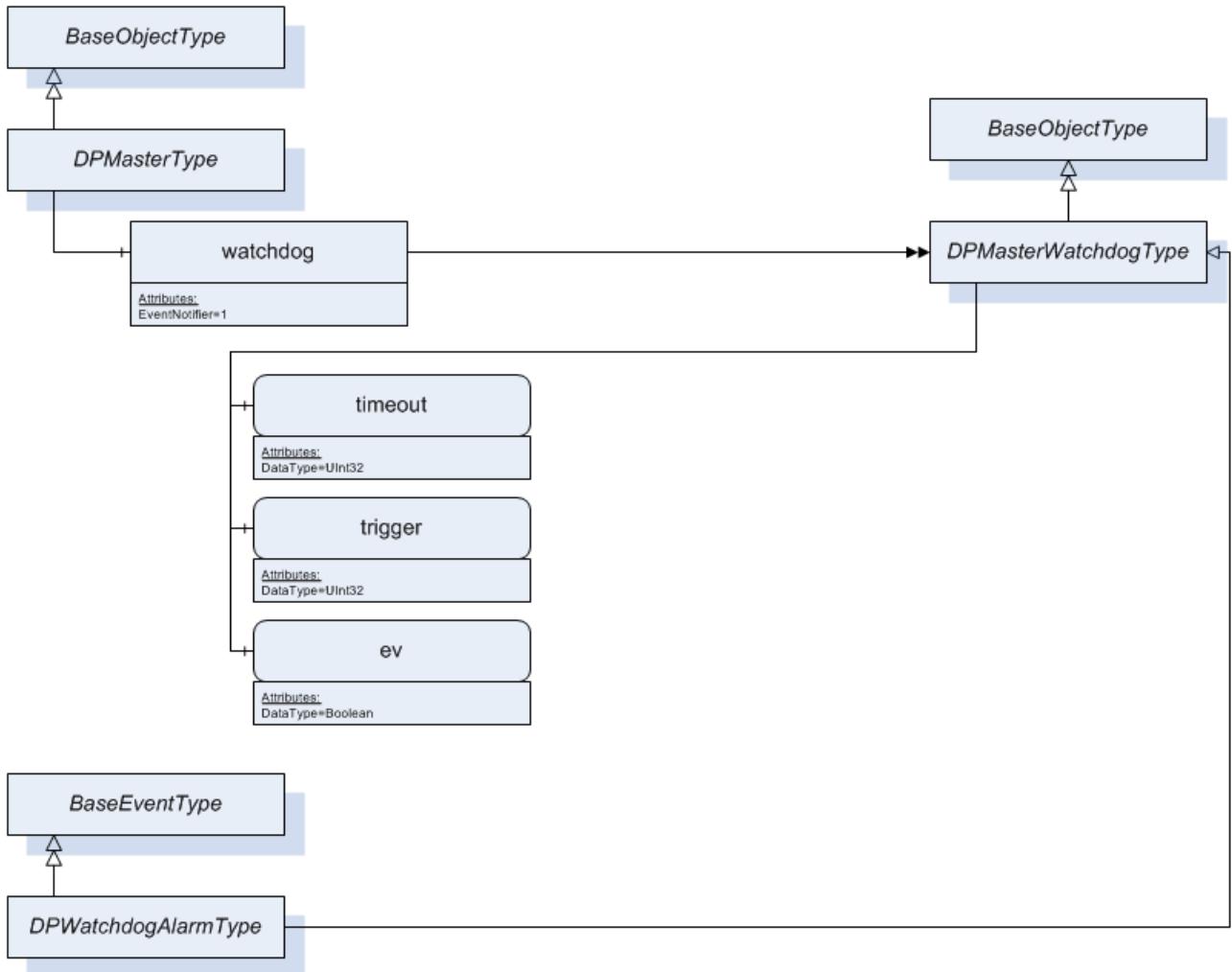


图 2-11 DP 主站的看门狗

<b>watchdog.timeout</b>	读出、设置、复位或禁用看门狗超时。OPC UA 服务器将写入的值自动舍入下一高值。 OPC UA 类型“UInt32”的数据变量，读写，单位是毫秒。	
	0	关闭看门狗

	值范围: 20 到 102,000 ms (含)	设置看门狗超时 Softnet DP (如 CP 5621) 的粒度: 400 ms Hardnet DP (如 CP 5623) 的粒度: 10 ms
watchdog.trigger	监视 OPC UA 服务器的看门狗功能。 OPC UA 类型“UInt32”的数据变量, 只读	
watchdog.ev	触发看门狗监视 OPC UA 类型“布尔”的数据变量, 只读	
	False	正常运行
	True	DP 协议栈已触发看门狗。

**示例:**

CP 5614.watchdog.ev

CP 5614.watchdog.timeout

**DP 主站上的 DP 从站****DP 主站上的 DP 从站信息变量的语法**

命名空间 URI: DP: (命名空间索引: 3)

&lt;DPmaster&gt;.&lt;slave&gt;.&lt;informationvariables&gt;

**说明****<DPmaster>**

协议特定的连接名称。连接名称在组态中指定。在 DP 协议中, 连接名称是通信模块的已组态名称。

**<slave>**

从站符号名称，用作访问 DP 从站的标识符。可在 STEP 7/NCM PC 中分配从站符号名称。

**说明**

如果在 STEP 7/NCM PC 中组态 V8.2 以下版本的 OPC 服务器且未分配从站的符号名称，则会采用标识符“slv”后接组态的从站地址形式作为从站的符号名称。

如果在 STEP 7/NCM PC 中组态 V8.2 及以上版本的 OPC 服务器且未分配从站的符号名称，则会将从站的标识符用作从站名称。在 STEP 7/NCM PC 中执行一致性检查，确保使用唯一从站名称且从站名称不为空。

**<address>**

DP 从站的 PROFIBUS 地址。

范围： 0 ... 126.

**<informationvariables>**

信息变量	说明		
type	标识 DP 从站类型。 UA 类型“MultistateDiscreteType”的数据变量，只读。		
	0	NORM	标准 DP 从站
	1	ET200_U	非标准从站： ET 200 U
	2	ET200K_B	非标准从站： ET 200 K/B
	3	ET200_SPM	非标准从站： 常规 SPM 站
dpv	查询 DP 从站的 DP 协议版本。 UA 类型“MultistateDiscreteType”的数据变量，只读。		
	0	DP-V0	
	1	DP-V1	
	2	DP-V2	
parcfgdata	使用此信息变量，CP 可以查询 DP 从站的 IO 子模块组态。参数源自组态。 UA 类型“ByteString”的数据变量，只读		
parprmdata	可使用此信息变量查询 DP 从站的参数分配数据。 UA 类型“ByteString”的数据变量，只读		

信息变量	说明
paruserdata	可使用此信息变量查询 DP 从站的用户参数分配数据。 UA 类型“ByteString”的数据变量，只读
partype	可使用此信息变量查询从站常规参数，如 SI flag、slave type 和 octet string。 UA 类型“ByteString”的数据变量，只读
paraddtab	可使用此信息变量读出 DP 从站的各部分从站参数。 UA 类型“ByteString”的数据变量，只读

示例：

CP 5611.motorcontrol.type

CP 5614.slv17.dpv

### DP 主站上的 DP 从站 IO 信息变量的语法

命名空间 URI: DP: (命名空间索引: 3)

可用选项如下:

- <DPmaster>.<slave>.q.<slaveproperties>
- <DPmaster>.<slave>.i.<slaveproperties>
- <DPmaster>.<slave>.q<moduleID>.<slavemoduleproperties>
- <DPmaster>.<slave>.i<moduleID>.<slavemoduleproperties>

### 说明

#### <DPmaster>

协议特定的连接名称。连接名称在组态中指定。在 DP 协议中，连接名称是通信模块的已组态名称。

**<slave>**

从站符号名称，用作访问 DP 从站的标识符。可在 STEP 7/NCM PC 中分配从站符号名称。

**说明**

如果在 STEP 7/NCM PC 中组态 V8.2 以下版本的 OPC 服务器且未分配从站的符号名称，则会采用标识符“slv”后接组态的从站地址形式作为从站的符号名称。

如果在 STEP 7/NCM PC 中组态 V8.2 及以上版本的 OPC 服务器且未分配从站的符号名称，则会将从站的标识符用作从站名称。在 STEP 7/NCM PC 中执行一致性检查，确保使用唯一从站名称且从站名称不为空。

**q**

输出的标识符。输出可读可写

**i**

输入的标识符。输入为只读。

**<moduleId>**

包含输入或输出区域的子模块的 ID。

**<slaveproperties>**

IO 信息变量	说明	
length	DP 从站组态数据中的模块长度（字节）。 UInt16 类型的属性，只读	
module type	DP 从站组态数据中的模块类型。 UA 类型“MultistateDiscreteType”的数据变量，只读。  0   未知 1   I (输入) 2   Q (输出) 3   IQ (输入/输出)  只有 DP 从站模块才会出现值“3”。 这种情况下，存在具有相同“moduleid”的输入和输出子模块。	
slaveid	相应 DP 从站的 PROFIBUS 地址 “字节”类型的属性，只读	

## &lt;slavemodeleproperties&gt;

IO 信息变量	说明	
length	DP 从站组态数据中的模块长度（字节）。 UInt16 类型的属性，只读	
module type	DP 从站组态数据中的模块类型。 UA 类型“MultistateDiscreteType”的数据变量，只读。	
	0	未知
	1	I (输入)
	2	Q (输出)
	3	IQ (输入/输出)
	只有 DP 从站模块才会出现值“3”。 这种情况下，存在具有相同“moduleid”的输入和输出子模块。	
slaveid	相应 DP 从站的 PROFIBUS 地址 “字节”类型的属性，只读	
moduleid	DP 从站组态数据中的模块地址。 “字节”类型的属性，只读	
slot	此子模块的插槽或模块标识符。 “字节”类型的属性，只读	
dataunitlength	DP 从站组态数据中的子模块数据单位长度。 UA 类型“MultistateDiscreteType”的数据变量，只读	
	0	未知
	1	字节
	2	字
consistence	DP 从站组态数据中的子模块所有数据单位的一致性。 “布尔”类型的属性，只读	

示例：

CP 5614.conveyorbelt.q.length

CP 5614.slv17.q.moduletype

CP 5614.slv17.q0.length

## DP 主站上的 DP 从站诊断变量的语法

命名空间 URI: DP: (命名空间索引: 3)

```
<DPmaster>.<slave>.diagnosis.data{.<diagnosticsvariables>}
```

### 说明

#### **<DPmaster>**

协议特定的连接名称。连接名称在组态中指定。在 DP 协议中，连接名称是通信模块的已组态名称。

#### **<slave>**

从站符号名称，用作访问 DP 从站的标识符。可在 STEP 7/NCM PC 中分配从站符号名称。

---

### 说明

如果在 STEP 7/NCM PC 中组态 V8.2 以下版本的 OPC 服务器且未分配从站的符号名称，则会采用标识符“slv”后接组态的从站地址形式作为从站的符号名称。

如果在 STEP 7/NCM PC 中组态 V8.2 及以上版本的 OPC 服务器且未分配从站的符号名称，则会将从站的标识符用作从站名称。在 STEP 7/NCM PC 中执行一致性检查，确保使用唯一从站名称且从站名称不为空。

---

#### **diagnosis**

DP 从站的诊断数据。

#### **data**

包含 DP 从站的上次诊断数据。前 6 个字节包含 DP 从站的标准诊断。

“ByteString”类型的数据变量，只读

## &lt;diagnosticsvariables&gt;

诊断变量	说明
masterlock	诊断数据的字节 1 中的位 7。 已由其它 DP 主站为 DP 从站分配参数；也就是说，本地 DP 主站当前无法访问此 DP 从站。 “布尔”类型的数据变量，只读
prmfault	诊断数据的字节 1 中的位 6。 如果最后一个参数分配帧不正确（如长度错误、标识号错误或参数无效），DP 从站将此位置位。 “布尔”类型的数据变量，只读
invalidslaveresponse	诊断数据的字节 1 中的位 5。 只要从寻址的 DP 从站收到不合理的响应，就将此位置位。 “布尔”类型的数据变量，只读
notsupported	诊断数据的字节 1 中的位 4。 只要请求 DP 从站不支持的功能（例如 DP 从站不支持的 SYNC 模式下运行），就将此位置位。 “布尔”类型的数据变量，只读
extdiag	诊断数据的字节 1 中的位 3。 此位由 DP 从站置位。 如果此位置位，则从站特定诊断区域（Ext_Diag_Data，字节 7-32）必须有诊断条目。 如果此位不置位，则从站特定诊断区域（Ext_Diag_Data，字节 7-32）可能有状态消息。 状态消息的含义必须与每个特定应用程序协商。 “布尔”类型的数据变量，只读
cfgfault	诊断数据的字节 1 中的位 2。 DP 主站上次发送的组态数据与 DP 从站确定的数据不匹配，即有组态错误时，则将此位置位。 “布尔”类型的数据变量，只读
stationnotready	诊断数据的字节 1 中的位 1。 DP 从站未准备好进行生产数据交换时，此位置位。 “布尔”类型的数据变量，只读

诊断变量	说明
stationnonexistent	<p>诊断数据的字节 1 中的位 0。</p> <p>无法通过总线访问 DP 从站时, DP 主站将此位置位。如果此位置位, 则诊断位包含上一诊断消息的状态或初始值。DP 从站始终将此位置零。</p> <p>“布尔”类型的数据变量, 只读</p>
deactivated	<p>诊断数据的字节 2 中的位 7。</p> <p>如果 DP 从站在本地参数集中指示为未激活, 并且已排除在外, 不参与周期性处理, 则会对此位进行置位。</p> <p>“布尔”类型的数据变量, 只读</p> <p>注: 位 6 保留。</p>
syncmode	<p>诊断数据的字节 2 中的位 5。</p> <p>只要收到 Sync 控制命令, DP 从站就将此位置位。</p> <p>“布尔”类型的数据变量, 只读</p>
freezemode	<p>诊断数据的字节 2 中的位 4。</p> <p>只要收到 Freeze 控制命令, DP 从站就将此位置位。</p> <p>“布尔”类型的数据变量, 只读</p>
wdon	<p>诊断数据的字节 2 中的位 3。</p> <p>此位由 DP 从站置位。如果此位置 1, 则在 DP 从站上激活看门狗监视。</p> <p>“布尔”类型的数据变量, 只读</p>
statdiag	<p>诊断数据的字节 2 中的位 1。</p> <p>如果 DP 从站将此位置位, 则 DP 主站必须接收诊断信息, 直至此位再次清零为止。例如在 DP 从站无法提供有效用户数据时, 它将此位置位。</p> <p>“布尔”类型的数据变量, 只读</p> <p>注: 位 2: DP 从站始终将此位置 1。</p>

诊断变量	说明
prmreq	<p>诊断数据的字节 2 中的位 0。</p> <p>如果 DP 从站将此位置位，则必须为从站重新分配参数和重新组态。此位保持置位状态，直到参数已分配。</p> <p>“布尔”类型的数据变量，只读。</p> <p>注：</p> <p>如果位 0 和位 1 同时置位，则位 0 的优先级更高。</p>
extdiagoverflow	<p>诊断数据的字节 3 中的位 7。</p> <p>如果此位置位，则可用诊断信息多于“Ext_Diag_Data”所示信息。例如在通道诊断数据多于 DP 从站可以输入其发送缓冲区的数据时，DP 从站将此位置位；或在 DP 从站发送的诊断数据量大于 DP 主站的诊断缓冲区容量时，DP 主站将此位置位。</p> <p>“布尔”类型的数据变量，只读</p>
masteraddr	<p>诊断数据的字节 4。</p> <p>将参数分配到此 DP 从站的 DP 主站的地址在字节 4“Diag.Master_Add”字节中输入。如果 DP 主站未给 DP 从站分配参数，则 DP 从站在此字节中设置地址 255。</p> <p>“字节”类型的数据变量，只读</p> <p>注：</p> <p>位 6 到位 0 保留。</p>
标识号	<p>诊断数据的字节 5 和 6。</p> <p>在字节 5 和 6 (“Ident_Number”字节) 中分配 DP 从站类型的供应商 ID。此标识符既可用于测试，又可用于精确标识从站。</p> <p>“Word”类型的数据变量，只读</p>
extdiagdata	<p>诊断数据的字节 7-32。</p> <p>从字节 7 (“Ext_Diag_Data”字节) 开始，DP 从站可以储存特定诊断信息。</p> <p>需要设备相关诊断和标识符相关诊断都有标头字节的块结构。</p> <p>“ByteString”类型的数据变量，只读</p>

#### 示例：

CP 5611.motorcontrol.diagnosis.data

CP 5611.slv17.diagnosis.data.stationnotready

CP 5611.slv17.diagnosis.data.identnumber

## DP 特定的信息变量的示例

下面几个示例说明了 DP 特定信息变量名称的语法。

### DP 主站的硬件信息

Nodeld:

- 命名空间 URI:

*DP: (命名空间索引: 3) 3)*

- 标识符:

*CP 5611 A2.hardware*, “int”类型

可能的返回值: SOFTNET、CP5613 和 CP5614 等

### DP 主站的看门狗

Nodeld:

- 命名空间 URI:

*DP: (命名空间索引: 3) 3)*

- 标识符:

– *CP 5624.watchdog.timeout*

可能的返回值: 0 = 看门狗未激活

– *CP 5624.watchdog.trigger*

可能的返回值: 2000 = DP OPC UA 服务器的看门狗功能

– *CP 5624.watchdog.ev*

可能的返回值: True = 看门狗已激活

## 2.5.8 DP 从站

### 说明

将 PROFIBUS 通信处理器作为 DP 从站时，只有将 OPC 作为 DP-V0 从站才能运行。由于指定了 HARDNET 模块，因此不能通过 OPC 读取或写入任何数据记录。

### 2.5.8.1 用于访问本地 DP 从站数据的变量服务

SIMATIC NET 的 OPC 服务器在 PROFIBUS DP 网络中不但可作为 DP 主站运行，还可扮演 DP 从站的角色。OPC 服务器管理该 DP 从站的输入和输出存储区，并将它们映射到 OPC 变量。一个 OPC 客户端可以读取主站设置的输出并在下一个循环中会被主站获取的输入中设置值。

DP 从站具有模块化结构。一个 DP

从站可以包括几个带有不同输入/输出区域的子模块。在组态期间分配子模块。

变量名称用于标识从站子模块中的输入或输出区域。

通过指定子模块编号和输入或输出区域访问从站的输入和输出。

也可以通过变量名称查询从站和 DP 主站的状态信息。

### 2.5.8.2 DP 从站过程变量的语法

#### 过程变量的语法

用于读写 IO 数据变量的 DP OPC UA Node ID 的过程变量优化语法：

命名空间 URI: DP: (命名空间索引: 3)

#### 典型语法

有两个选项：

- <DPslave>.i{ {<number>} { .<offset>{ ,<DPtype>{ ,<quantity>} } }
- <DPslave>.q{ {<number>} { .<offset>{ ,<DPtype>{ ,<quantity>} } }

## 说明

### <DPslave>

协议特定的连接名称。连接名称在组态中指定。在 DP 协议中，连接名称是通信模块的已组态名称。

**q**

输出标识符。输出为只读。

**i**

输入标识符。输入可读取和写入。

### <number>

包含输入或输出区域的子模块的编号。

### <offset>

要寻址元素所在从站地址空间中的字节偏移。

如果指定了子模块，则在子模块内应用偏移。

如果未指定子模块，则偏移适用于从站的整个输入/输出区域。

### <SRtype>

DP 数据类型转换为 OPC UA 服务器上相应的 OPC UA 数据类型。

下表列出了类型标识符以及可用于表示变量值的相应 OPC UA 数据类型。

数据类型	OPC UA 数据类型	注意
x<bitaddress>	布尔	位（布尔） 除区域内的字节偏移量外，还必须指定相关字节中的 <bitaddress>。 值范围是 0 到 7
b	字节	字节（无符号） 如果未指定 <DPtype>，则用作默认值。OPC UA 不识别“Byte[]”，但对此使用标量数据类型“ByteString”。
	字节字符串	
w	UInt16	字（无符号）
dw	UInt32	双字（无符号）
lw	UInt64	长字（无符号）
c	SByte	字节（有符号）
i	Int16	字（有符号）
di	Int32	双字（有符号）
li	Int64	长字（有符号）

数据类型	OPC UA 数据类型	注意
r	Float	浮点 (4 字节)
lr	Double	浮点 (8 字节)
s<stringlength>	String	必须指定为字符串保留的 <stringlength>。 值范围是 1 到 254 写入时还可写入较短的字符串，以使传送的数据长度始终为保留的字符串长度（字节）加 2 个字节。不必要的字节用值 0 填充。 读写字符串和字符串数组在内部映射到读写字节数组。

### <quantity>

元素的数量。变量的数据类型是具有指定格式元素的数组。

指定多个数组元素导致形成相应类型的数组，即使在只对一个数组元素进行寻址时也是如此。

#### 2.5.8.3 DP 从站的过程变量示例

您将在此处找到几个示例。

### 输入

- CP 5611.i3.0,b  
标识 DP 从站的子模 3 中的输入字节 0 (偏移 0)
- CP 5611.i3.1,b,3  
标识一个 3 字节数组，从 DP 从站的子模块 3 中的输入字节 1 (偏移 1) 开始
- CP 5614.i3.0,x0  
DP 从站的子模块 3 中字节 0 的输入位 0
- CP 5614.i3.3,b,8  
标识一个包含 8 个输入字节的数组，从 DP 从站子模块 3 的偏移 3 处开始

## 输出

- CP 5611.q4.3,w  
标识位于 DP 从站子模块 4 中地址 3 的输出字
- CP 5611.q3.2,dw  
标识位于 DP 从站子模块 3 中地址 2 的输出字
- CP 5611,q0.3,x7  
标识 DP 从站的字节 3 中输出位 7
- CP 5614.q1.0,w,4  
标识 DP 从站子模块 1 中包含 4 个输出字的数组

### 2.5.8.4 DP 从站的状态

#### DP 从站的状态语法

命名空间 URI: DP: (命名空间索引: 3)

```
<DPslave>.<statusvariables>
```

#### 说明

##### **<DPslave>**

协议特定的连接名称。连接名称在组态中指定。在 DP 协议中，连接名称是通信模块的已组态名称。

##### **state**

指示 DP 从站的状态。

##### **<statusvariables>**

状态变量	说明		
state.state	查询 DP 从站的状态。 UA 类型“MultistateDiscreteType”的数据变量，只读		
	0	NO_DATA_EXCHANGE	DP 主站尚未组态和分配 DP 从站的参数。
	1	DATA_EXCHANGE	DP 从站与 DP 主站进行周期性数据交换。

### 2.5.8.5 DP 从站的模式

#### DP 从站的模式语法

<DPslave>.mode

#### 说明

##### <DPslave>

协议特定的连接名称。连接名称在组态中指定。在 DP 协议中，连接名称是通信模块的已组态名称。

##### mode

	说明	
mode	查询或设置 DP 从站的模式。 UA 类型“MultistateDiscreteType”的数据变量，读写	
	0 OFFLINE 1 ONLINE	总线上的 DP 从站无响应。 DP 从站准备从 DP 主站接收参数分配和组态帧，成功完成参数分配和组态后变为“DATA_EXCHANGE”状态。

### 2.5.8.6 DP 从站特定信息变量

DP 从站有几个用于诊断的预定义信息变量。

### 2.5.8.7 DP 从站特定信息变量的语法

#### DP 从站上的 DP 从站信息变量的语法

命名空间 URI: DP: (命名空间索引: 3)

<DPslave>.<informationvariables>

#### 说明

##### <DPslave>

协议特定的连接名称。连接名称在组态中指定。在 DP 协议中，连接名称是通信模块的已组态名称。

## &lt;informationvariables&gt;

信息变量	说明				
type	查询从站类型 UA 类型“MultistateDiscreteType”的数据变量，只读				
	0	NORM	标准 DP 从站		
	OPC 服务器运行的 DP 从站始终为标准 DP 从站。				
dpv	查询 DP 从站的 DP 协议版本。 UA 类型“字节”的数据变量，只读				
	0	DP-V0			
	1	DP-V1			
	2	DP-V2			
	始终为 0，OPC 服务器运行的 DP 从站仅支持 DP-V0。				
parcfgdata	DP 从站的组态数据。组态数据包含 IO 子模块和 DP 从站的组态信息，OPC UA 服务器使用组态数据设置属于 DP 从站的 IO 模块对象。 UA 类型“ByteString”的数据变量，只读				
paruserdata	DP 从站的用户参数分配数据。通常不会为此返回数据。 UA 类型“ByteString”的数据变量，只读				
hardware	硬件。 UA 类型“MultistateDiscreteType”的数据变量，只读				
	0	SOFTNET			
	1	CP 5613 (电气 PROFIBUS 端口)			
	2	CP 5613 FO (光学 PROFIBUS 端口)			
	3	CP 5614 (电气 PROFIBUS 端口)			
	4	CP 5614 FO (光学 PROFIBUS 端口)			
	5	接有外部电源的 CP 5614 FO (光学 PROFIBUS 端口)			
	6	接有外部电源的 CP 5613 FO (光学 PROFIBUS 端口)			
	7	CP 5613 A2 (电气 PROFIBUS 端口)			
	8	CP 5614 A2 (电气 PROFIBUS 端口)			
	9	CP 5623 (电气 PROFIBUS 端口)			

信息变量	说明	
	10	CP 5624 (电气 PROFIBUS 端口)
	11	CP 5613 A3 (电气 PROFIBUS 端口)
	12	CP 5614 A3 (电气 PROFIBUS 端口)
hardwareversion	硬件版本。 UA 类型“字节”的数据变量，只读	
firmwareversion	固件版本。 UA 类型“UInt16”的数据变量，只读	
标识号	认证标识号。 UA 类型“UInt16”的数据变量，只读	

**示例：**

CP 5611.hardware

**DP 从站上的 DP 从站 IO 信息变量的语法**

有 4 个选项：

- <DPslave>.q.<slaveproperties>
- <DPslave>.i.<slaveproperties>
- <DPslave>.q<number>.<slavemoduleproperties>
- <DPslave>.i<number>.<slavemoduleproperties>

**说明****<DPslave>**

协议特定的连接名称。连接名称在组态中指定。在 DP 协议中，连接名称是通信模块的已组态名称。

**q**

输出的标识符。输出为只读。

**i**

输入的标识符。输入可读取和写入。

**<number>**

包含输入或输出区域的子模块的编号。

## &lt;slaveproperties&gt;

IO 信息变量	说明	
length	DP 从站组态数据中的模块长度（字节）。 UInt16 类型的属性，只读	
module type	DP 从站组态数据中的模块类型。 UA 类型“MultistateDiscreteType”的数据变量，只读。	
	0	未知
	1	I (输入)
	2	Q (输出)
	3	IQ (输入/输出)
	只有 DP 从站模块才会出现值“3”。 这种情况下，存在具有相同“moduleid”的输入和输出子模块。	
slaveid	相应 DP 从站的 PROFIBUS 地址 “字节”类型的属性，只读	

## &lt;slavemodeleproperties&gt;

IO 信息变量	说明	
length	DP 从站组态数据中的模块长度（字节）。 UInt16 类型的属性，只读	
module type	DP 从站组态数据中的模块类型。 UA 类型“MultistateDiscreteType”的数据变量，只读。	
	0	未知
	1	I (输入)
	2	Q (输出)
	3	IQ (输入/输出)
	只有 DP 从站模块才会出现值“3”。 这种情况下，存在具有相同“moduleid”的输入和输出子模块。	
slaveid	相应 DP 从站的 PROFIBUS 地址 “字节”类型的属性，只读	

IO 信息变量	说明	
moduleid	DP 从站组态数据中的模块地址。 “字节”类型的属性，只读	
slot	此子模块的插槽或模块标识符。 “字节”类型的属性，只读	
dataunitlength	DP 从站组态数据中的子模块数据单位长度。 UA 类型“MultistateDiscreteType”的数据变量，只读	
	0	未知
	1	字节
	2	字
consistence	DP 从站组态数据中的子模块所有数据单位的一致性。 “布尔”类型的属性，只读	

示例：

CP 5614\_s.q.length

CP 5614\_s.q.modulename

CP 5614\_s.q0.length

## 2.5.9 DP OPC UA 模板数据变量

### 2.5.9.1 DP OPC UA 模板数据变量

通过 DP OPC UA

协议的过程变量，您可拥有灵活的设置选项以所需的数据格式获取设备的过程数据。

不过，在完全可浏览的命名空间中，不能将各种寻址选项放在一起。

即使单字节长度的数据块也有大约 40 个不同的数据格式选项 - 从 Byte 和 SByte 开始，由这些数据类型的一个元素组成的数组、单个位、最多 8 个数组元素的位数组，每个元素均起始于不同的位偏移。

因此，OPC UA 服务器支持 DP 命名空间中具有模板数据变量的用户。在典型的 OPC UA 客户端文本输入框中，只需更改几个字符，就可将这些模板转换为有效的 ItemID。

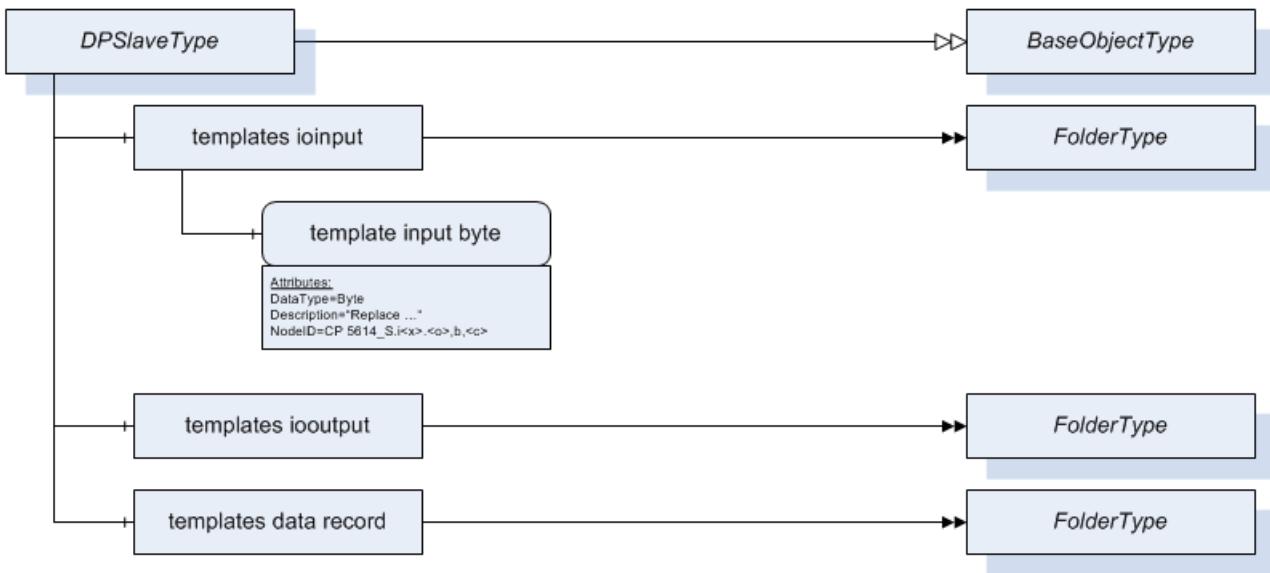


图 2-12 DP OPC UA 中的模板数据变量

### 说明

可在“通信设置”组态程序中启用和禁用 DP OPC UA 模版数据变量的可用性：“OPC 协议选择”(OPC protocol selection) > 单击“DP”旁的箭头符号。

### 浏览层级结构中的模板数据变量

在命名空间表示中相应的文件夹旁将模板数据变量进行排序以便需要时便于使用。

### 模板数据变量的语法

有两个选项：

- <DPmaster>.<slave>.i<x>.<o>,<DPtypetemplate>,<c>
- <DPmaster>.<slave>.q<x>.<o>,<DPtypetemplate>,<c>

### 说明

#### <DPmaster>

协议特定的连接名称。连接名称在组态中指定。在 DP 协议中，连接名称是通信模块的已组态名称。

**<slave>**

从站符号名称，用作访问 DP 从站的标识符。可在 STEP 7/NCM PC 中分配从站符号名称。

**说明**

如果在 STEP 7/NCM PC 中组态 V8.2 以下版本的 OPC 服务器且未分配从站的符号名称，则会采用标识符“slv”后接组态的从站地址形式作为从站的符号名称。

如果在 STEP 7/NCM PC 中组态 V8.2 及以上版本的 OPC 服务器且未分配从站的符号名称，则会将从站的标识符用作从站名称。在 STEP 7/NCM PC 中执行一致性检查，确保使用唯一从站名称且从站名称不为空。

!

输入的标识符。输入可读取和写入。

**q**

输出的标识符。输出为只读。

**<x>**

包含输入或输出区域的子模块的编号占位符。

**<o>**

DP 从站的地址空间内字节偏移量占位符。

**<DPtypetemplate>**

将 DP 模版数据类型转换为 OPC UA 服务器上相应的 OPC UA 数据类型。

下表列出了类型标识符以及可用来表示变量值的对应 OPC 数据类型。

数据类型	OPC UA 数据类型	注意
x<bit>	布尔	位偏移量（0 到 7）的占位符
b	字节	字节（无符号）的占位符
w	UInt16	字（无符号）的占位符
dw	UInt32	双字（无符号）的占位符
lw	UInt64	长字（无符号）的占位符
c	SByte	字节（有符号）的占位符
i	Int16	字（有符号）的占位符
di	Int32	双字（有符号）的占位符
li	Int64	长字（有符号）的占位符

数据类型	OPC UA 数据类型	注意
r	Float	浮点 (4 字节) 的占位符
lr	Double	浮点 (8 字节) 的占位符
s<sl>	String	字符串长度的占位符

&lt;c&gt;

元素数量的占位符。

示例：

NodeId	BrowseName	说明
CP 5624.slv<v>.i<x>.<o>,x<bit>,<c>	模板输入位数组	<v> DP 从站的 PROFIBUS 地址 <x> 子模块的地址 <o> 子模块内的偏移量 <bit> 位偏移量 (0 到 7) <c> 数组大小
CP 5624.slv<v>.i<x>.<o>,b,<c>	模板输入字节数组	<v> DP 从站的 PROFIBUS 地址 <x> 子模块的地址 <o> 子模块内的偏移量 <c> 数组大小
CP 5624.slv<v>.i<x>.<o>,w,<c>	模板输入字数组	<v> DP 从站的 PROFIBUS 地址 <x> 子模块的地址 <o> 子模块内的偏移量 <c> 数组大小
CP 5624.slv<v>.i<x>.<o>,s<sl>,<c>	模板输入字符串数组	<v> DP 从站的 PROFIBUS 地址 <x> 子模块的地址 <o> 子模块内的偏移量 <sl> 字符串长度 <c> 数组大小

Nodeld	BrowseName	说明
CP 5624.slv<v>.q<x>.<o>,x<bit>,<c>	模板输出位数组	<v> DP 从站的 PROFIBUS 地址 <x> 子模块的地址 <o> 子模块内的偏移量 <bit> 位偏移量 (0 到 7) <c> 数组大小
CP 5624.slv<v>.dr<dr>,<l>.<o>,b	模板数据记录字节	<v> DP 从站的 PROFIBUS 地址 <x> 子模块的地址 <dr> 数据记录号 <l> 文件长度 <o> 文件内的偏移量
CP 5624_S.i<x>.<o>,b	模板输入字节	<x> 子模块的地址 <o> 子模块内的偏移量

## 2.6 S7 通信

针对 PROFIBUS 和工业以太网的 S7 协议用于 SIMATIC S7 可编程控制器内部系统组件之间的通信以及 SIMATIC S7 系统组件与编程设备或 PC 之间的通信。

### S7 与 OPC 通信的属性

SIMATIC NET 的 OPC 服务器具有以下特性:

- 变量服务  
访问和监视 S7 变量。
- 面向缓冲区的服务  
由程序控制较大数据块传送。
- 服务器功能  
PC 可用作数据块服务器。
- 块管理服务  
与 S7 之间传送可装载数据区域。

- S7 密码功能  
设置密码以访问受保护块。
- 事件  
作为 OPC 报警和事件处理 S7 消息（S7 报警）。

## 2.6.1 适用于 S7 协议的功能强大的 SIMATIC NET OPC 服务器

### 简介

以下部分介绍一种满足更高性能要求的 S7 协议组态形式。要使用这种形式，所有底层 S7 协议库和作为进程内服务器的 COM 服务器都在进程外 OPC 服务器上进行加载。

#### 协议在 OPC

服务器的进程中进行处理，这就避免了在进程与多协议模式之间进行切换所需的额外执行时间。不过，OPC 客户端与 OPC 服务器之间的进程切换仍然必不可少。

## 组态

通过在“通信设置”组态程序中选择“S7”协议作为唯一协议，隐式启用此高速版本（如果选择了其它协议或 OPC UA 接口，则会失去下面介绍的性能优点）：



图 2-13 “通信设置”组态程序中用于选择 S7 协议的窗口

也可选择“符号”(Symbols)。

### 说明

使用 STEP 7 或符号编辑器创建符号时，允许使用以下字符：A-Z、a-z、0-9、\_、-、^、!、#、\$、%、&、'、/、(、)、<、>、=、?、~、+、\*、'、'、:、!、@、[、]、{、}、"。使用 STEP 7 创建符号时，还应记住，如果符号文件中的符号同时具有 `<symbolname>` 和 `<symbolname>[<索引>]` 形式，则解析数组时会出现问题。

## 优点/缺点

使用功能强大的 S7 OPC 服务器确实有缺点，就是只能采用 S7 单协议模式。

但另一方面，则具有以下优点：

- 与多协议模式相比，性能更高。
- 组态简单。
- 多个客户端可同时使用服务器。
- OPC 服务器的稳定性不取决于客户端。

## 注意

### 说明

如果在连接属性中（通过组态）将 OPC 服务器组态为被动连接伙伴，确保 OPC 服务器组态的 S7 连接数不超过可以同时运行的 S7 连接数。

背景：

如果 PC 站中组态的连接数多于同时允许的值，OPC 服务器无法为所有连接处理请求连接，也无法建立每条 S7 连接。

### 说明

可以在“按需”建立的 SIMATIC STEP 7/NCM PC 中组态 S7 连接。

只能根据需要建立 OPC 功能通信；也就是说建立连接可能需要更长的时间，且 OPC 可能会在启动阶段发出错误信号。

## 2.6.2 协议 ID

S7 协议的协议 ID 是“S7”。

## 2.6.3 连接名称

### STEP 7 连接

连接名称是在 STEP 7 中组态的名称，用于标识连接。在 STEP 7 中，此名称称为“本地 ID”。在 OPC 服务器内，本地 ID 是唯一的。

OPC 服务器支持以下连接类型:

- S7 连接
- 容错 S7 连接

#### 说明

与 2.2 及更早版本的兼容性 - 请注意以下内容:

在早先版本中, 连接信息由关于 S7 连接、VFD 和通信处理器的信息组成。

这些元素仍被变量的 **ItemID** 所接受, 但已不存在于命名空间。如果在某一组态中  
OPC 服务器使用多个 VFD, 则在 STEP 7 中只能出于兼容性原因组态这些 VFD。

### S7 连接名称允许使用哪些字符?

对于 <connectionname>, 可使用数字“0-9”、大小写字母字符“A-z”和特殊字符“\_-+()”。  
连接名称长度可达 24 个字符。名称不区分大小写。

不允许使用其它可打印的字符。

连接名称“SYSTEM”和“@LOCALSERVER”已保留, 不得使用。

### 连接名称示例

典型示例包括:

- S7\_connection\_1
- S7 OPC connection

### 连接对象

有下列 S7 连接对象:

- S7 生产连接  
S7 连接用于可编程控制器之间的数据交换, 通常在 STEP 7 中进行组态。
- DEMO 连接  
仅用于测试。
- @LOCALSERVER 连接  
使 S7 服务器功能的本地 S7 数据块可用。

## 2.6.4 变量服务

利用变量服务，可以访问并监视可编程控制器中的 S7 变量。对 S7 变量进行符号寻址。访问类型取决于 S7 工具的表示法。

### 可编程控制器上的对象

OPC 服务器支持以下对象：

- 数据块
- 背景数据块
- 输入
- 输出
- 外设输入
- 外设输出
- 存储器位
- 定时器
- 计数器

一些 S7 可编程控制器并不支持所有对象类型。

### 2.6.4.1 S7 变量服务的过程变量语法

#### 语法

有三个选项：

```
S7: [<connectionname>] DB<no>, {<type>}<address>{, <quantity>}
```

```
S7: [<connectionname>] DI<no>, {<type>}<address>{, <quantity>}
```

```
S7: [<connectionname>]<object>{<type>}<address>{, <quantity>}
```

#### 说明

##### **S7**

用于访问过程变量的 S7 协议。

##### **<connectionname>**

协议特定的连接名称。连接名称在组态中指定。

**DB**

数据块。 数据块中 S7 变量的标识符。

**DI**

背景数据块。 背景数据块中 S7 变量的标识符。

**<no>**

数据块或背景数据块的编号。

**<object>**

在 S7 PLC 上指定块/区类型。

可能值如下：

I	输入
Q	输出
PI	外设输入
PQ	外设输出
M	位存储器
T、TBCD、TDA	定时器
C	计数器

**S7 定时器变量（类型 T）的时间基准和值范围：**

S7 的定时器类型 (T) OPC 过程变量值范围以十进制编码（单位为 ms）。时间基准（用于写入）取自下表所示值范围：

可能值	时间基准	注释
1,000,000 ms 到 9,990,000 ms	10 s	值必须是 10,000 ms 的倍数。
100,000 ms 到 999,000 ms	1 s	值必须是 1,000 ms 的倍数。
10,000 ms 到 99,900 ms	100 ms	值必须是 100 ms 的倍数。
10 ms 到 9,990 ms	10 ms	值必须是 10 ms 的倍数。 不允许更小的时间基准。 允许 0 ms，但它不起作用。

位数决定时间基准，前导数字（1 到 999）乘以时间基准决定时间。

示例：

值 90,000 ms 的时间基准为 100 ms。

值 123,000 ms 的时间基准为 1 s。

值 150 ms 的时间基准为 10 ms。

S7 定时器变量（类型 T）的 OPC 数据类型是字（有符号，VT\_UI4）。

#### S7 定时器变量（类型 TBCD）的时间基准和值范围：

S7 的 OPC 过程变量（类型 TBCD）值范围采用 BCD

编码。时间基准（用于写入）取自下表所示值范围：

位号	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
含义（符号）	0	0	x	x	t	t	t	t	t	t	t	t	t	t	t	t
<b>说明：</b>																
“0”的含义	不相关															
“x”的含义	指定时间基准															
	<b>位 13 和 12</b>										<b>时间基准（秒）</b>					
	00										0.01					
	01										0.1					
	10										1					
	11										10					
“t”的含义	BCD 编码的时间值（0 至 999）															

示例：

位号	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
值	0	0	0	1	0	0	0	0	0	1	1	1	0	1	0	1

位 0-11 指定数字 075。位 12 和 13 指定时间基准为 0.1。

$75 * 0.1 = 0.75$  秒

S7 定时器变量（类型 TBCD）的 OPC 数据类型是字（无符号，VT\_UI2）。

#### S7 定时器变量（类型 TDA）的时间基准和值范围：

尽管使用 T

类型的简单定时器变量易于操作定时器，但无法设置所有可能的时间基准与值范围的组合，因为值范围可能重叠。

这种情况下可使用 TDA（十进制数组）类型的 S7 定时器变量。

#### **<object>: TDA**

数据类型：由两个字组成的字段 {以 ms 为单位的时间基准 VT\_UI2 / 时间值 VT\_UI2}。

值范围：

以 ms 10, 100, 1000, 10000.

为单位的时间基准

:

时间值： 0 到 999，允许 0 但它不起作用。

时间范围： 10 ms: 0 到 9990 ms; 100 ms: 0 到 99900 ms; 1000 ms: 0 到 999000 ms;  
10,000 ms: 0 到 9990000 ms

TDA 对象无法输入 <quantity>。

#### **示例：**

将值 {100|50} 写入定时器 TDA3 以值  $50*100\text{ms}=5000\text{ ms}$  初始化定时器 3，这以 100 ms 为步长递减 50 次。

此设置不可用于 T 类型。

#### **S7 计数器变量（类型 C）的值范围：**

S7 的 OPC 过程变量（计数器 (C) 类型）值范围是 0 到 999（十进制编码）。

S7 计数器变量（类型 C）的 OPC 数据类型是字（无符号，VT\_UI2）。

#### **<type>**

S7 数据类型。

S7 数据类型转换成 OPC 服务器中的相应 OLE 数据类型。所有指定的 OLE 数据类型都可通过 OPC 的自动化接口进行读取。但是，某些开发工具（如 Visual Basic）只提供有限数量的数据类型。下表列出了可用来表示变量值的相应 Visual Basic 类型。

对于 *T*、*TDA* 和 *C* 类型的对象，无法指定类型。

S7 数据类型	说明	OLE 数据类型	Visual Basic 类型
X	位（布尔）。 指定位号（0 到 7）。	VT_BOOL	布尔
B	字节（无符号）	VT_UI1	字节
W	字（无符号）	VT_UI2	Long
D	双字（无符号）	VT_UI4	Double
CHAR	字节（有符号）	VT_I1	Integer
INT	字（有符号）	VT_I2	Integer
DINT	双字（有符号）	VT_I4	Long
REAL	浮点数	VT_R4	Single
String	字符串。 指定字符串长度。字符串必须以 S7 上的有效值初始化。	VT_BSTR	String
DT	日期和时间，8 字节 BCD 格式 (S7 CPU DATE-AND-TIME)	VT_DATE	Date
DATE	日期和时间，8 字节 时间始终为 00:00:00，值范围始于 01.01.1990。 CPU 数据类型 DATE 映射（无符号，16 位）。	VT_DATE	Date
TIME	时间值（有符号），IEC 格式，单位为 ms	VT_I4	Long
TOD	日时钟（无符号） 0 至 86399999 ms	VT_UI4	Double
S5TIMEBCD	映射 CPU 数据类型 S5TIME（无符号，16 位），值范围限制为 0 至 9990000 ms。	VT_UI2	Long

**S7 数据类型 S5TIMEBCD 的时间基准和值范围:**

S5TIMEBCD 数据类型的时间变量值范围采用 BCD 编码。值范围请参见下表：

位号	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
含义 (符号)	0	0	x	x	t	t	t	t	t	t	t	t	t	t	t	t
说明：																
“0”的含义	不相关															
“x”的含义	指定时间基准															
	位 13 和 12										时间基准 (秒)					
	00										0.01					
	01										0.1					
	10										1					
	11										10					
“t”的含义	BCD 编码的时间值 (0 至 999)															

**示例：**

位号	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
值	0	0	0	1	0	0	0	0	0	1	1	1	0	1	0	1

位 0-11 指定数字 075。位 12 和 13 指定时间基准为 0.1。

$$75 * 0.1 = 0.75 \text{ 秒}$$

时间变量（数据类型 S5TIMEBCD）的 OPC 数据类型是字（无符号，VT\_UI2）。写入时，相应限制值范围。

**<address>**

要寻址的第一个变量的地址。可能值如下：

- 字节偏移量
- 字节偏移量.位 (只适用于数据类型 X)  
位地址的值范围之后可为 0 到 7。
- 字节偏移量字符串长度 (只适用于字符串数据类型，字符串长度为 1 字节到 254 字节)

字节偏移的值范围为 0 到 65534。根据设备和类型，实际可用的地址值可能更低。

#### **<quantity>**

要寻址的某一类型变量的数目，从 *address* 参数中指定的偏移量开始。

值范围取决于组态。

对于数据类型 *X*，写访问的数量只能输入 8 的倍数。之后位地址必须为零。

对于数据类型 *X*，不限制数量的输入。

#### **示例：**

S7:[S7-OPC-1]DB1,X10.0,64, access rights RW

S7:[S7-OPC-1]DB1,X10.3,17, access rights R

### 2.6.4.2 S7 变量服务的过程变量的示例

下面几个示例说明了变量服务的变量名称的语法。

#### 数据块参数

*S7:[S7-connection-1]DB5,B12*

DB5,B12

标识数据块 5 中的数据字节 12。

#### 实例数据块参数

*S7:[S7-connection-1]DI5,W10,9*

DI5,W10,9

标识起始于实例数据块 5 中的字节地址 10 的 9 个数据字。

#### 对象参数

*S7:[S7connection1]IB0*

IB0

标识输入字节 0。

### 2.6.4.3 具有最佳结构的数据项的示例

OPC 服务器为 S7 变量服务提供了优化算法：

- 同时读取或监视的 OPC 数据项在伙伴设备的命名空间中应连续排列。  
也会处理相关部分之间的小间隙，但会导致数据吞吐量减少。
- 同时写入的 OPC 数据项在命名空间中必须连续排列。  
为获得最佳写入访问，相关部分之间不能有空隙。

如果 OPC 客户端无需使用过程变量的单个 OPC  
数据项，则可以直接使用数组访问相关数据，然后将所读取的数组元素分配给各个过程变  
量。

#### 读取访问的数据块组织结构

DB10,W10

DB10,B12

DB10,B13

DB10,DW14

DB10,W20

执行效果如同：通过通信系统对 DB10,B10,12 进行读取访问。

尽管字节 18 和字节 19 之间有间隙，但仍可将其转换为单个读取作业。

不必放弃数据读取。仅通过网络传输一个请求，而不再传输五个单个变量请求。

#### 写入访问的数据块组织结构

DB10,W10

DB10,B12

DB10,B13

DB10,DW14

DB10,W20

执行效果如同：通过通信系统进行两次写入操作，即写入 DB10,B10,8  
数组和写入单个变量 DB10,W20。

## 2.6.5 面向缓冲区的服务

利用面向缓冲区的服务，可通过程序控制较大数据块的传送。

通过以下变量执行数据传送：

- 发送数据块的变量
- 接收数据块的变量

无论 PDU 的大小如何，可传送的最多有效负载数据为 65534 个字节。

数据的分段由函数本身来处理。

### 说明

面向缓冲区的服务只能在双端点连接中使用。必须将所创建的连接组态下载到 S7 可编程控制器或相关 PC。

## 面向缓冲区的服务的使用示例

下图说明了 S7400 如何通过 S7 OPC 服务器向 PC 站发送数据包。

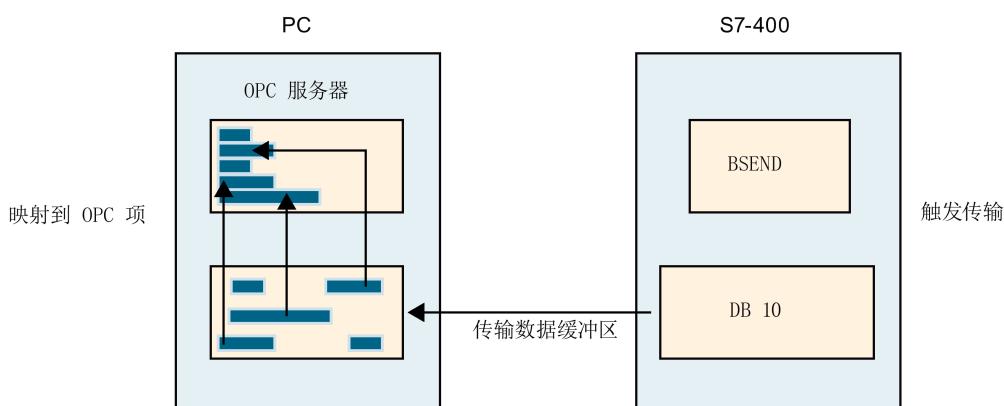


图 2-14 通过 S7 OPC 服务器从 S7-400 设备向 PC 站发送数据包

为允许 OPC 服务器接收数据，将一个或多个 *BRCV*类型的 OPC 数据项添加到 PC 上的激活组。

在 S7400 上，用户程序启动 *BSEND* 函数块。*BSEND* 开始将整个数据区域作为一个缓冲区传送到 PC。

在 PC 上，将所接收的数据传送到 OPC 服务器。现在，OPC 服务器即会将数据块的子区域映射到对应的 OPC 数据项。如果监视这些 OPC 数据项，则 OPC 服务器会在值发生更改时向 OPC 客户端发送回调。

### 2.6.5.1 面向缓冲区的服务的过程变量语法

#### 语法

有两个选项：

```
S7: [<connectionname>] BRCV, <RID>{, {<type>}<address>{, <quantity>}}  
S7: [<connectionname>] BSEND<length>, <RID>{, {<type>}<address>  
    {, <quantity>}}
```

#### 说明

##### S7

用于访问过程变量的 S7 协议。

##### <connectionname>

协议特定的连接名称。连接名称在组态中指定。

##### BRCV

BRCV 包含从伙伴接收的最后一个数据块。接收数据的内容和长度由发送伙伴指定。  
该变量为只读变量。

使用这些变量进行监视。OPC 服务器收到数据块后，随即会就此事向 OPC 客户端应用程序发出信号。

在连接超时（在组态中为特定连接指定的时间）或接收数据块时，返回用于读取这些变量的显式作业。

OLE 数据类型	Visual Basic 类型
VT_ARRAY   VT_UI1	Byte()

**BSEND**

**BSEND** 包含要发送到伙伴设备的数据块。

只有在写入变量后，才能将数据块发送至伙伴。

读取获得最后一个成功传送的数据块的内容。

OLE 数据类型	Visual Basic 类型
VT_ARRAY   VT_UI1	Byte()

**<length>**

要发送的数据块的长度（字节）。

**<RID>**

寻址参数的 ID。此项针对块对 (*BSEND/BRCV*) 进行设置，并在连接中唯一定义。

您可以通过一个连接发送若干个 *BSEND* 数据块或接收若干个 *BRCV*

数据块，但每个数据块必须具有不同的 ID。可在其它连接上使用相同的 ID。

*BSEND/BRCV* 的取值范围为 0 至 4294967295。

**<type>**

S7 数据类型。

S7 数据类型转换成 OPC 服务器中的相应 OLE 数据类型。

所有指定的 OLE 数据类型都可通过 OPC 的自动化接口进行读取。

但是，某些开发工具（例如 Visual Basic）只提供有限数量的数据类型。

因此，下表列出了可用于表示变量值的相应 Visual Basic 类型。

格式 标识符	说明	OLE 数据类型	Visual Basic 类型
X	位（布尔）。 必须指定位编号。	VT_BOOL	布尔
BYTE 或 B	字节（无符号）	VT_UI1	字节
WORD 或 W	字（无符号）	VT_UI2	长整型
DWORD 或 D	双字（无符号）	VT_UI4	双精度
CHAR	字节（有符号）	VT_I1	整型
INT	字（有符号）	VT_I2	整型
DINT	双字（有符号）	VT_I4	长整型
REAL	浮点数	VT_R4	单精度

格式 标识符	说明	OLE 数据类型	Visual Basic 类型
字符串	字符串。 需要指定字符串长度	VT_BSTR	字符串
DT	日期和时间， 8 个字节的 BCD 格式	VT_DATE	Date
TIME	时间值（有符号）， IE C 格式， 单位为 ms	VT_I4	长整型
TOD	日时钟（无符号） 0 至 86399999 ms	VT_UI4	双精度

**<address>**

要寻址的第一个变量的地址。

可能值如下：

字节编号

bytenumber.bit (仅适用于 X 格式)

字节编号的取值范围为 0 至

65534。根据所使用设备及类型的不同，地址的实际可用值可能会较低。

**<quantity>**

要寻址的某一类型变量的数目，从 *address* 参数中指定的地址开始（取值范围为 0 至 65535）。

**说明**

可读取 (BRCV) 和写入 (BSEND) 单个位 (X 格式)。

也可以在数据块中读取和写入各个位组成的数组，而无需考虑起始地址和数组长度。

示例： *S7:[S7-OPC-1]BRCV,1,X10.4,78*

### 说明

请注意下列各项：

- 通过指定可选信息类型、地址和数量，可以定义对数据块子区域的结构化访问。
- 长度不同或具有不同 RID 的发送数据或接收数据的变量具有独立的存储区。
- 为独立存储区创建数据项时，将分配发送数据缓冲区并将其初始化为零。对 BSEND 数据项执行的写入操作将写入内部写入缓冲区，然后进行传送。
- 数据块进行非周期性传送。可以对相同的数据同时执行多个网络作业。

始终传送完整的发送数据块。

此规则也适用于子元素访问或几个客户端同时写入此数据项的情况。

几个客户端同时写入同一发送数据块或发送数据块的子区域会造成不一致。

因此我们建议

- 您始终读取或写入完整的数据块，
- 您在 NCM S7 中将“并行网络作业的最大数目”(Maximum number of parallel network jobs) 设置为“一”(one)，但这会降低传输性能。

### 2.6.5.2 面向缓冲区的服务的过程变量示例

下面几个示例说明了面向缓冲区的服务的变量名称的语法。

#### 接收整个缓冲区中的数据块

*S7:[S7-OPC-1]BRCV,1*

BRCV,1

在 RID 1 的接收缓冲区中接收数据块。将完整的缓冲区映射到字节数组。

#### 交替访问所接收的数据块

*S7:[S7-OPC-1]BRCV,1,W2,4*

BRCV,1,W2,4

映射所接收数据块的内容，起始地址为 4 字数组上的偏移 2。总共有 8 个字节的相关数据块。

## 传送双字

*S7:[S7-OPC-1]BSEND16,1,D2*

BSEND16,1,D2

在数据块中寻址双字，长度为 16，RID 1 起始于偏移

2。如果写入命令应用于变量，则在数据块中指定位置输入写入值，然后发送数据块。

## 传送浮点数

*S7:[S7-OPC-1]BSEND32,1,REAL20,2*

BSEND32,1,REAL20,2

在数据块中寻址浮点数数组，长度为 32，RID 1 起始于偏移

20。如果写入命令应用于变量，则在数据块中指定位置输入写入值，然后发送数据块。

## 2.6.6 S7 特定的信息变量

提供 S7 特定的变量，可通过这些变量获得有关 S7 通信和已建立连接的信息。

可获取以下信息：

- 虚拟现场设备 (VFD) 的属性
- S7 连接的状态
- 虚拟设备的状态
- 用于诊断的 S7 连接的已组态参数和延迟时间参数

### 2.6.6.1 S7 特定的信息变量的语法

#### S7 特定的信息变量

语法：

*S7: [<connectionname>]<informationvariable>*

#### 说明

##### S7

用于访问过程变量的 S7 协议。

**<connectionname>**

协议特定的连接名称。连接名称在组态中指定。

**<informationparameter>**

可能值如下：

**&identify()**

通信伙伴的供应商属性。

返回值（作为字符串字段的元素）：

- 供应商
- 型号
- 版本

OLE 数据类型	Visual Basic 类型
VT_ARRAY of VT_BSTR	String()

**&vfdstate()**

虚拟现场设备的状态。

数据类型： *VT\_ARRAY*, 由 *VT\_VARIANT* 组成，包括以下组成部分：

逻辑状态

所支持的服务

返回值：

- *S7\_STATE\_CHANGES\_ALLOWED*

允许所有服务

OLE 数据类型	Visual Basic 类型
VT_BSTR	字符串

物理状态

实际设备的可操作性

返回值:

- *S7\_OPERATIONAL*  
实际设备为可操作设备
- *S7 NEEDS COMMISSIONING*  
本地设置完成后才能使用实际设备

OLE 数据类型	Visual Basic 类型
VT_BSTR	字符串

有关本地 VFD 状态的详细信息。

以八位位组串形式返回状态。有关返回值含义的详细信息，请参见伙伴设备的文档。

OLE 数据类型	Visual Basic 类型
VT_ARRAY of VT_BSTR	String()

### **&statepath()**

与伙伴设备的通信连接状态。

返回值:

- *DOWN*  
未建立连接
- *UP*  
已建立连接
- *RECOVERY*  
正在建立连接
- *ESTABLISH*  
(保留供将来扩展使用)

OLE 数据类型	Visual Basic 类型
VT_BSTR	字符串

**&statepathval()**

与伙伴设备的通信连接状态。

返回值：

- 1

未建立连接

- 2

已建立连接

- 3

正在建立连接

- 4

(保留供将来扩展使用)

OLE 数据类型	Visual Basic 类型
VT_UI1	字节

## S7 特定的信息变量和返回值的示例

下面几个示例说明了 S7 特定的信息变量名称的语法。

### 有关虚拟设备的供应商属性的信息

*S7:[S7-OPC-1]&identify()*

**&identify()**

例如，可返回以下值：

- 供应商： *SIEMENS AG*
- 虚拟设备的型号： *6ES7413-1AE0-0AB0*
- 版本： *V1.0*

### 设备的状态

*S7:[S7-OPC-1]&vfdstate()*

**&vfdstate()**

例如，可返回以下值：

- 逻辑状态： *S7\_STATE\_CHANGES\_ALLOWED*
- 允许所有服务。
- 物理状态： *S7\_OPERATIONAL*
- 实际设备为可操作设备。
- 详细信息： *02.00.00*
- 有关本地 VFD 状态的详细信息。

### 字符串形式的通信连接的状态

*S7:[S7-OPC-1]&statepath()*

**&statepath()**

例如，可返回以下值：

- 连接状态： *RECOVERY*
- 当前正在建立连接。

### 数字形式的通信连接的状态

*S7:[S7-OPC-1]&statepathval()*

**&statepathval()**

例如，可返回以下值：

- 连接状态： *2*
- 已建立连接。

### 2.6.6.2 S7 特定的诊断变量

以下诊断变量可用于检查 S7 连接的组态参数和当前运行期间参数。在 OPC Scout V10 中，有单独的诊断视图；这些变量也显示在“组态”(Configuration) 文件夹中相关 S7 连接下的命名空间中。

## 语法

```
S7: [<connectionname>]<diagnosticsvariable>
```

诊断变量	说明
&vfd()	连接所分配到的 OPC 服务器的名称。对于在 NCM 中组态的连接，此项通常包含文本“OPC 服务器”。 数据类型 VT_BSTR，只读。
&cp()	连接所分配到的接口参数分配的名称。 数据类型 VT_BSTR，只读。
&remoteaddress()	连接伙伴的地址。 数据类型 VT_BSTR，只读。 连接伙伴的地址是一个数据缓冲区，其数据长度取决于连接类型。 为使其更便于管理，以供用户进行评估，数据缓冲区以字符串格式显示。 <b>PROFIBUS 地址</b> 格式：“ddd”（1-3 个十进制数） <b>IP 地址 (ISO-on-TCP)</b> 格式：“ddd.ddd.ddd.ddd”（每组 1-3 个十进制数） <b>MAC 地址 (ISO)</b> 格式：“xx-xx-xx-xx-xx-xx”（每组 2 个十六进制数）
&localsap()	连接的本地 SAP。 数据类型 VT_BSTR，只读。 连接伙伴的本地 SAP 是一个数据缓冲区，其数据长度取决于连接类型。 为使其更便于管理，以供用户进行评估，数据缓冲区以字符串格式显示。 格式：“xx.xx”（每组 2 个十六进制数）
&remotesap()	连接的远程 SAP。 数据类型 VT_BSTR，只读。 连接伙伴的远程 SAP 是一个数据缓冲区，其数据长度取决于连接类型。 为使其更便于管理，以供用户进行评估，数据缓冲区以字符串格式显示。 格式：“xx.xx”（每组 2 个十六进制数）
&connect()	连接建立的类型。 数据类型 VT_UI4，只读。 0 被动，连接永久保持为已建立状态。

诊断变量	说明
	<p>1 主动，仅根据需要建立连接，如果在等待时间过后还不使用，则终止连接。</p> <p>2 主动，连接永久保持为已建立状态。</p>
&abortconnectionafter()	<p>自动终止连接。</p> <p>自动终止连接的延迟时间：如果在延迟时间期间没有其它的变量访问操作，则延迟时间结束后，OPC 服务器会自动终止连接。</p> <p>这样可在变量访问间隔时间较长的情况下减少所需的连接数量。</p> <p>数据类型 VT_UI4，只读。</p> <p>0: 不终止 &gt;0: 终止前的空闲时间 (ms)</p>
&optimizes7read()	<p>优化 S7 块访问的读取访问。</p> <p>数据类型 VT_BOOL，只读。</p> <p>“True”：优化 “False”：不优化</p> <p>优化含义： 将各个变量的若干访问作业内部转换为对通信伙伴的单个数组访问。</p>
&optimizes7write()	<p>优化 S7 块访问的写入访问。</p> <p>数据类型 VT_BOOL，只读。</p> <p>“True”：优化 “False”：不优化</p> <p>优化含义： 将各个变量的若干访问作业内部转换为对通信伙伴的单个数组访问。</p>
&autopasswordreset()	<p>自动重设用于块访问的 S7 密码</p> <p>数据类型 VT_BOOL，只读。</p> <p>“True”：启用重设 “False”：禁用重设</p> <p>在 S7 上，通过密码启用的域服务保持激活状态直至将其显式重设。 在建立连接时自动重设密码可确保密码启用不会占用过多的时间，特别是与不使用连接时的自动终止结合使用的情况下。</p>

## 2.6 S7 通信

诊断变量	说明
&fastconnectionstate-returnenable()	如果连接中断，则快速返回写入/读取作业。 数据类型 VT_BOOL，只读。 “True”: 启用 “False”: 禁用
&connecttimeout()	连接建立超时 数据类型 VT_UI4，只读 0: 无超时 >0: 超时 (ms)
&timeout()	S7 通信中生产作业的作业超时。 数据类型 VT_UI4，只读 0: 无超时 >0: 超时 (ms)
&receivecredit()	在接收方向上同时进行的网络作业的最大数目 连接建立的建议值 数据类型 VT_UI2，只读 ≥1, 连接建立的建议值
&tradedreceivecredit()	建立连接后，在所协商的接收方向上同时进行的协议作业数目。 数据类型 VT_UI2，只读 如果连接中断，则此变量的质量为“不良”。
&sendcredit()	在发送方向上同时进行的网络作业的最大数目 连接建立的建议值 数据类型 VT_UI2，只读 ≥1, 连接建立的建议值 在组态中与 &receivecredit() 一起设置。
&tradedsendcredit()	建立连接后，在所协商的发送方向上同时进行的协议作业数目。 数据类型 VT_UI2，只读 如果连接中断，则此变量的质量为“不良”。
&pdu size()	PDU 的大小 连接建立的建议值 数据类型 VT_UI2，只读 ≥1, 连接建立的建议值

诊断变量	说明
&tradedpdusize()	连接建立后所协商的 PDU 大小。 数据类型 VT_UI2, 只读 如果连接中断，则此变量的质量为“不良”。
&defaultalarmseverity()	未组态报警事件的默认报警严重程度。 数据类型 VT_UI2, 只读 1: 低优先级 ... 1000: 高优先级 在组态中, S7 报警和 S7 诊断报警的默认报警优先级只有一个选项。
&defaultdiagnosisseverity()	未组态诊断事件的默认报警严重程度。 数据类型 VT_UI2, 只读 1: 低优先级 ... 1000: 高优先级 在组态中, S7 报警和 S7 诊断报警的默认报警优先级只有一个选项。
&events()	在连接伙伴上注册报警和事件。 数据类型 VT_UI4, 只读。 可组合各个值。 <ul style="list-style-type: none"> <li>0x00000001 扫描数据项（不再支持）</li> <li>0x00000002 简单报警（不再支持）</li> <li>0x00000004 简单符号相关报警（不再支持）</li> <li>0x00000008 Simotion TO 报警</li> <li>0x00000010 连接监视报警</li> <li>0x00000020 块相关报警（作为条件事件）</li> <li>0x00000040 符号相关报警（作为条件事件）</li> <li>0x00000080 诊断报警</li> </ul>

## 2.6 S7 通信

诊断变量	说明
&connectiontype()	S7 连接类型 数据类型 VT_UI2, 只读 2:S7D_STD_TYPE; 标准连接 3:S7D_H_TYPE; 容错连接 如果 S7 连接尚未建立, 则此数据项将报告为质量“不良”, 且值无效。
&connectionstate()	S7 连接状态 数据类型 VT_UI2, 只读 0x11:STD_DOWN; 故意终止标准连接 0x12:STD_ABORT; 无意终止标准连接 (错误) 0x13:STD_NOT_USED; 从未建立标准连接 0x14:STD_OK; 已建立标准连接 0x20:H_OK_RED; 已建立容错连接 (冗余) 0x21:H_OK_RED_PATH_CHG; 已建立容错连接 (冗余, 提供故障切换) 0x22:H_OK_NOT_RED; 非冗余的已建立容错连接 0x23:H_ABORT; 无意终止容错连接 (错误) 0x24:H_NOT_USED; 从未建立容错连接 0x25:H_DOWN; 故意终止容错连接 数据变量与 &statepath() 变量的区别在于, 数据变量从协议栈角度反映连接状态, 还提供有关所用冗余 H 连接的详细信息 (对 OPC 服务器透明)。
&hconnectionwaystate()	H 连接路径的状态 数据类型 VT_ARRAY   VT_UI2 这些值可用于描述 H 连接的路径状态。 0x30:HW_PROD; 路径是生产连接 0x31:HW_STBY; 路径是备用连接 0x32:HW_ABORT; 无意终止路径 (错误) 0x33:HW_NOT_USED; 从未建立路径 0x34:HW_DOWN; 故意终止路径 0x35:HW_CN_BREAK; 无法建立路径 如果 S7 连接尚未建立, 则此数据项将报告为质量“不良”, 且值无效。

### 2.6.6.3 系统特定信息变量的语法

S7:[SYSTEM]&version()

#### 说明

在 STEP 7 中组态的连接名称不能是“SYSTEM”，因为这是保留名称。

#### &version()

返回 S7 OPC 服务器的版本 ID。例如，此处的返回字符串为

*SIMATIC NET Core Server S7 V 7.xxxx.yyyy.zzzz Copyright 2012*

数据类型： VT\_BSTR

访问权限： 只读

S7:[SYSTEM]&sapiversion()

#### &sapiversion()

返回 SAPI S7 用户界面的版本 ID。

数据类型： VT\_BSTR

访问权限： 只读

### 2.6.7 块管理服务

块管理服务可控制 PC 和可编程控制器间的数据及程序元素的传送和返回。在 PC 上，数据和程序元素存储在文件里。可通过密码保护块管理服务。

#### 块

数据和程序元素以块的形式存储在 S7 可编程控制器上。这些块使用 STEP 7 进行编译，并被传送到 S7 设备。S7 设备上存在以下块：

- 组织块
- 函数块
- 数据块 (DB/DI)

## 块管理函数

可通过 SIMATIC NET OPC 服务器执行下列功能：

- 在 PC 和可编程控制器之间传送块
- 删除块
- 链接块
- 压缩可编程控制器的存储器

### 说明

不能使用 OPC 服务器创建块，必须使用 STEP 7 创建块。

### 2.6.7.1 块管理服务的控制变量的语法

#### 语法

S7: [<connectionname>]<serviceparameter>

#### 说明

##### S7

用于访问过程变量的 S7 协议。

##### <connectionname>

协议特定的连接名称。连接名称在组态中指定。

##### <serviceparameter>

只能对这些变量执行写入操作。通过写入值触发服务。

可能值如下：

##### **&blockread()**

将可编程控制器中的块传送到 PC 并存储在文件中。

写入值包含服务的参数设置。它表示为以下元素的变型数组：

**标志**

可能有以下十六进制数字：

0x0001	读取未链接块。 如果目标文件存在，则不覆盖此文件，并显示一条错误消息。
0x0040	读取已链接块。 如果目标文件存在，则不覆盖此文件，并显示一条错误消息。
0x1001	读取未链接块。 覆盖现有目标文件。
0x1040	读取已链接块。 覆盖现有目标文件。

OLE 数据类型	Visual Basic 类型
VT-BSTR	字符串

**块**

块类型和编号

OB	组织块
FB	函数块
FC	函数
DB	数据块

OLE 数据类型	Visual Basic 类型
VT-BSTR	String

**文件**

用于存储块的文件的完整路径

OLE 数据类型	Visual Basic 类型
VT-BSTR	String

**&blockwrite()**

将块从 PC 传送到可编程控制器。

传送完成后，可编程控制器中的块仍处于被动（未链接）状态。必须使用 **&blocklinkin** 函数将块设置为已链接状态。

写入值包含此服务的参数设置。它表示为以下元素的变型数组：

#### 标志

可能有以下十六进制数字：

0x1000	将覆盖自动化系统中的同名未链接块。
0x0000	不覆盖自动化系统中的同名未链接块。

OLE 数据类型	Visual Basic 类型
VT-BSTR	String

#### 文件

用于存储块的文件的完整路径

OLE 数据类型	Visual Basic 类型
VT-BSTR	字符串

---

#### 说明

域服务 **blockread()** 和 **blockwrite()** 不允许访问网络驱动器上块的源文件和目标文件。

只能在链接可执行部分后才能覆盖 (0x1000) 块。

---

#### **&blocklinkin()**

将被动状态的块链接到可编程控制器的程序顺序中。

将块的可执行部分复制到可编程控制器的工作存储器。

这样自动化系统的程序就可访问块了。

通过链接块，就可在不出现错误消息的情况下覆盖一个现有的激活块。

写入值包含作为此服务的参数设置而链接的块的规范。

**块**

块类型和编号。

可能的类型如下：

OB	组织块
FB	函数块
FC	函数
DB	数据块

OLE 数据类型	Visual Basic 类型
VT-BSTR	字符串

**说明**

不能终止 *blockwrite()* 和 *blocklinkin()* 之间的 S7 连接，否则将会立即再次删除远程 CPU 上的块。随后 *blocklinkin()* 函数不再工作。

**&blockdelete()**

删除可编程控制器上的块。将移除被动块以及链接到程序顺序中的块。

写入值包含此服务的参数设置。它表示为以下元素的变型数组：

**标志**

可能有以下十六进制数字：

0x0001	删除未链接块
0x0040	删除已链接块
0x0041	删除已链接块和未链接块

OLE 数据类型	Visual Basic 类型
VT-BSTR	String

块

块类型和编号。

可能的类型如下：

OB	组织块
FB	函数块
FC	函数
DB	数据块

OLE 数据类型	Visual Basic 类型
VT-BSTR	String

### &blockcompress()

压缩可编程控制器的存储器。

碎片存储区移动到一起以为要传送的新块腾出空间。

只能对此变量执行写入操作。通过写入空字符串触发服务。

### 说明

某些块管理服务的参数通过写入值进行设置。此处，使用数组作为数据类型。

从用户角度来看，作为数据类型 *VT\_BSTR* 的值来传值参数更有利。OPC 服务器将值自动转换为所需数据类型。例如，字符串数组可表示为

*{firstelement|secondelement|thirdelement}*.

也可参见示例。

## 2.6.7.2 使用块管理服务的示例

下面几个示例说明了块管理服务的变量名称的语法。

### 读取块

*S7:[S7-OPC-1]&blockread()*

### &blockread()

例如，要读取文件“c:\temp\ob1.blk”中的块 *OB1*，就必须用以下值写入数据项：

---

{0x0040|OB1|c:\temp\ob1.blk}

---

### 说明

要指定此示例中的值，就必须请求字符串作为数据类型。  
然后可用数组的转换表示来指定值。

---

## 写入块

*S7:[S7-OPC-1]&blockwrite()*

### &blockwrite()

例如，要将文件“c:\temp\ob1.blk”中的块发送到可编程控制器，就必须用以下值写入数据项：

{0x1000|c:\temp\ob1.blk}

---

### 说明

要指定此示例中的值，就必须请求字符串作为数据类型。  
然后可用数组的转换表示来指定值。

---

## 链接块示例

*S7:[S7-OPC-1]&blocklinkin()*

### &blocklinkin()

例如，要将块 DB1 链接到可编程控制器上的程序执行中，就必须用以下值写入数据项：

DB1

## 删除块示例

*S7:[S7-OPC-1]&blockdelete()*

### &blockdelete()

要删除链接块 DB1，就必须用以下值写入数据项：

{0x0040|DB1}

## 压缩可编程控制器的存储器的示例

*S7:[S7-OPC-1]&blockcompress()*

**&blockcompress()**

要压缩存储器，就必须用空字符串写入数据项。

## 2.6.8 密码

在组态过程中，密码可保护块管理服务对 CPU 的读访问和写访问。与 CPU 的按键开关相比密码有更高的优先级。

### 保护等级

S7 可编程控制器有三个保护等级。激活这些保护等级需要借助于 STEP 7：

- 保护等级取决于按键开关设置
- 写保护
- 读写保护

在传送正确密码时，会撤消此连接的所有保护等级。

传送空字符串作为密码时，会重新激活连接保护。

两端连接组态期间，可发送默认的保护等级。

如果组态未激活两端连接的保护，则所有服务均可能位于这些连接上，即使按键开关设置为 *RUN* 时也是如此。

组态为单端的连接取决于按键开关设置。

下表显示了在单端上进行组态的连接上密码传送与保护等级之间的关系。

单端组态连接	生成的激活保护等级				
组态的保护等级	不传送密码		传送正确的密码		
	按键开关设置			按键开关设置	
	RUN	RUN-P、 STOP	RUN	RUN-P、 STOP	
按键开关设置 <b>RUN -</b> 可通过密码取消	读取	全部	全部	全部	
按键开关设置 <b>RUN -</b> 不能通过密码取消	读取	全部	读取	全部	

单端组态连接	生成的激活保护等级			
带密码的写保护	读取	读取	全部	全部
读写保护	无	无	全部	全部

要点：

全部 = 所有可能的服务

读取 = 仅允许读取块

无 = 不允许块服务

### 2.6.8.1 密码的控制变量的语法

#### 语法

S7: [<connectionname>] &password()

#### 说明

##### S7

用于访问过程变量的 S7 协议。

##### <connectionname>

协议特定的连接名称。连接名称在组态中指定。

##### &password()

密码服务的标识符。

通过写入值触发密码传送。

可通过以下表示方法传送密码：

- 八位位组串  
由密码的各个字符的十六进制代码组成的字符串，由句点分隔（最多 8 个字符）
- 字符串  
以“&”字符开始的任意字母数字字符串（最多 8 个字符）

写入密码返回以下值：

- OPC\_E\_BADRIGHTS  
密码无效
- S\_OK  
密码有效

OLE 数据类型	Visual Basic 类型
字符串	字符串

### 2.6.8.2 使用密码的示例

下面即为传送密码的示例。

#### 传送密码的示例

*S7:[S7-OPC-1]&password()*

**&password()**

要将密码 *SetMeFre* 传送到可编程控制器，就必须用以下值写入变量：

**&SetMeFre**

### 2.6.9 服务器服务

#### 访问 DB 1

带有适用于 S7 协议的激活的 OPC 服务器的 PC 站提供了用于读取和写入的 S7 数据块，因此成为了 S7 服务器。通过 PUT 和 GET 函数块，S7 站可在程序中写入或读取 PC 站的数据块。两端均可建立连接。

数据块 (DB 1) 的长度为 65535 个字节。

数据块即没有符号名称，也不能根据变量进行结构化。

远程和本地都能查询数据块的编号，例如：装有 OPC Scout 的 SIMATIC NET OPC 服务器可查询数据块编号，然后在浏览命名空间时将其显示出来。

只有一个数据块可用。没有位存储器、输入、输出、计数器或定时器。

无需组态或下载块。

通过本地 S7 连接 *@LOCALSERVER*，OPC 客户端（例如 OPC

Scout）可读取或写入数据块的值。OPC

客户端也可监视数据块，并在数据更改时得到通知。

如果有多个客户端尝试写入相同的数据区域，则要在每个作业全部完成后再进行下一个（尽管同时访问，但仍可保持数据的一致性）。

PC 站重新启动后，值会保留在数据块中（永久数据）。

---

#### 说明

要运行 S7 服务器服务，本地 OPC 客户端必须已激活，并且 S7 数据项要位于 PC 站中。如果需要在 NCM 中组态建立连接，则 S7 数据项必须包含 S7 连接，远程伙伴可通过这些连接读取或写入数据。也可以选择组态建立永久连接。被动连接需要建立永久连接。然后便可激活任意 S7 数据项，以启动 S7 OPC 服务器。

---

---

#### 说明

仅发布使用 PC 站的 OPC 服务器 V6.3 组态的 S7 服务器服务。

---

### 2.6.9.1 使用服务器服务的示例

例如，S7 客户端可为 S7 300 站，可向 PC 站报告状态数据，但不需要连续不断地轮询状态数据，请参见下图。然后（偶尔）将状态值写入数据块。

状态值变化时将通知 PC 站上的本地客户端，请参见图“通过定期周期性读取值进行监视...”。这样便可避免连续不断地轮询 S7 站上的状态值。

下图说明了 S7 设备如何将数据写入装有 S7 OPC 服务器的 PC 站。

## 2.6 S7 通信

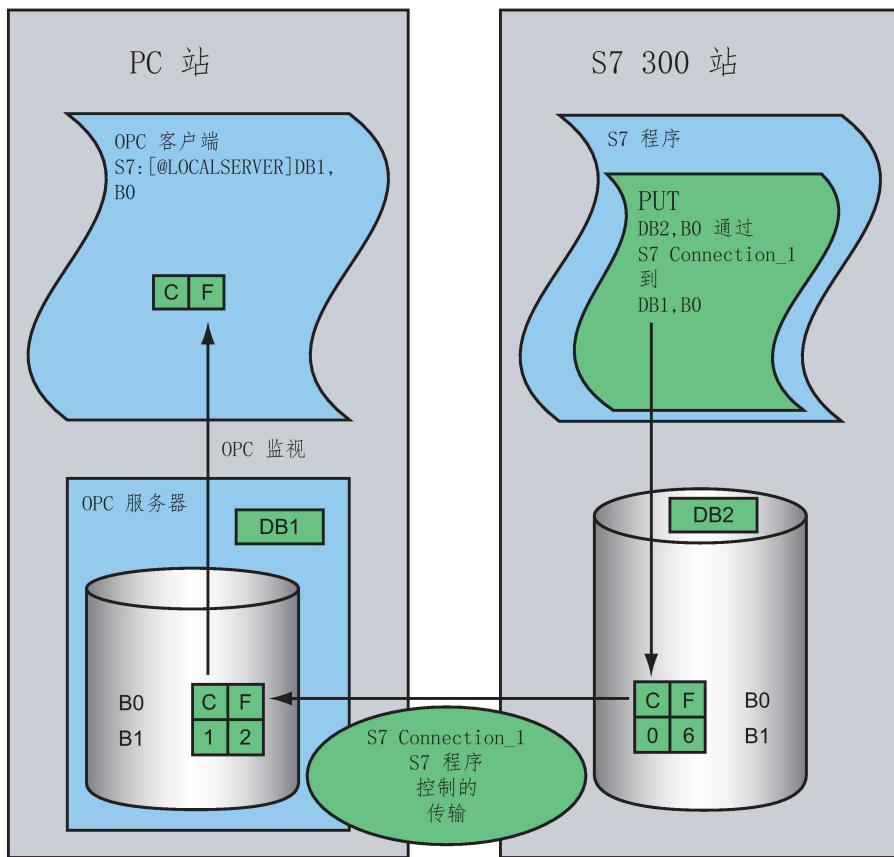


图 2-15 服务器服务：偶尔写入 (PUT) 值可减轻 S7 站的负担

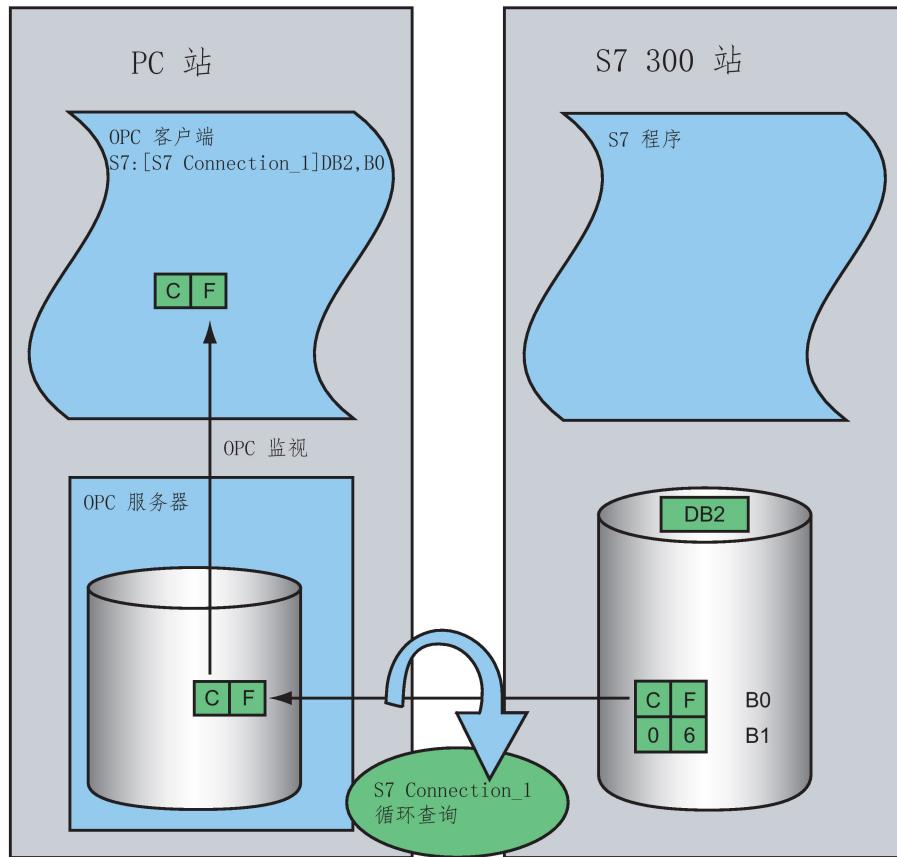


图 2-16 通过定期周期性读取值进行监视（轮询）；S7 站上的额外负载

## 2.6.10 S7 模板数据变量

### 通过 S7

协议的过程变量，您可拥有灵活的设置选项以所需的数据格式获取设备的过程数据。

不过，在完全可浏览的命名空间中，不能将各种寻址选项放在一起。

即使单字节长度的数据块也有大约 40 个不同的数据格式选项 - 从 Byte 和 Char 开始，由这些数据类型的一个元素组成的数组、单个位、最多 8 个数组元素的位数组，每个元素均起始于不同的位偏移。

因此，OPC 服务器支持 S7 命名空间中具有这些所谓模板数据变量的用户。在典型的 OPC 客户端文本输入框中，只需更改几个字符，就可将这些模板转换为有效的 ItemID。

示例：

`S7: [S7-connection_1] DB<db>, DWORD<o> //DWORD 数据块模板`

通过将 `<db>` 替换为块地址，将 `<o>` 替换为数据块中的偏移，用户便可获得有效的数据项语法。

## 2.6 S7 通信

```
⇒ S7:[S7-connection_1]DB100,d3
```

另一个示例：

```
S7:[S7-connection_1]bsend<len>,<rid>,x<o>.<bit>,<c>//位数组缓冲区服务的模板
```

```
⇒ S7:[S7-connection_1]bsend100,21,x0.0,24
```

---

### 说明

可以在“通信设置”组态程序中启用和禁用 S7 OPC UA 模块数据变量的可用性：在“OPC 协议选择”(OPC protocol selection) 中单击“S7”旁的箭头符号。

---

### 2.6.10.1 命名空间中的模板数据变量

模板数据变量位于单独文件夹内的命名空间中。

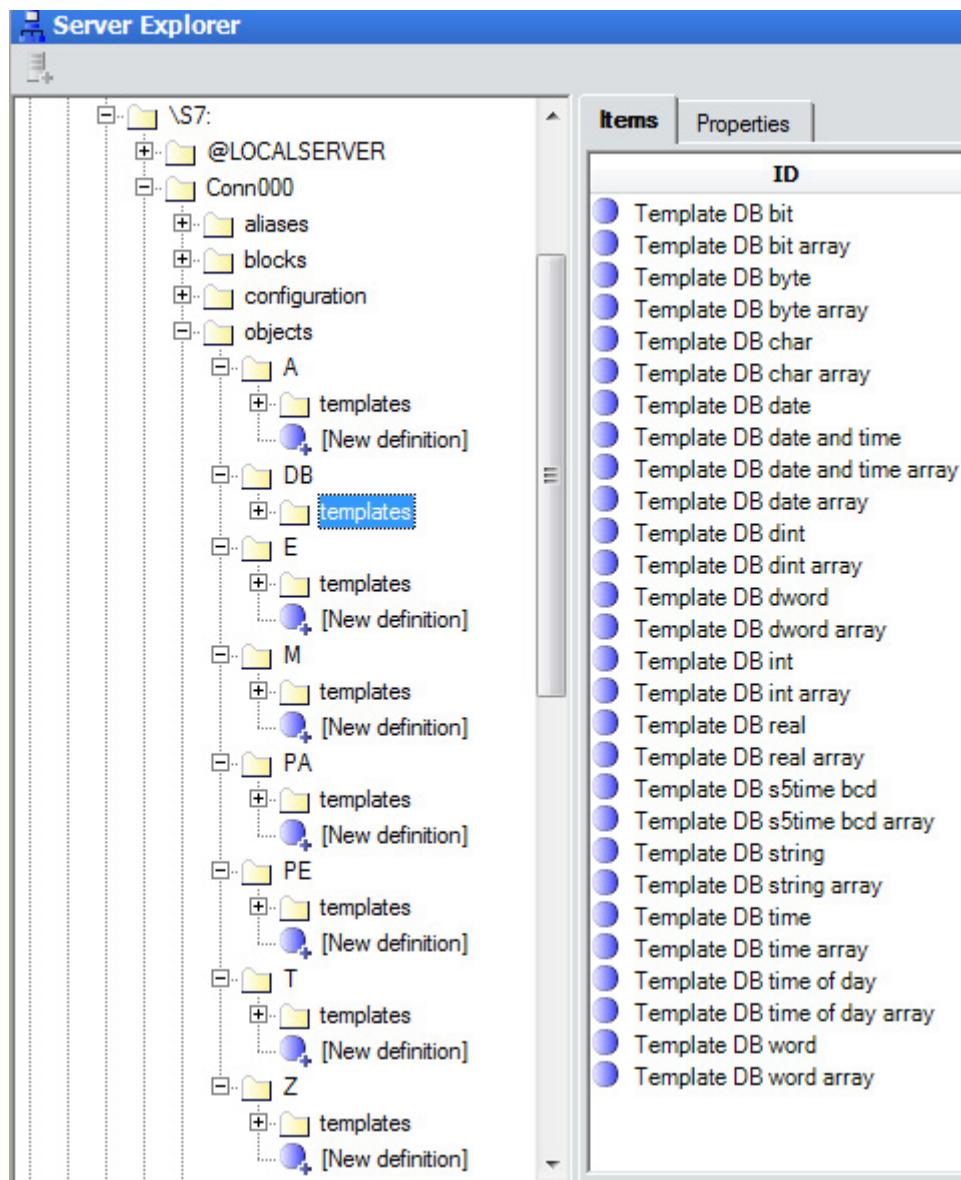


图 2-17 模板数据变量的层次结构

命名空间中的叶名称可提供相当大的帮助（例如：“模板 M 字节”(Template M Byte)）。

### 2.6.10.2 模板数据变量的语法

存在以下模板：

## 用于位存储器、输入和输出

```
S7: [<connectionname>] <object><type><o>, <c>

<object>:=    "I" | // 输入
                "Q" | // 输出
                "M" | // 存储器位
                "PI" | // 外围设备输入
                "PQ" // 外围设备输出

<type><o>:=   "X<o>.<bit>" | // 位 <bit>= 位地址模板
                "B<o>" | // 字节（无符号）
                "W<o>" | // 字（无符号）
                "D<o>" | // 双字（无符号）
                "CHAR<o>" | // 字节（有符号）
                "INT<o>" | // 字（有符号）
                "DWORD<o>" | // 双字（有符号）
                "REAL<o>" | // 浮点数，4 字节
                "DT<o>" | // 日期和时间，8 字节，BCD 格式
                "DATE<o>" | // 日期和时间，8 字节，时间格式始终采用 00:00:00
                "TIME<o>" | // 时间值（有符号），IEC 格式，单位 ms
                "S5TIMEBCD<o>" | // 时间变量（无符号，16 位），0 至 9990000 ms
                "TOD<o>" | // 日时钟（无符号），0 至 86399999 ms
                "STRING<o>.<len>" // 字符串。<len> 是字符串长度模板
```

<o> // 第一个变量地址的模板，其为地址范围内的字节偏移。

<c> // 某一要寻址类型的变量数目的模板，起始于地址参数中指定的偏移。

## 对于数据块

```
S7: [<connectionname>] DB<db>, <type><o>, <c>
```

<db> // 数据块编号模板

## 缓冲区发送/接收服务

S7: [<connectionname>] bsend<len>,<rid>,<type><o>,<c>

S7: [<connectionname>] brcv,<rid>,<type><o>,<c>

<len> // 要发送的数据块的长度（字节）模板。

<寻址参数 ID> // 寻址参数的 ID 模板。此模板针对块对 (BSEND/BRCV) 指定并在连接内唯一定义。

## S7 定时器

S7: [<connectionname>] TDA<i> // S7 定时器变量（类型 TDA）：

S7: [<connectionname>] TBCD<i>{,<c>} // S7 定时器变量（类型 BCD）：

## 计数器

S7: [<connectionname>] C<i>{,<c>} // 计数器，地址信息 <i> 是计数器编号。

## 2.6.11 未组态 S7 连接

### 访问未组态伙伴

通常在组态中定义伙伴设备的连接。在 STEP 7 或 NCM PC 程序中完成此定义。

但也有特殊情况，例如，在无组态连接的情况下，必须从伙伴设备或已写入或监视的变量中读取数据。可以在未组态的情况下执行这些任务。

### 要求

对于未组态的设备访问，与通信相关的伙伴设备的所有数据必须已知。

其中包括连接名称、访问点和站地址。下节列出了所需参数。

选择连接名称时，确保名称唯一且在已组态名称中还不存在。

### 服务访问点 (SAP)

服务访问点是一个点，在该点 ISO/OSI 参考模型的某一层可向下一个更高层提供服务。

两个相邻层之间的整个信息交换通过服务访问点进行。

该点为较高层和较低层之间的接口。如果 SAP 位于第 3 层（网络层）与第 4 层（传输层）之间，则服务访问点称为 NSAP（Network Service Access

Point, 网络服务接入点)。如果 SAP 在第 4 层和第 5 层(会话层)之间，则该 SAP 称为 TSAP (Transport Service Access Point, 传输服务接入点)。

通常将某些资源和通信伙伴分配给服务访问点，因此通信所需的 SAP 通过唯一名称和编号进行标识。

## 未组态连接的参数

未组态连接的语法如下：

S7:[<connectionname>|<VFD>|<accesspoint>|<address\_specification>]<dataelement>

各个组成部分具有下列意义：

### <connectionname>

不能使用已存在的连接名称。如果所选连接名称已组态并用于另一个自由指定的 S7 连接，则将忽略自由指定的连接的其它信息，并会将数据项分配给现有连接。

### <VFD>

VFD 名称与 NetPro 的 PC 站组态中的应用程序名称相同。

任意 VFD 名称。所有连接都可以在同一个 VFD 上创建。VFD 名称最长可为 32 个字符。不能使用已存在的所选名称。

### <accesspoint>

通信模块的访问点必须事先通过“通信设置”程序进行组态。

<address\_specification> = <local TSAP>,<stationaddress>,<remote TSAP>,<mode>

地址规范包含以下信息，各个值通过逗号分隔：

- **Local TSAP (Local Transport Service Access Point, 本地传输服务访问点)**

对于 S7 协议，本地 TSAP

正好由两个编号组成，各部分由空格或句点分隔，其含义如下：

- 第一个字节可包括设备 ID，允许值为 02 或 03:

02 OS (操作站控制和监视)

03 其它

- 第二个字节始终为 0。

推荐设置：02.00

- **stationaddress**

站地址有三种可能的表示方法：

传输技术	站地址表示法	示例
PROFIBUS	PROFIBUS 地址, 十进制表示法	65
TCP/IP	TCP/IP 地址	192.168.0.7
ISO	MAC 地址	08-06-05-e4-3a-00

**remote TSAP (Remote Transport Service Access Point, 远程传输服务访问点)**

其表示方法与 local TSAP 相同, 但第二个字节具有不同含义:

- 第 1 个字节: 包含设备 ID, 允许值为 *02* 或 *03*:
  - 02 OS* (操作站控制和监视)
  - 03* 其它
 推荐设置: *02*
- 第 2 个字节: 包含 SIMATIC S7-CPU 寻址,
 分成:
  - 位 7 至 5, S7-CPU 的机架 (子系统)
  - 位 4 至 0, S7-CPU 的插槽

**说明**

设置“local TSAP”和“remote TSAP”时, 建议为第一个字节选择相同的设置。

**Mode**

S7 连接由 OPC 服务器或连接伙伴主动建立。

在此类连接上, 也可以使用优化的写入或读取访问。模式可能使用以下值:

- *1*: 由 OPC 服务器主动建立连接 (优化)
- *3*: 由 OPC 服务器主动建立连接 (无优化)

**<dataelement>**

例如, 此处可指定数据块及其编号和类型 (字节、字等) 以及地址 (例如: 字节偏移)。有关可用于 S7 的数据元素, 请参见“S7 通信 (页 133)”。

**说明**

更多连接参数:

无法指定 PDU 大小。连接建立期间建议最大大小为 960 字节, 并协商伙伴设备的最大可能大小。

用于连接建立超时和作业超时的固定值 (在每种情况下均为 15000 ms)。并行网络作业的最大数目的建议值为 2。

## 2.6 S7 通信

### 示例

*S7:[S7-connection 1|VFD1|S7ONLINE|01.00,192.168.0.7,02.02,1]DB10,B0*

*S7:[S7\_connection 2|VFD2|S7ONLINE|02.00,65,02.02,1]DB10,B0*

*S7:[S7\_connection 3|VFD3|S7ONLINE|03.00,08.06.05.e4.3a.00,02.02,1]MB0*

## 定义访问点

1. 打开“通信设置”组态程序，定义访问点并指定接口参数分配。

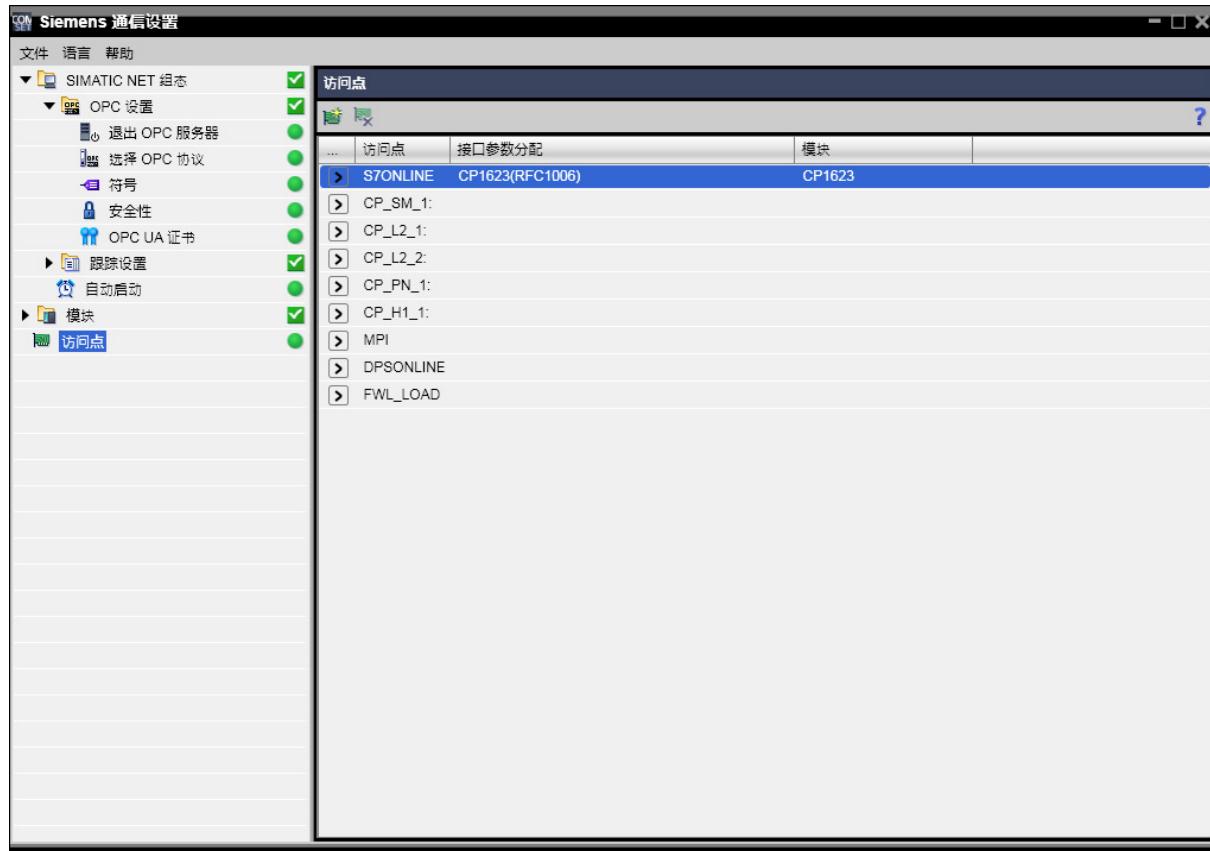


图 2-18 打开“通信设置”组态程序

2. 单击“S7Online”旁的箭头符号。
3. 使用“相关接口参数分配”(Associated interface parameter assignment)下拉列表选择访问点的接口参数分配。

## 2.6 S7 通信

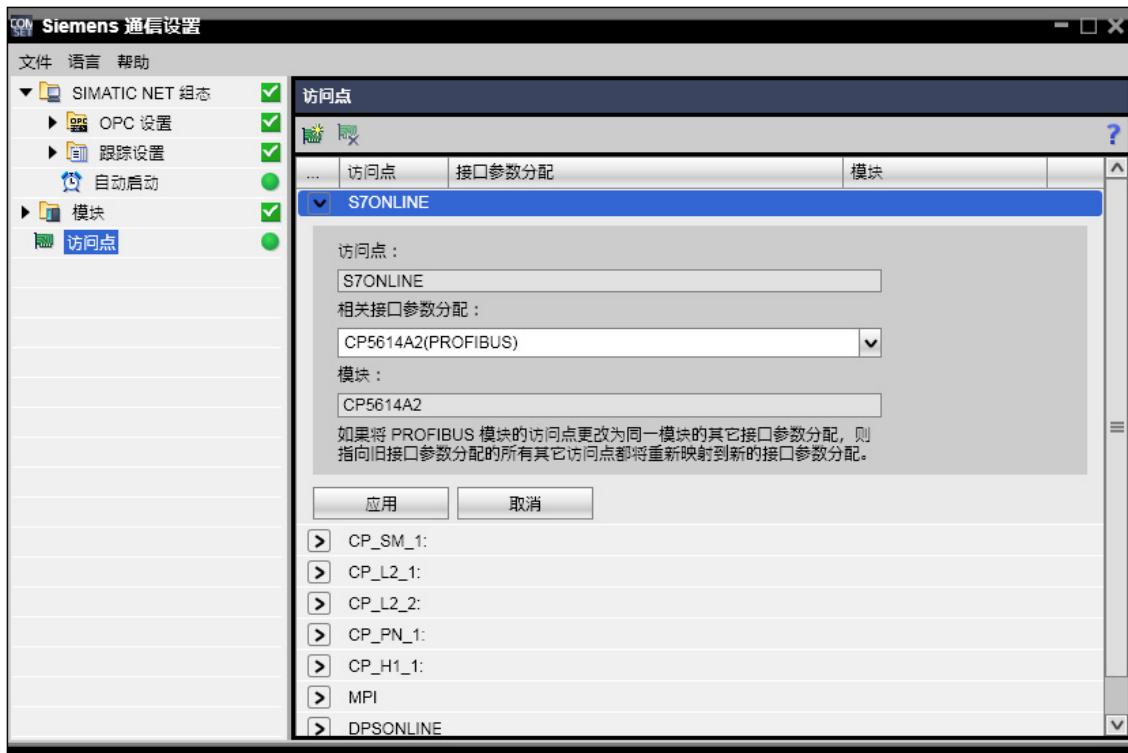


图 2-19 指定接口参数分配

### 添加数据项

打开客户端程序，使用上面所示的语法创建数据项。在 OPC Scout V10 程序中，将任意数据项从“S7”文件夹拖动到 DA 视图，然后单击“ID”表列右边的“...”按钮。现在输入语法，然后单击“确定”(OK) 进行确认。

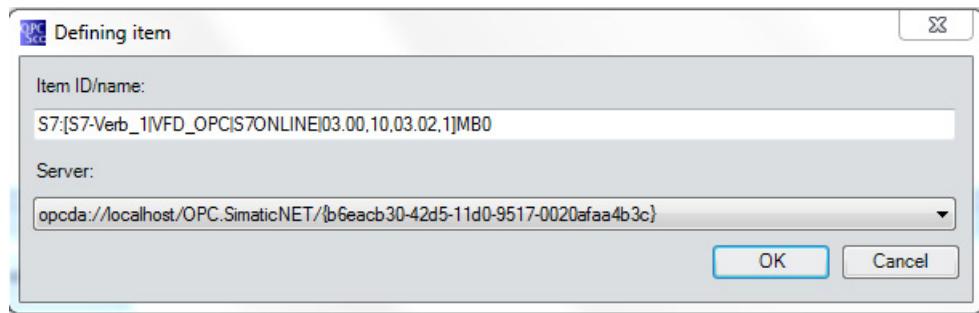


图 2-20 OPC Scout V10 - 插入数据项

添加数据项后，只要数据项处于激活状态，就可以像使用组态连接一样使用相应连接。这意味着，您可以在命名空间中浏览，也可以添加更多数据项，而无需使用未组态连接的语法。您只需要指定连接名称，例如 S7:[S7-Verb\_1]MB1。

## 2.6.12 COML S7 连接

使用本地 COML S7 连接访问伙伴设备

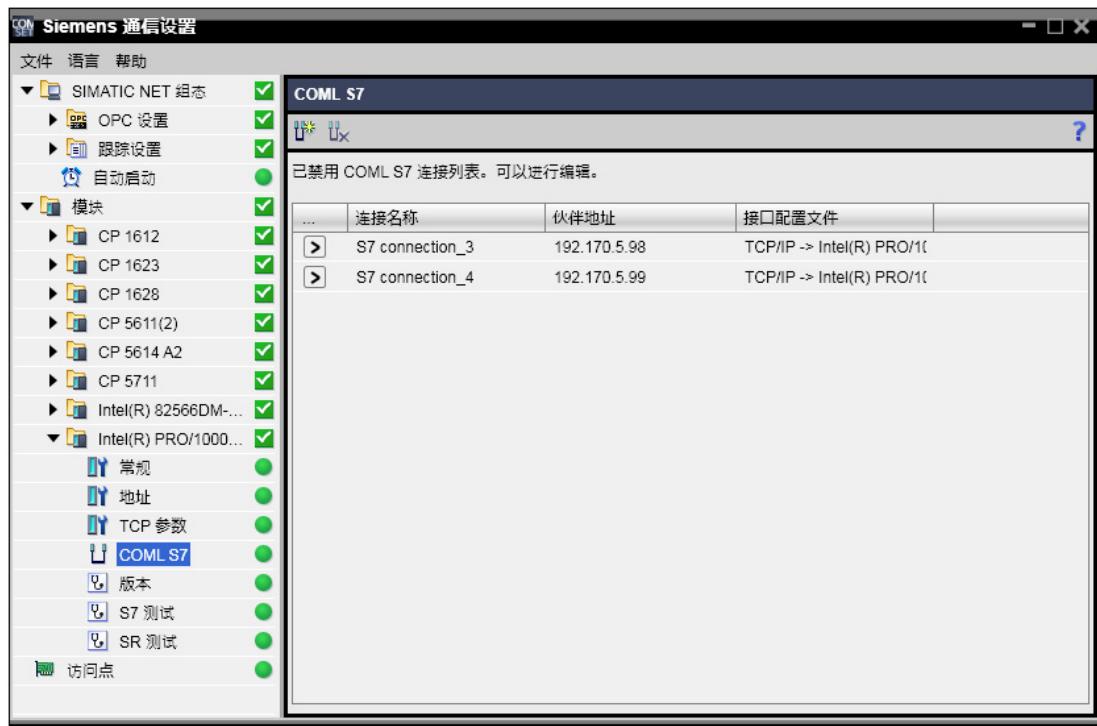


图 2-21 COML S7 - 本地组态管理 - S7 连接的组态软件

可使用 COML S7 组态软件创建 S7 CPU 或 PC 的 S7 连接。

“通信设置”组态程序中的每个模块下面都有“COML S7”选项卡。如果单击此选项卡，则 COML S7 连接列表将会显示在对话框窗口的右侧，您可以为所选模块创建伙伴设备的 S7 连接。组态工作产生的数据将收集在适用于所有模块的数据库中，然后保存在 PC 上。该数据库可导出为备份，然后重新导入。此后，必须先在“模块”(Modules) 层级的快捷方式存储器中激活 COML S7 组态，然后才能使用 COML S7 连接列表中的已创建 S7 连接进行 S7 通信。

## 使用 COML S7 连接的要求

对于使用 COML-S7 组态的设备访问，所有与通信相关的伙伴设备数据必须已知。

例如，其中包括伙伴 TSAP 和伙伴地址。创建 COML S7

连接时，有关各参数的更多详细信息，请参见在线帮助。

---

### 说明

不能同时运行 STEP 7/SIMATIC NCM PC 和 COML S7 的已组态 S7 连接。

因为它们处于互锁状态。

启用 COML S7 连接后，会拒绝加载 STEP 7/SIMATIC NCM PC 的 S7 连接。

---

### 说明

不能同时运行已组态连接和未组态连接。因为它们处于互锁状态。

启用已组态连接后，会拒绝运行未组态连接。

---

## 2.7 通过 OPC UA 进行的 S7 通信

### 2.7.1 通过 OPC UA 进行的 S7 通信的属性

SIMATIC NET OPC 服务器允许使用通过 OPC UA 进行的 S7 通信。

SIMATIC NET 的 S7 OPC UA 服务器具有以下特性：

- 变量服务

访问和监视 S7 变量

- 面向缓冲区的服务

通过程序控制较大数据块的传送

- 服务器功能

PC 可用作数据字段和数据块的服务器。

- 块服务

与 S7 之间传送可加载数据区域

- S7 密码功能  
设置用于访问受保护块的密码
- 事件、条件和报警  
处理 S7 报警和 S7 诊断事件

## 2.7.2 S7 协议的 SIMATIC NET OPC UA 服务器

### 简介

以下部分介绍了同样支持 OPC UA 的 S7 协议的组态变型。为此，将 S7 COM OPC 数据访问服务器设置为进程外 OPC 服务器。

## 2.7 通过 OPC UA 进行的 S7 通信

### 组态

通过在“通信设置”组态程序的“OPC 协议选择”(OPC protocol selection) 目录中选择“S7”和“OPC UA”激活 S7 UA 服务器：



图 2-22 “通信设置”组态程序中用于选择 S7 协议的 OPC UA 的窗口

也可选择“符号”(Symbols)。

### 说明

使用 STEP 7 Professional (TIA 门户) 或符号编辑器创建符号时，允许使用以下字符：

A-Z、a-z、0-9、\_、-

、^、!、#、\$、%、&、'、/、(、)、<、>、=、?、~、+、\*、'、:、|、@、[、]、{、}、"。 使用 STEP 7 创建符号时，还应记住，如果符号文件中的符号同时具有 <symbolname> 和 <symbolname>[<索引>] 形式，则解析数组时会出现问题。

## 优点/缺点

在使用 S7 OPC UA 服务器时，只有 S7 的进程外模式可用。必须运行 S7 OPC UA 服务器进程以维持准备接收的状态。即使所有 OPC UA 客户端都已注销，S7 OPC UA 服务器也不会更不能退出。

但另一方面，则具有以下优点：

- 不再需要 COM/DCOM 组态。
- 高速、安全的通信
- 事件和数据访问仅需要一个服务器。
- 缺点：不能同时运行高速进程内的 OPC DA S7 服务器。

## 注意

---

### 说明

如果激活了 S7 协议的 OPC UA，则当首个 OPC UA 客户端登录到 S7 OPC UA 服务器时，会立即建立永久组态的 S7 连接。如果尚未连接任何 OPC UA 客户端（例如，计算机启动后），则不会建立永久的 S7 连接。

---

### 说明

可以在“按需”建立的 SIMATIC STEP 7/NCM PC 中组态 S7 连接。

只能根据需要建立 OPC 功能通信；也就是说建立连接可能需要更长的时间，且 OPC 可能在启动阶段发出错误信号。

---

### 2.7.3 如何寻址 S7 OPC UA 服务器？

#### 服务器 URL

对于本地二进制 TCP 协议，OPC 客户端有两种对服务器进行寻址的方法：

- 直接寻址：
  - opc.tcp://<hostname>:55101
  - 或
  - opc.tcp://<IP-Adresse>:55101
  - 或
  - opc.tcp://localhost:55101

S7 OPC UA 服务器使用端口 55101。

使用常用冗余 IP 地址从外部寻址冗余 S7 OPC UA 服务器。

- 也可以使用 OPC UA 发现服务查找 S7 OPC UA 服务器的 URL。

要查找发现服务器，请输入以下内容：

- opc.tcp://<hostname>:4840
- 或
- opc.tcp://<IP-Adresse>:4840
- 或
- http://<hostname>:52601/UADiscovery/
- 或
- http://<IP-Adresse>:52601/UADiscovery/

发现服务器使用端口 4840（用于 TCP 连接）和端口 52601（用于 HTTP 连接）。

#### IPv6 地址

IPv6 地址也可以用作 IP 地址。地址必须位于括号中，例如

[fe80:e499:b710:5975:73d8:14]

## 端点和安全模式

SIMATIC NET S7 OPC UA 服务器支持具有本地二进制 TCP 协议的端点，并要求使用证书进行验证以及进行加密传送。

已寻址主机上的“发现”服务将用信号通知服务器的端点，换句话说就是服务器的安全要求和协议支持。

S7 OPC UA 服务器的服务器 URL "opc.tcp://<hostname>:55101" 提供了以下端点：

- 在“签名和加密”安全模式下的端点：

必须进行签名和加密才能与服务器进行通信。通过交换证书和输入密码来保护通信。

除了安全模式，还显示了安全策略 Basic128Rsa15。

- 在“无”安全模式下的端点：

在此模式下，服务器不需要使用安全功能（安全策略“无”）。

有关安全功能的详细信息，请参见“对 OPC UA 接口进行编程 (页 586)”部分。

安全策略“Basic128Rsa18”和“无”在 OPC 基金会的 UA 规范中，其 Internet 地址如下所示：

[> "Specifications" > "Part 7"](http://opcfoundation.org/UA)

有关更多的详细信息，请参见以下 Internet 页面：

OPC 基金会 ([> "Security Category" > "Facets" > "Security Policy"](http://www.opcfoundation.org/profilereporting/index.htm))

### OPC Scout V10 的 OPC UA 发现

在 OPC Scout V10 中，您可以打开“OPC UA 发现”(OPC UA Discovery) 对话框并在 OPC Scout V10 的导航区域中输入 UA 端点。

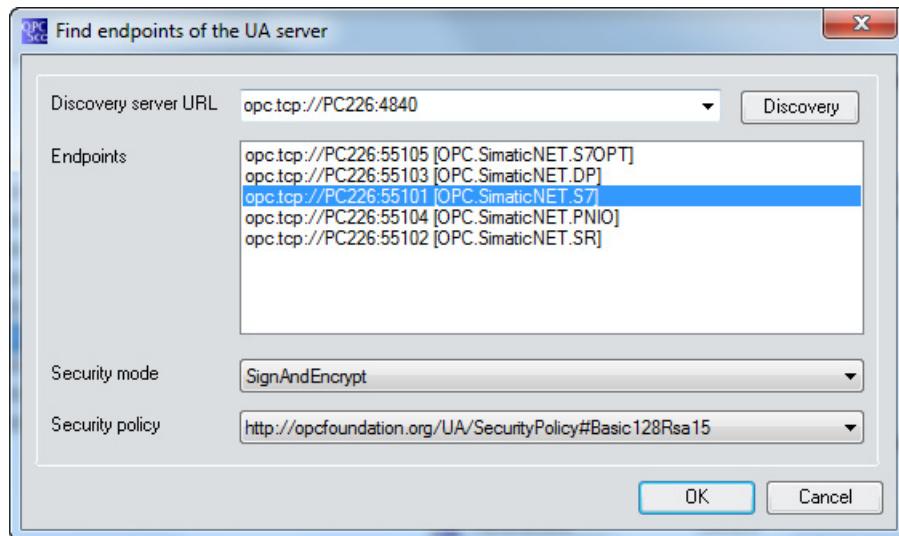


图 2-23 OPC Scout V10 的“查找 UA 服务器端点”对话框

可通过 OPC UA 发现服务查找 S7 OPC UA 服务器。相关条目，请参见上述“服务器 URL”。

OPC Scout V10 包含了一系列 OPC UA 端点。

然后，被寻址主机上的“发现”服务会用信号通知已注册的 OPC UA 服务器、其端口和安全模式。

更多详细信息，请参见 OPC Scout V10 的在线帮助。

## 2.7.4 S7 OPC UA 服务器提供了哪些命名空间？

S7 OPC UA 服务器提供了以下命名空间：

表格 2-2 OPC UA 的命名空间

命名空间索引	“标识符”（命名空间 URI）/注释
0	“ <a href="http://opcfoundation.org/UA/">http://opcfoundation.org/UA/</a> ” (由 OPC 基金会指定)
1	“urn:Siemens.Automation.SimaticNET.S7:(GUID)” 本地高速 S7 OPC UA 服务器的唯一标识符。
2	“S7TYPES:” S7 特定对象类型的定义。
3	“S7:” 本地高速 S7 OPC UA 服务器的标识符，使用新的简化语法（可浏览并可与 UA 一起使用）
4	“S7COM:” 服务器的标识符，使用旧的语法，与 S7 OPC DA 兼容（可与 UA 一起使用，但不能浏览）
5	“S7SOURCES:” 报警和用户诊断事件来源的标识符。
6	“S7AREAS:” 报警层级结构区域的标识符。
7	“SYM:” 可选服务器，使用 ATI-S7 符号；取决于项目工程组态和 PC 站组态（可浏览，也可与 UA 一起使用）。 或者，此处也可使用在符号的参数分配中指定的前缀（“通信设置”） 。

命名空间索引 0 和 1 保留，其意义由 OPC 基金会指定。

将剩余命名空间索引分配至标识符（命名空间  
URI），这一操作必须由指定该标识符的客户端通过 OPC UA  
会话开始处的数据变量“NamespaceArray”来完成。  
标识符“S7TYPES:”、“S7:”、“S7COM:”、“S7AREAS:”和“S7SOURCES:”始终与 S7 OPC  
UA 服务器一起存在。

## 2.7 通过 OPC UA 进行的 S7 通信

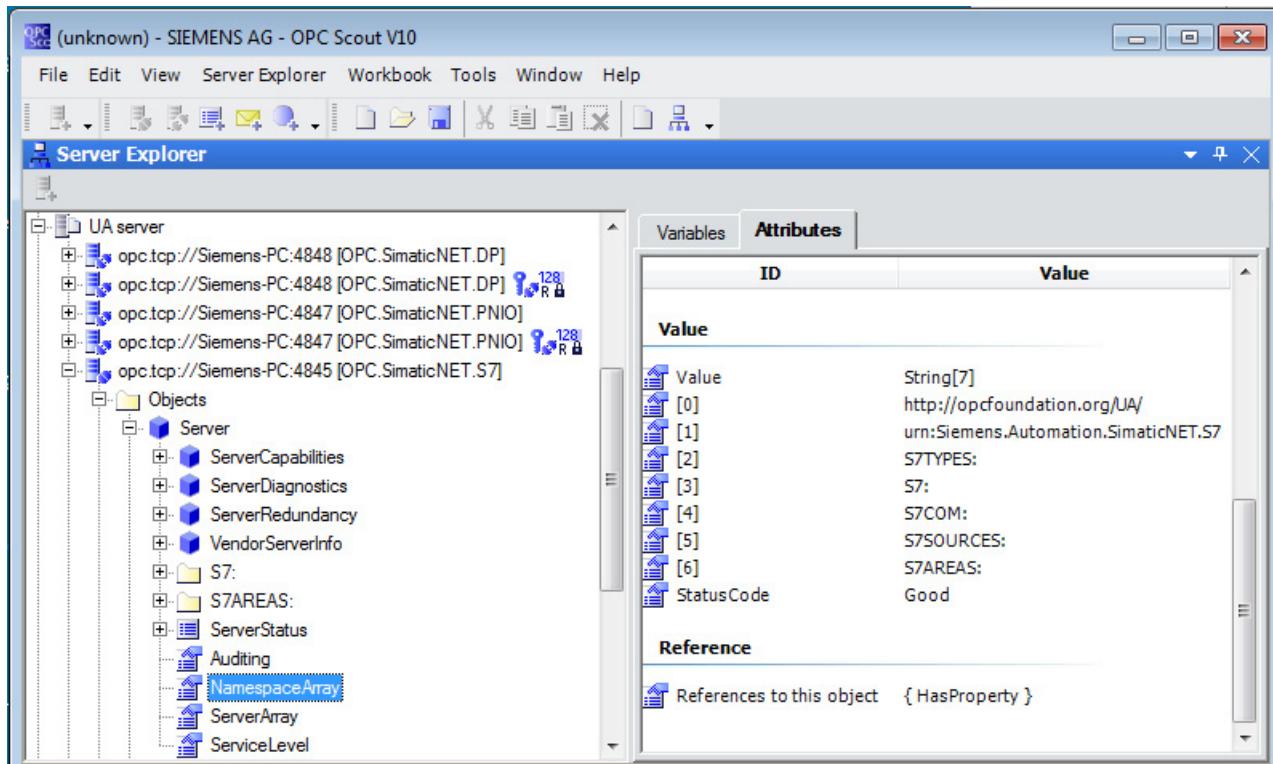


图 2-24 使用 OPC Scout V10 的浏览功能显示 S7 OPC UA 的命名空间

### 2.7.5 Nodeld

#### S7 过程变量的标识

Nodeld 借助下列元组，可标识 S7 过程变量：

- 命名空间索引
- 标识符（字符串、数字值）

## 示例

- NodId:
  - 命名空间 URI:  
*S7:*  
(= 命名空间索引 3) 针对 Siemens.Automation.SimaticNET.S7
  - 标识符:  
*S7\_connection\_1.m.10, b*
  
- NodId:
  - 命名空间 URI:  
*S7COM:*  
(= 命名空间索引 4) 针对 OPC.SimaticNET; 语法兼容 S7 OPC DA
  - 标识符:  
*S7:[S7\_connection\_1]mb10*

## 新的命名空间是如何适应 OPC UA 运作的?

用于读写过程变量的 COM 服务器的“OPC 数据访问”项是一个独立的领域。  
报警领域与其并存但又相互独立。

另一方面，自动化对象的 OPC UA 视图也与对象的各种属性相关。OPC UA 不再单独访问各项，而是还会访问对象及其子对象。

- 诸如数据变量、方法及某种程度上的事件都是 S7 连接对象的子对象。  
特性和属性可更详细地定义对象。
- 访问数据块的“OPC 数据访问”数据项最接近 OPC UA 数据变量。
- 域服务的“OPC 数据访问”数据项最接近 OPC UA 方法。

合格的 NodId 标识符在 OPC UA 中比在“OPC  
数据访问”中更有意义。每次单独访问对象、子对象、特性及属性均使用其 NodId。

OPC UA 提供显示名称等内容以支持本地语言。

也就是说，可通过不同方式浏览相同对象（例如在 OPC UA  
客户端指定的各种语言环境下），每次都显示同一个 NodId。显示名称的选择与相关  
NodId 类似。整个命名空间的文本均采用英语。

## S7 OPC UA 数据对象的语法

OPC UA 中引入了一种优化语法。所有 OPC UA 对象的 NodId 均具有以下结构：

`<connectionobject>."<subobject>."<property>`

子对象可包含其它子对象。

无法解析的 `NodeId` 会被拒绝，并会显示错误消息。任何项中的字母“**A-Z**”都不区分大小写。

## 符号对象表示法

OPC UA 规范建议对地址空间的层级描述使用统一的符号表示法。

本文档中使用以下符号：

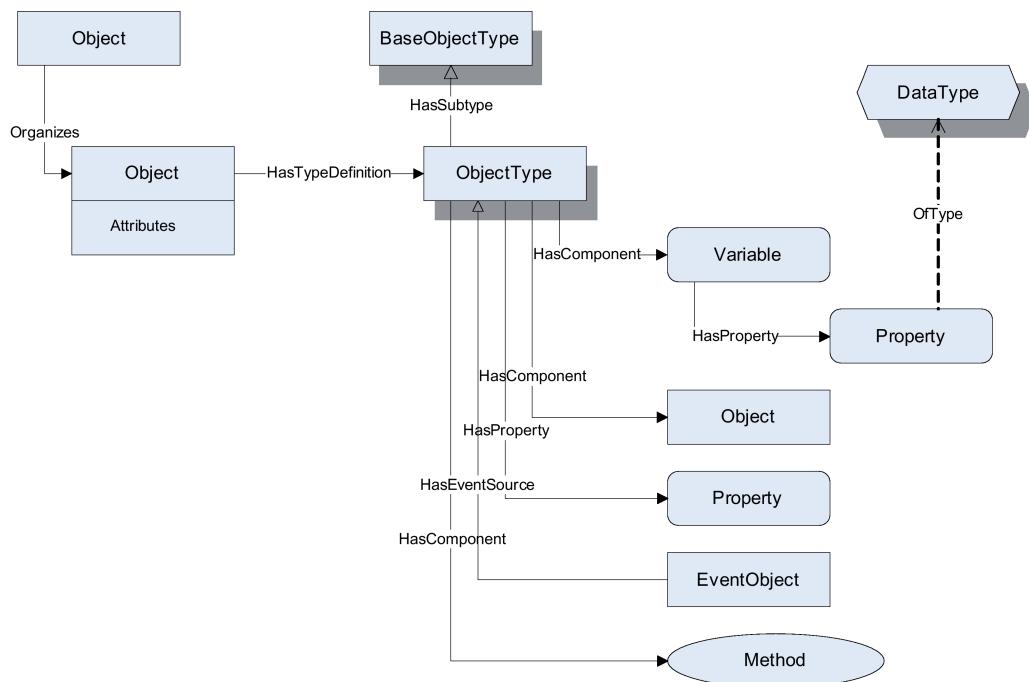


图 2-25 OPC UA 地址空间的符号

## 2.7.6 连接对象

### S7 连接对象

有下列 S7 连接对象：

- 生产 S7 连接

S7 连接用于自动化设备间的数据交换，通常使用 STEP 7 进行组态。

- DEMO 连接

此连接仅供测试使用。

- @LOCALSERVER 连接

此连接提供了用于 S7 服务器功能的本地 S7 数据块。

### Demo 连接

```
<connectionobject>:= "DEMO"
```

称为 DEMO 的演示连接下存在命名空间与 S7 连接类似的对象。

演示连接旨在帮助用户熟悉 SIMATIC NET OPC 系统，可在组态过程中进行添加。

### 说明

名为 DEMO 的演示连接不得与同名的 S7 连接配合使用。

如果对组态添加演示连接，则会忽略组态为该名称的 S7 连接。

### 本地服务器连接

```
<connectionobject>:= "@LOCALSERVER"
```

名为“@LOCALSERVER”的虚拟本地服务器连接下具有数据变量，可用于访问由服务器管理的本地数据块。安装后，仅数据块 DB1 可用。

数据变量的语法与数据块 S7 变量的语法相同，例如：

```
@LOCALSERVER.DB1.100,B
```

后文将介绍数据块的 S7 变量。

不存在基于正常 S7 连接找到的其它数据变量和方法，如状态路径、缓冲区或域服务。

#### 说明

保留连接名称“@LOCALSERVER”，不得将其用作 S7 连接的名称。

### 2.7.6.1 连接名称

#### S7 连接的连接名称

连接名称是在 STEP 7 或 COML S7 中组态的名称，用于标识连接。在 STEP 7 中，此名称称为“本地 ID”。在 OPC 服务器内，本地 ID 是唯一的。

#### 连接类型

OPC 服务器支持以下连接类型：

- S7 连接
- 容错 S7 连接

#### S7 连接名称允许使用哪些字符？

对于 <connectionname>，可使用数字“0-9”、大小写字母字符“A-z”和特殊字符“\_-+()”。连接名称长度可达 24 个字符。名称不区分大小写。

不允许使用其它可打印的字符。

连接名称“SYSTEM”和“@LOCALSERVER”已保留，不得使用。

#### 连接名称示例：

典型示例包括：

- S7\_connection\_1
- S7 OPC connection

## 2.7.7 生产 S7 连接对象的结构和函数

### 什么是 S7 连接对象？

所有生产的协议特定对象都始终要分配到连接。在 S7 中，这些是与通信伙伴间的连接（S7 连接）。系统连接和演示连接是例外情况。

#### 2.7.7.1 S7 连接对象的类型定义

##### S7 连接对象的类型定义

定义了可通过生产 S7 连接使用的对象和功能的特定 OPC UA 对象类型：

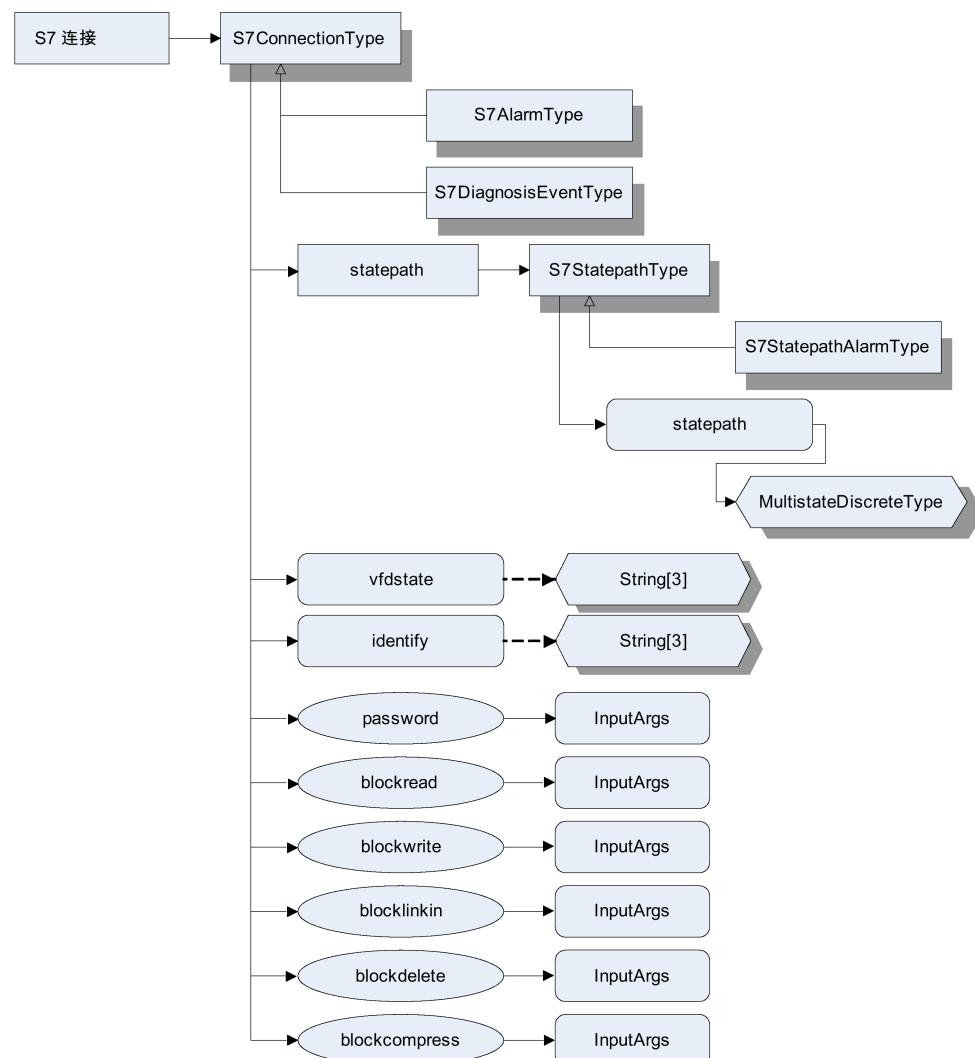


图 2-26 OPC UA 命名空间中 S7 连接对象的类型

针对对象的 OPC UA 命名空间内显示了此类型的实例。  
能在“类型”下构造化读取类型本身。

### 2.7.7.2 S7 连接信息对象

#### 特定于 S7 的数据信息变量

提供特定于 S7 的数据变量，可通过这些变量获得有关 S7 通信和已建立连接的信息。

可获取以下信息：

- 虚拟现场设备 (VFD) 的属性
- S7 连接的状态
- 虚拟设备的状态
- 登陆消息的状态

(请参见“诊断和组态信息 (页 219)”部分中的“事件”参数。)

#### S7 特定的信息变量的语法

Nodeld:

命名空间索引：3 // 针对 Siemens.Automation.SimaticNET.S7  
`<connectionobject>.<informationparameter>`

#### 说明

`<informationparameter>:= "identify"|"vfdstate"|"statepath"`

<code>identify</code>	连接伙伴的供应商属性。	
	数据类型“字符串数组”(3 个数组元素)，只读。	
	“identify”可以返回如下值：	
	供应商	Siemens AG
	型号	6ES7 416-3XR05-0AB0
	版本	V5.0

vfdstate	虚拟设备的状态 数据类型“字符串数组”（3 个数组元素），只读。	
	逻辑状态	S7_STATE_CHANGES_ALLOWED 允许所有服务。
	物理状态	S7_OPERATIONAL 可使用实际设备。 S7_NEEDS_COMMISSIONING 本地设置完成后才能使用实际设备。
	VFD 状态的详细信息， 与设备相关	<bytestring(3)> (3 字节数组) 以八位位组串形式返回状态。 有关返回值含义的详细信息，请参见伙伴设备的文档。
statepath	与伙伴设备之间的通信连接状态 该变量的值以数字形式读出，能通过额外读取相关 Enumstring {UNKNOWN, DOWN, UP, RECOVERY, ESTABLISH} 将其分配到文本。 UA 类型 MultistateDiscreteType 的变量，只读	
	0	UNKNOWN 为将来的扩展保留
	1	DOWN 未建立连接
	2	UP 已建立连接
	3	RECOVERY 未建立连接。正在尝试建立连接。
	4	ESTABLISH 为将来的扩展保留

### 2.7.7.3 S7 特定的信息变量和返回值的示例

下面几个示例说明了 S7 特定信息变量名称的语法。

#### 有关虚拟设备的供应商属性的信息

- Nodeld:
  - 命名空间 URI:  
*S7: (命名空间索引: 3) // 针对 Siemens.Automation.SimaticNET.S7*
  - 标识符:  
*S7\_connection\_1.identify*

可能的返回值:

- 供应商: *SIEMENS AG*
- 虚拟设备的型号: *6ES7413-1AE0-0AB0*
- 版本: *V1.0*

设备的状态

- Nodeld:

- 命名空间 URI:  
*S7*: (命名空间索引: 3) // 针对 *Siemens.Automation.SimaticNET.S7*
- 标识符:  
*S7\_connection\_1.vfdstate*

可能的返回值:

- 逻辑状态: *S7\_STATE\_CHANGES\_ALLOWED*  
允许所有服务。
- 物理状态: *S7\_OPERATIONAL*  
可使用实际设备。
- 详细信息: *02.00.00*  
本地 VFD 状态的详细信息

字符串形式的通信连接的状态

- Nodeld:

- 命名空间 URI:  
*S7*: (命名空间索引: 3) // 针对 *Siemens.Automation.SimaticNET.S7*
- 标识符:  
*S7\_connection\_1.statepath*

类型“MultistateDiscreteType”的可能返回值:

- 连接状态: *RECOVERY*  
当前正在建立连接。

## 2.7.7.4 S7 块服务的方法

### 块服务的用途是什么？

块管理服务可控制 PC 和可编程控制器间的数据及程序元素的传送和返回。在 PC 上，数据和程序元素存储在文件里。可通过密码保护块管理服务。

### 块

数据和程序元素以块的形式存储在 S7 可编程控制器上。这些块使用 STEP 7 进行编译，并被传送到 S7 设备。S7 设备上存在以下块：

- 组织块 (OB)
- 函数块 (FB)
- 函数 (FC)
- 数据块 (DB/DI)

### 块管理函数

可通过 SIMATIC NET OPC 服务器执行下列功能：

- 在 PC 和可编程控制器之间传送块
- 删除块
- 链接块
- 压缩可编程控制器的存储器

### 说明

无法使用 OPC 服务器创建块。需要对此使用 STEP 7。

OPC UA 支持产品特定的方法。S7 块服务与执行方法相同。

### S7 块服务方法的语法

**Nodeld:**

命名空间 *URI*: *S7*: (命名空间索引: 2)  
针对 Siemens.Automation.SimaticNET.S7

*<connectionname>. <DomainMethod>*

**<DomainMethod>:=**

"password()"|"blockread()"|"blockwrite()"|"blocklinkin()"|"blockdelete()"|"blockcompress()"

参数	含义	
password()	将域服务密码传递至伙伴设备 输入参数 1: “密码”; 数据类型: ByteString; 只写 要将密码传递至连接伙伴, 必须将合适的字符串作为密码参数进行传送。 有两种可能的字符串	
	"" (空字符串)	重置密码
	<bytestring(8)>	以字节字符串表示密码 (8 字节数组)
blockread()	将可编程控制器中的块传送到 PC 并存储在文件中: 输入参数 1: “flags”, 数据类型 UInt32 输入参数 2: “block”, 数据类型 String 输入参数 3: “filename”; 数据类型: 字符串	
	标记 <unsigned32>	标记 可能有以下十六进制数字: 0x0001 读取未链接数据块。如果存在目标文件, 不会将其覆盖。 输出一条错误消息。 0x0040 读取已链接数据块。如果存在目标文件, 不会将其覆盖。 输出一条错误消息。 0x1001 读取未链接数据块。覆盖现有目标文件。 0x1040 读取已链接数据块。覆盖现有目标文件。
	块类型和编号: "OB"<unsigned16> "FB"<unsigned16>  "FC"<unsigned16> "DB"<unsigned16>	
	用于存储块的文件的完整路径: <string(511)>	

参数	含义		
blockwrite()	<p>块从 PC 传送到可编程控制器。</p> <p>传送完成后，可编程控制器中的块仍处于被动（未链接）状态。必须使用 <code>&amp;blocklinkin</code> 函数将块由被动状态更改为已链接状态。</p> <p>输入参数 1: “flags”，数据类型 UInt32</p> <p>输入参数 2: “filename”，数据类型: String</p>		
	<table border="1"> <tr> <td>标记 <code>&lt;unsigned32&gt;</code></td><td>标记 可能有以下十六进制数字: 0x1000 将覆盖自动化系统中的同名未链接块。 0x0000 不覆盖自动化系统中的同名未链接块。</td></tr> </table>	标记 <code>&lt;unsigned32&gt;</code>	标记 可能有以下十六进制数字: 0x1000 将覆盖自动化系统中的同名未链接块。 0x0000 不覆盖自动化系统中的同名未链接块。
标记 <code>&lt;unsigned32&gt;</code>	标记 可能有以下十六进制数字: 0x1000 将覆盖自动化系统中的同名未链接块。 0x0000 不覆盖自动化系统中的同名未链接块。		
	<p>用于存储块的文件的完整路径。</p> <p>数据类型 <code>&lt;string(511)&gt;</code></p>		
blocklinkin()	<p>将处于被动状态的块链接到自动化系统的程序顺序中。</p> <p>将块的可执行部分复制到可编程控制器的工作存储器。</p> <p>这样自动化系统的程序就可访问块了。</p> <p>通过链接块，就可在不出现错误消息的情况下覆盖一个现有的激活块。</p> <p>写入值包含作为此服务的参数设置而链接的块的规范。</p> <p>输入参数 1: “block”，数据类型: String</p>		
	<p>块类型和编号:</p> <p>"OB"&lt;unsigned16&gt; "FB"&lt;unsigned16&gt;  "FC"&lt;unsigned16&gt; "DB"&lt;unsigned16&gt;</p>		

参数	含义	
blockdelete()	<p>删除可编程控制器上的块</p> <p>输入参数 1: “flags”，数据类型 UInt32</p> <p>输入参数 2: “block”，数据类型: String</p>	
	<p>标记</p> <p>数据类型 VT_BSTR</p> <p>&lt;unsigned32&gt;</p>	<p>标记</p> <p>可能有以下十六进制数字:</p> <p>0x0001 删除未链接数据块。</p> <p>0x0040  删除已链接数据块。</p> <p>0x0041 删除已链接和未链接数据块。</p>
	<p>块类型和编号</p> <p>数据类型 VT_BSTR</p> <p>"OB"&lt;unsigned16&gt; "FB"&lt;unsigned16&gt;  "FC"&lt;unsigned16&gt; "DB"&lt;unsigned16&gt;</p>	
blockcompress()	<p>压缩可编程控制器的存储器</p> <p>"" (空字符串)</p>	

### 说明

“blockread”和“blockwrite”块服务不允许访问网络驱动器上的块的源文件和目标文件。  
只能在链接可执行部分后才能覆盖 (0x1000) 块。

### 密码

在组态过程中，密码可保护块管理服务对 CPU 的读访问和写访问。与 CPU 的按键开关相比密码有更高的优先级。

有关密码和保护级别的更多信息，请参见“密码 (页 168)”部分。

## 2.7.8 变量服务

### 2.7.8.1 变量服务

**变量服务的用途是什么？**

利用变量服务，可以访问并监视可编程控制器中的 S7 变量。

使用已寻址对象的短名称寻址 S7 变量。访问类型取决于 S7 工具的表示法。

S7 OPC UA 服务器的可编程控制器中的对象支持以下对象：

- 数据块
- 背景数据块
- 输入
- 输出
- I/O 输入
- I/O 输出
- 位存储器
- 定时器
- 计数器

并不是每种 S7 可编程控制器都支持全部对象类型。

### 2.7.8.2 变量服务的语法

**过程变量的语法**

S7 OPC UA Node ID 过程变量的简化语法：

命名空间 URI: S7: (命名空间索引: 3)

**典型语法**

有三个选项：

- <连接名称>. <S7 对象>. <地址>{, <S7 类型>{, <数量>} }
- <连接名称>. <S7 时间对象>. <地址>{, <S7 时间类型>{, <数量>} }
- <连接名称>. <S7 计数器对象>. <地址>{, <S7 计数器类型>{, <数量>} }

## 说明

命名空间 URI: S7: (命名空间索引: 3)

### <connectionname>

协议特定的连接名称。连接名称在组态中指定。

其后的分隔符为句点 (\*.\*)。

### <S7object>

指定 S7 PLC 上的区域类型。可能值如下:

参数	含义
db<no>	数据块或实例数据块。数据块中 S7 变量的标识符。在 S7 通信中，不区分实例数据块和正常数据块。 因此，无需为省事而分配附加标识符。 <b>&lt;编号&gt;</b> 数据块或实例数据块的编号，无前导零。
m	位存储器
i	输入 可读写 为简化起见，仅使用英语标识符。
q	输出 可读写 为简化起见，仅使用英语标识符。
pi	I/O 输入 只读 为简化起见，仅使用英语标识符。
pq	I/O 输出 只写 为简化起见，仅使用英语标识符。

其后的分隔符为句点 (\*.\*)。

### <S7Timerobject>

参数	含义
t	定时器。字（无符号）。 后面的地址信息为定时器编号。

### <S7Counterobject>

参数	含义
c	计数器。字（无符号）。 后面的地址信息是计数器编号。

### <address>

要寻址的第一个变量的字节地址。不支持前导零（如 001）。

字节地址的取值范围为 0 至

65534。根据所使用设备及类型的不同，地址的实际可用值可能会较低。

### <S7type>

S7 数据类型

S7 数据类型转换成 OPC 服务器中的相应 OPC UA 数据类型。

下表列出了类型“标识符”以及可用于表示变量值的相应 OPC UA 数据类型。

S7 数据类型 <S7type>	OPC UA 数据类型	说明
b	字节	字节（无符号） 如果未指定 <S7type>，则用作默认值。
w	UInt16	字（无符号）
c	SByte	字节（有符号）
i	Int16	字（有符号）
di	Int32	双字（有符号）
dw	UInt32	双字（无符号）

## 2.7 通过 OPC UA 进行的 S7 通信

S7 数据类型 <S7type>	OPC UA 数据类型	说明
r	Float	浮点 (4 字节)
dt	DateTime	日期和时间，取值范围自 01.01.1990 起，(在 CPU DATE_AND_TIME 上)
日期	DateTime	日期和时间 (8 字节)，时间始终为 00:00:00，值范围从 1990 年 1 月 1 日开始。 CPU 数据类型 DATE 映射 (无符号，16 位)。
t	Int32	有符号时间值，单位为毫秒
tod	Int32	时钟，从午夜开始，0 到 86399999 ms
s5bcd	UInt16	CPU 数据类型 S5TIME 映射到 UInt 16 (无符号，16 位)，取值范围限制为 0 至 9990000 ms。*)
x<bitaddress>	布尔	位 (布尔) 除区域内的字节偏移量外，还必须在字节中指定 <位地址>。 值范围是 0 到 7
s<stringlength>	String	还必须指定为字符串保留的 <stringlength>。 值范围是 1 到 254 写入时还可写入较短的字符串，使传送的数据长度始终为保留的字符串长度 (字节) 加 2 个字节。不必要的字节用值 0 填充。 读写字符串和字符串数组在内部映射到读写字节数组。 字符串必须以 S7 上的有效值初始化。

\*) 请参见下表“S7 数据类型 s5bcd 的时间基准和取值范围”

数据类型“c”、“i”、“di”和“r”可在数据块 (db) 或实例数据块中用于位存储器 (m)、输入 (i)、输出 (q)、I/O 输入 (pi) 和 I/O 输出 (pq)。

**S7 数据类型 s5bcd 的时间基准和值范围:**

s5bcd 数据类型的时间变量值范围采用 BCD 编码。值范围请参见下表:

位号	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
含义 (符号)	0	0	x	x	t	t	t	t	t	t	t	t	t	t	t	t
说明:																
“0”的含义	不相关															
“x”的含义	指定时间基准															
	位 13 和 12								时间基准 (秒)							
	00								0.01							
	01								0.1							
	10								1							
	11								10							
“t”的含义	BCD 编码的时间值 (0 至 999)															

**示例:**

位号	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
值	0	0	0	1	0	0	0	0	0	1	1	1	0	1	0	1

位 0-11 指定数字 075。位 12 和 13 指定时间基准为 0.1。

$$75 * 0.1 = 0.75 \text{ 秒}$$

时间变量（数据类型 s5bcd）的 OPC 数据类型是字（无符号， UInt16）。

写入时，相应限制值范围。

**<S7Timertype>**

S7 数据类型 <S7type>	OPC UA 数据类型	说明
tbcd	UInt16	定时器，BCD 编码 如果未指定 <S7 定时器类型>，则用作默认值。
tda	UInt16[2]	定时器，十进制时间基准和时间值

## 2.7 通过 OPC UA 进行的 S7 通信

S7 定时器变量“t”和“bcd”（`<S7Timerobject> = t, <S7Timertype> = bcd`）的时间基准和取值范围。

S7 的 OPC 过程变量（类型定时器 (t)）值范围采用 BCD 编码。

时间基准（用于写入）取自下表所示值范围：

位号	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
含义（符号）	0	0	x	x	t	t	t	t	t	t	t	t	t	t	t	t
<b>说明：</b>																
“0”的含义	不相关															
“x”的含义	指定时间基准															
	位 13 和 12								时间基准（秒）							
	00								0.01							
	01								0.1							
	10								1							
	11								10							
“t”的含义	BCD 编码的时间值（0 至 999）															

**示例：**

位号	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
值	0	0	0	1	0	0	0	0	0	1	1	1	0	1	0	1

位 0-11 指定数字 075。位 12 和 13 指定时间基准为 0.1。

$75 * 0.1 = 0.75$  秒

S7 定时器变量“t”和“tda”（`<S7Timerobject> = t, <S7Timertype> = tda`）的时间基准和取值范围。

使用类型“t”的简单定时器变量便于控制定时器。

并非时间基准与取值范围的全部可能组合均可设置，因为取值范围会重叠。

这种情况下可使用“tda”（十进制数组）类型的 S7 定时器变量。

数据类型：由两个字组成的字段 {以毫秒为单位的时间基准 UInt16 | 时间值 UInt16}。

表格 2-3 可能值

时间基准 [ms]	10, 100, 1 000, 10 000
时间值	0...999 允许使用 0 但不起作用。
时间基准 [ms]	10 ms: 0...9 990 100 ms: 0 到 99 900 ms 1 000 ms: 0...99 9000 10,000 ms: 0...9 990 000

TDA 对象无法输入 **<quantity>**。

**示例：**

将值 {100|50} 写入定时器“T.3”，则“tda”以值  $50 * 100 \text{ ms} = 5000 \text{ ms}$  初始化定时器 3，以 100 ms 为基准递减时钟 50 次。

使用类型“t”和“bcd”时无法进行此设置。

**<S7Countertype>**

c	UInt16	计数器 S7 的取值范围：0 到 999，十进制编码 如果未指定 <b>&lt;S7Countertype&gt;</b> ，则用作默认值。
---	--------	--

**<quantity>**

要寻址的某一类型变量的数目，从“address”参数中指定的偏移量开始（取值范围为 1 到 65 535）。

指定多个数组元素将导致形成相应类型的数组，即使只对一个数组元素进行寻址也是如此。

分隔符为逗号（“,”）。

对于数据类型“x”，写访问的数量只能输入 8 的倍数。之后位地址必须为零。

对数据类型“x”，不限制读访问数量的输入。位地址的取值范围之后可为 0 到 7。

TDA 对象无法输入 **<quantity>**。

**示例：**

命名空间 URI: *S7:* (命名空间索引: 3)

*S7-OPC-1.db1.10,x0,64*, 访问权限 RW

*S7-OPC-1.db1.10,x3,17*, 访问权限 R

### 2.7.8.3 S7 OPC UA 变量服务过程变量示例

下面几个示例说明了变量服务的变量名称的语法。

#### 数据块 DB 单字节

命名空间 URI: *S7:* (命名空间索引: 3) // 针对 Siemens.Automation.SimaticNET.S7

*S7\_connection-1.db5.12, B*

通过 *S7\_connection-1* 指示数据块 5 中的数据字节 12。

#### 数据块 DB, 字数组

命名空间 URI: *S7:* (命名空间索引: 3)

*S7\_connection-1.db5.10,w,9*

通过 *S7\_connection-1* 指示数据块 5 中自字节地址 10 起的 9 个数据字。

#### 数据块 DB, 字符串数组

命名空间 URI: *S7:* (命名空间索引: 3)

*S7\_connection-1.db100.50,s32,3*

通过 *S7\_connection-1* 指示实例数据块 100 中自字节地址 50 起长度为 32 的 3 个字符串。

#### 输入 0

命名空间 URI: *S7:* (命名空间索引: 3)

*S7\_connection-1.i.0*

通过 *S7\_connection-1* 指示输入字节 0。

#### 输出 0 位 0

命名空间 URI: *S7:* (命名空间索引: 3)

*S7-connection-1.q.0,x0*

通过 S7\_connection-1 指示输出地址 0, 位 0。

**可读存储器位数组**

命名空间 URI: S7: (命名空间索引: 3)

*S7-connection-1.m.3,x4,12 access rights R*

通过 S7\_connection-1 指示自位存储器地址 3 起的 12 位以及自位地址 4 起的 12 位。只读。

**定时器 22 BCD 编码**

命名空间 URI: S7: (命名空间索引: 3)

*S7\_connection-1.t.22*

通过 S7\_connection-1 指示定时器 22, TBCD 默认值。

**2.7.9 面向缓冲区的服务****面向缓冲区的服务的用途是什么？**

利用面向缓冲区的服务，可通过程序控制较大数据块的传送。以变量为基础执行传送：

- 发送数据块的变量
- 接收数据块的变量

最大传送数据量为 65534 字节，与 PDU 大小无关。数据的分段由函数本身来处理。

**说明**

面向缓冲区的服务只能用于两端已组态的连接。建立的组态连接必须加载到 S7 可编程控制器。

**2.7.9.1 面向缓冲区的服务的语法****简化语法**

过程变量 S7 OPC UA NodId 的简化语法：

命名空间 URI: *S7:* (命名空间索引: 2) 针对“Siemens.Automation.SimaticNET.S7”

## 典型语法

可用选项如下:

- <connectionname>.brcv<rid>{.<address>{,{<s7type>}},{<quantity>}}}
- <connectionname>.brcv<rid>{,<address>{,{<s7type>}},{<quantity>}}}
- <连接名称>.bsend<寻址参数 ID>.<缓冲区长度>{.<地址>{,{<s7 类型>}}{,<数量>}}}
- <连接名称>.bsend<寻址参数 ID>.<缓冲区长度>{,<地址>{,{<s7 类型>}}{,<数量>}}}

## 说明

名称变量的参数	含义
connectionname	协议特定的连接名称。连接名称在组态中指定。
brcv	数据类型: 字节数组  brcv 包含从伙伴接收的最后一个数据块。 接收数据的内容和长度由发送伙伴设置。 该变量为只读变量。设置此变量进行监视。这会由 OPC 服务器向 OPC 客户端应用程序发出接收数据块的信号。
bsend	数据类型: 字节数组  BSEND 包含向伙伴设备传送的发送数据块。 只有在写入变量后, 才能将数据块发送至伙伴设备。 通过读访问, 可返回上一成功传送数据块的内容。
<rid>	寻址参数的 ID。此模板针对块对 (BSEND/BRCV) 指定并在连接内唯一定义。  您可通过一个连接发送多个 BSEND 块或接收多个 BRCV 块, 但始终使用不同 ID。相同 ID 可用于其它连接。
<bufferlength>	要发送的数据块的长度 (字节)。 可定义任意数量的不同长度写入缓冲区。 这可实现向伙伴发送不同长度的子区域。

名称变量的参数	含义
<address>, <S7type>	<p>可结构化数据缓冲区。从中，可选择一个或多个子区域。指定数据类型 &lt;S7type&gt; 和 &lt;address&gt;，这两个数据类型对 S7 数据块变量也有效。“变量服务 (页 138)”部分的开头对这两种数据类型进行了介绍。</p> <p>读：</p> <p>由于不需要固定从伙伴接收的数据块的结构和长度，因此不允许在区域外定义和请求变量。</p> <p>如果接收数据时无法再填充相关区域，则变量的数量将发生变化以反映此情况。</p> <p>写：</p> <p>通过对发送到伙伴的数据块子区域的执行写访问，将发送整个数据缓冲区；其长度在 &lt;缓冲区长度&gt; 中指定。如果在 OPC UA 组调用过程中写入多个数据缓冲区的子区域，则仅在数据缓冲区的子区域全部更新后才会发送数据缓冲区。</p> <p>从缓存中填充未指定的子区域。</p>
quantity	<p><b>注意</b></p> <p>在数据缓冲区中，数据类型以 Motorola 格式进行解释，然后转换为 Intel 格式进行写入。</p>

## 读写注意事项

- 可读取 (brcv) 和写入 (bsend) 单个位 (x 格式)。
- 可读写单个位的字段，起始位地址无限制，数据块中可使用任意字段长度。  
示例：S7-OPC-1.brcv1.10.x4,78
- 不同长度、不同寻址参数 ID (RID)  
或不同连接名称的发送或接收数据的变量具有独立的存储区域。
- 为独立存储区创建数据项时，将分配发送数据缓冲区并将其初始化为零。对 BSEND 数据项执行的写入操作将写入内部写入缓冲区，然后进行传送。
- 数据块进行非周期性传送。可以对相同的数据同时执行多个网络作业。  
始终传送完整的发送数据块。

此规则也适用于子元素访问或几个客户端同时写入此数据项的情况。

由多个客户端并行写入同一发送数据块或发送数据块的子区域会造成不一致。

因此我们建议...

- 您始终读取或写入完整的数据块，或者
- 在 NCM S7 中将并行网络作业的最大数目设置为 1。但是，这会降低传送性能。
- 如果使用 OPC UA 监视服务（`brcv` 作为 `MonitoredItem`）读取数据块的接收信息，则可在监视服务的组态中设置对接收数据块的响应。假设将监控服务中的 `DataChangeTrigger` 设置为 `OpcUa_DataChangeTrigger_StatusValueTimestamp`，则新接收到的具有相同数据的数据块时还可以向 OPC UA 客户端发送信号。  
这样，客户端就可以从伙伴接收未更改的数据缓冲区。`DataChange` 通知不及协商的更新速率快。因此对于此功能，应该始终将更新速率设置为比 `bsend/brcv` 数据的发送速率更快。

### 2.7.9.2 面向缓冲区的服务的过程变量示例

下面几个示例说明了面向缓冲区的服务的变量名称的语法。

#### 接收整个缓冲区中的数据块

命名空间 *URI*: *S7*: (命名空间索引: 3)

*S7-OPC-1.brcv1*

通过 *S7-OPC-1* 在带有 RID 1  
的接收缓冲区中接收数据块。将完整缓冲区映射到字节数组。

#### 交替访问所接收的数据块

命名空间 *URI*: *S7*: (命名空间索引: 3)

*S7-OPC-1.brcv1.2,w,4*

从地址 2 开始，将接收的数据块的内容映射到 4 字数组。从而将数据块的总共 8  
字节考虑在内。

#### 传送双字

命名空间 *URI*: *S7*: (命名空间索引: 3)

*S7-OPC-1.bsend1.16,2,d*

从地址 2 开始的双字通过 S7-OPC-1 在长度为 16、RID 1 的数据块中寻址。如果写入命令用于访问变量，则在数据块中指定位置输入写入值，然后发送数据块。

### 传送浮点数

命名空间 URI: S7: (命名空间索引: 3)

S7S7-OPC-1.bsend1.32,20,r,2

从偏移量 20 开始的带有浮点数的字段通过 S7-OPC-1 在长度 32、RID 1 的数据块中寻址。如果写入命令用于访问变量，则在数据块中指定位置输入写入值，然后发送数据块。

## 2.7.10 S7 连接的块信息对象

### 2.7.10.1 长度信息

#### S7 连接对象下的块对象

S7 可编程控制器内块结构的类型和大小在运行系统中进行计算。在 S7 连接对象下，有以下块对象为 OPC-UA 客户端提供此信息。

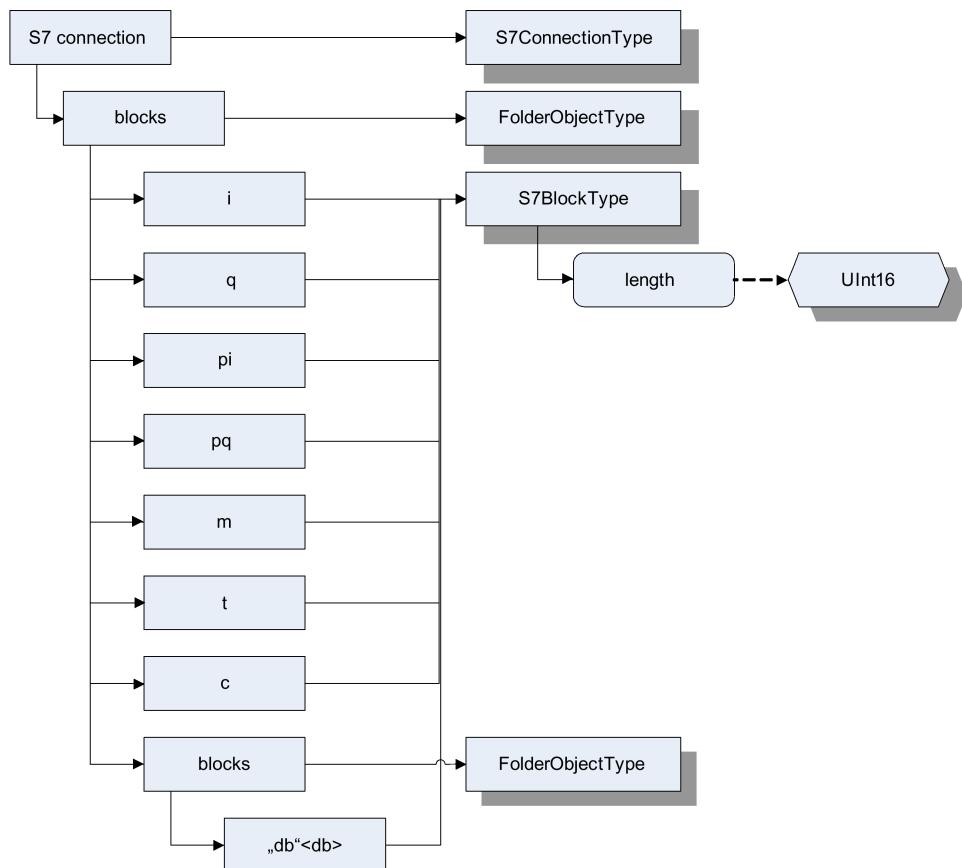


图 2-27 S7 连接对象下的块对象

每个块对象都具有关于块长度/大小信息的 OPC UA 属性。

#### 示例：

命名空间 URI: S7: (→ 命名空间索引: 3)

*S7-connection\_1.db10.length* // 块 DB10 的长度属性 (字节长度)

### 2.7.10.2 模板对象

#### 块对象的模板对象

对于浏览时显示的每个块对象均会显示模板对象，该模板对象的 **NodeId** 可用作进一步用户自定义数据对象的模板。

模板对象具有用于相关块对象的标准数据类型 B（或 c、tbcd），始终从地址 0 开始。如果此项目在连接伙伴上不可访问，访问结果和质量代码将指示此情况。

#### 示例：

命名空间 URI: **S7:** (命名空间索引: 3) 3)

*S7\_connection\_1.db10.0,b*

*S7\_connection\_1.m.0,b*

### 2.7.10.3 诊断和组态信息

#### S7 连接对象的属性

通常，由 STEP 7 组态工具对 S7 连接的属性进行组态。

运行过程中，评估某些组态参数会非常有用。

对 OPC UA 提供了一些组态参数作为 S7 连接对象的属性：

## 2.7 通过 OPC UA 进行的 S7 通信

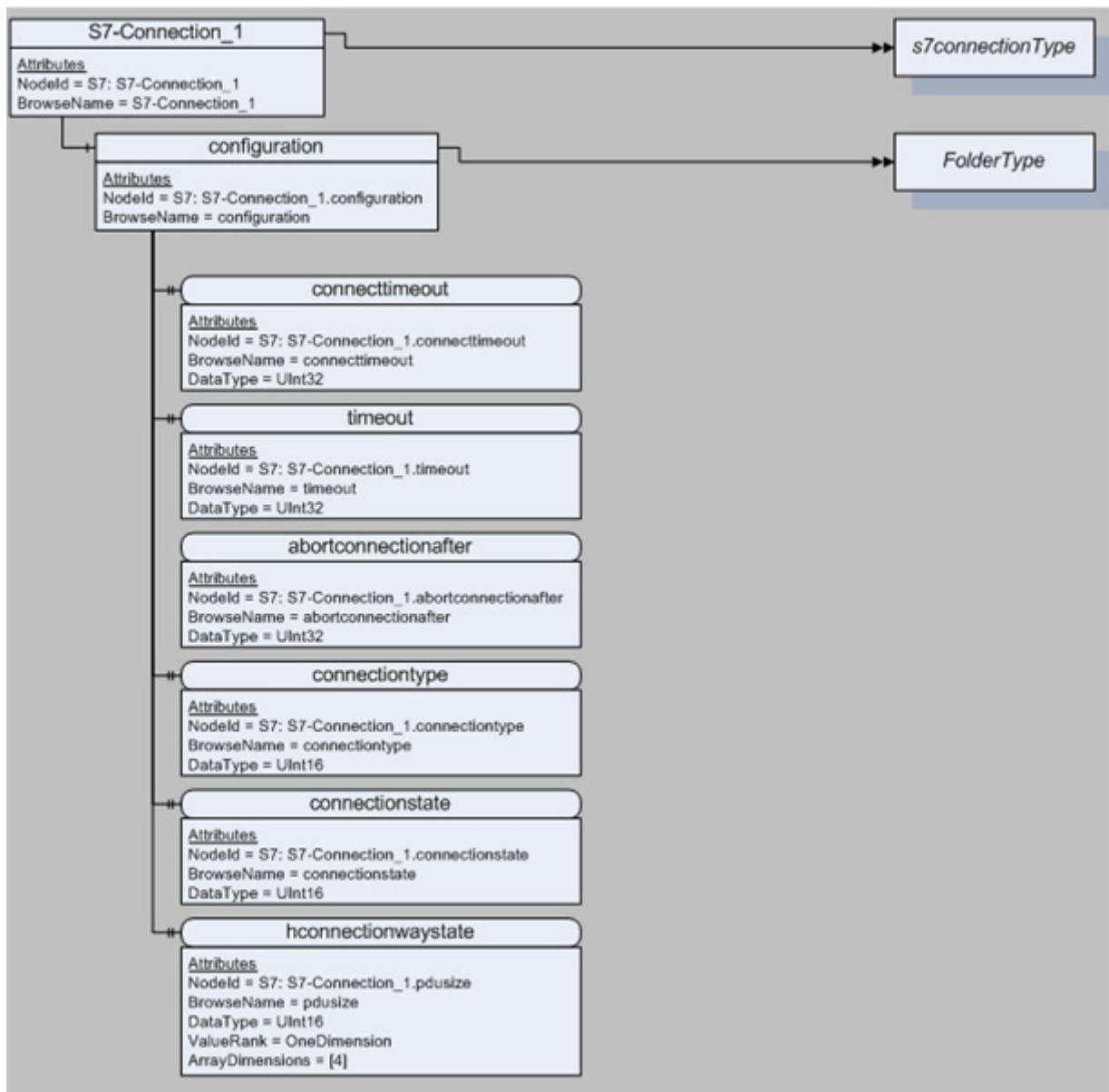


图 2-28 S7 连接对象的属性（第 1 部分）

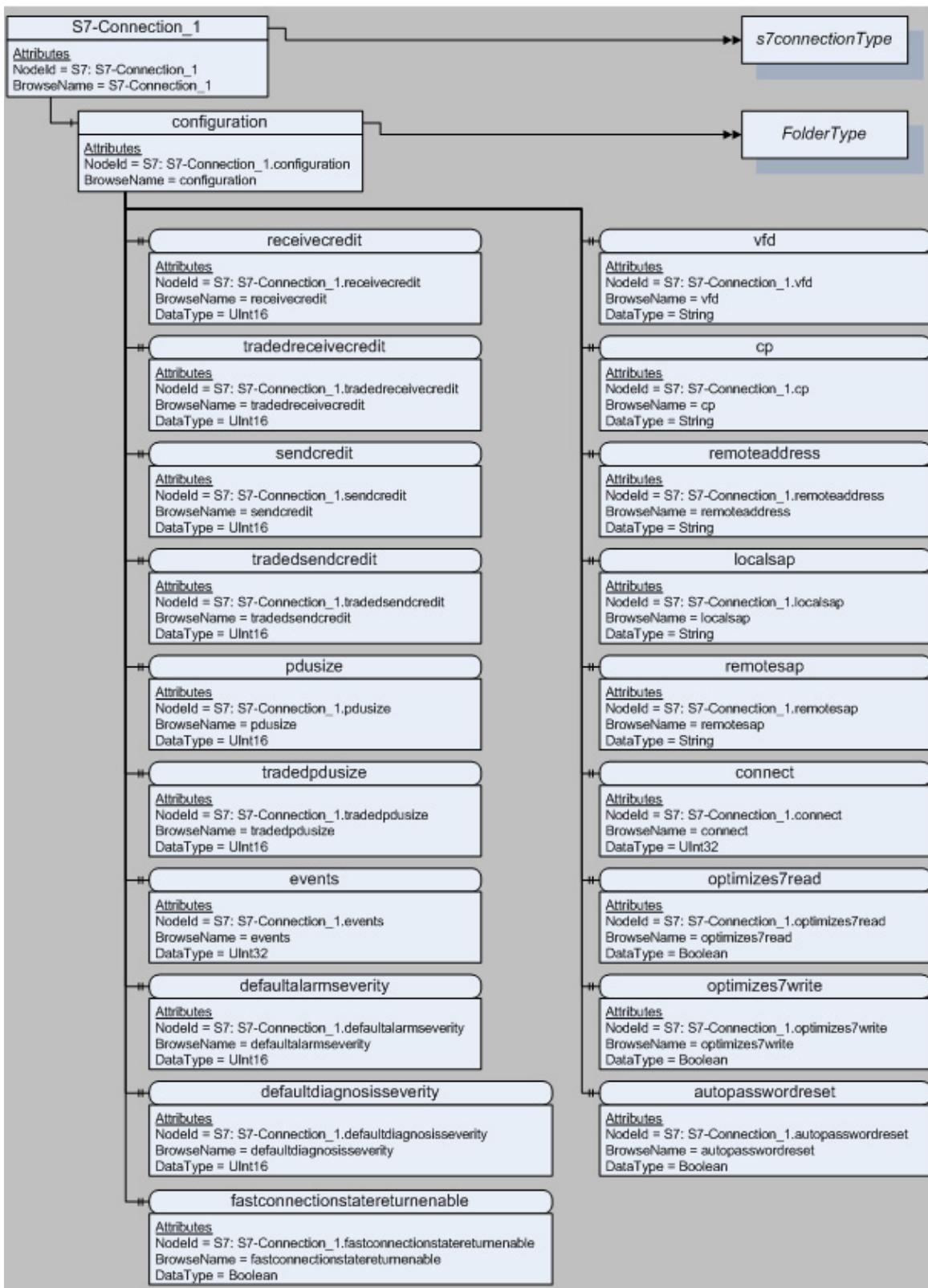


图 2-29 S7 连接对象的属性（第 2 部分）

## 诊断和组态信息的语法

命名空间 URI: S7: (--> 命名空间索引: 3) 3)

<connectionname>. <S7connectionproperty>

<S7connectionproperty>:= "vfd"|"cp"|"remoteaddress"|"localsap"|"remotesap"  
 "connect"|"autopasswordreset"|"fastconnectionstatereturnable"  
 "sendcredit"|"receivecredit"|"pdusize"  
 "tradedsendcredit"|"tradedreceivecredit"|"tradedpdusize"  
 "connecttimeout"|"timeout"|"abortconnectionafter"  
 "optimizes7read"|"optimizes7write"|"defaultalarmseverity"  
 "defaultdiagnosisseverity"|"events"|"connectiontype"  
 "connectionstate"|"hconnectionwaystate"

S7 连接诊断属性	含义
vfd	连接所分配到的 OPC 服务器的名称。对于在 NCM 中组态的连接，此项通常包含文本“OPC 服务器”。 数据类型字符串，只读。
cp	连接所分配到的接口参数分配的名称。 数据类型字符串，只读。
remoteaddress	连接伙伴的地址。 数据类型字符串，只读。 连接伙伴的地址是一个数据缓冲区，其数据长度取决于连接类型。 为使其更便于管理，以供用户进行评估，数据缓冲区以字符串格式显示。 PROFIBUS 地址 格式：“ddd”（1-3 个十进制数） IP 地址 (ISO-on-TCP) 格式：“ddd.ddd.ddd.ddd”（每组 1-3 个十进制数） MAC 地址 (ISO) 格式：“xx-xx-xx-xx-xx-xx”（每组 2 个十六进制数）

S7 连接诊断属性	含义						
localsap	<p>连接的本地 SAP。</p> <p>数据类型字符串，只读。</p> <p>连接伙伴的本地 SAP 是一个数据缓冲区，其数据长度取决于连接类型。为使其更便于管理，以供用户进行评估，数据缓冲区以字符串格式显示。</p> <p>格式：“xx.xx”（每组 2 个十六进制数）</p>						
remotesap	<p>连接的远程 SAP。</p> <p>数据类型字符串，只读。</p> <p>连接伙伴的远程 SAP 是一个数据缓冲区，其数据长度取决于连接类型。为使其更便于管理，以供用户进行评估，数据缓冲区以字符串格式显示。</p> <p>格式：“xx.xx”（每组 2 个十六进制数）</p>						
connect	<p>连接建立的类型。</p> <p>数据类型 UInt32，只读。</p> <table border="1"> <tr> <td>0</td><td>被动，连接永久保持为已建立状态。</td></tr> <tr> <td>1</td><td>主动，仅根据需要建立连接，如果在等待时间过后还不使用，则终止连接。</td></tr> <tr> <td>2</td><td>主动，连接永久保持为已建立状态。</td></tr> </table>	0	被动，连接永久保持为已建立状态。	1	主动，仅根据需要建立连接，如果在等待时间过后还不使用，则终止连接。	2	主动，连接永久保持为已建立状态。
0	被动，连接永久保持为已建立状态。						
1	主动，仅根据需要建立连接，如果在等待时间过后还不使用，则终止连接。						
2	主动，连接永久保持为已建立状态。						
autopasswordreset	<p>自动重设用于块访问的 S7 密码</p> <p>数据类型布尔型，只读。</p> <p><b>True:</b> 启用重设 <b>False:</b> 禁用重设</p> <p>在 S7 上，通过密码启用的域服务保持激活状态直至将其显式重设。</p> <p>在建立连接时自动重设密码可确保密码启用不会占用过多的时间，特别是与不使用连接时的自动终止结合使用的情况下。</p>						
fastconnectionstatereturnenable	<p>如果连接中断，则快速返回写入/读取作业。</p> <p>数据类型布尔型，只读。</p> <p><b>True:</b> 启用 <b>False:</b> 禁用</p>						
sendcredit	<p>在发送方向上同时进行的网络作业的最大数目</p> <p>连接建立的建议值。</p> <p>数据类型 UInt16，只读。</p> <p><b>&gt;=1</b>，连接建立的建议值</p> <p>在组态中与 <b>&amp;receivecredit()</b> 一起设置。</p>						

## 2.7 通过 OPC UA 进行的 S7 通信

S7 连接诊断属性	含义
receivecredit	在接收方向上同时进行的网络作业的最大数目 连接建立的建议值。 数据类型 UInt16, 只读。 <b>&gt;=1</b> , 连接建立的建议值
pdusize	PDU 的大小 连接建立的建议值 数据类型 UInt16, 只读。 <b>&gt;=1</b> , 连接建立的建议值
tradedsendcredit	建立连接后, 在所协商的发送方向上同时进行的协议作业数目。 数据类型 UInt16, 只读。 如果连接中断, 则此属性的质量为“不良”。
tradedreceivecredit	建立连接后, 在所协商的接收方向上同时进行的协议作业数目。 数据类型 UInt16, 只读。 如果连接中断, 则此属性的质量为“不良”。
tradedpdusize	连接建立后所协商的 PDU 大小。 数据类型 UInt16, 只读。 如果连接中断, 则此属性的质量为“不良”。
connecttimeout	连接建立超时 数据类型 UInt32, 只读。 0: 未超时 <b>&gt;0</b> : 超时 (ms)
timeout	生产通信作业超时 (ms)。 数据类型 UInt32, 只读。 0: 未超时 <b>&gt;0</b> : 超时 (ms)

S7 连接诊断属性	含义
abortconnectionafter	<p>自动终止连接。</p> <p>自动终止连接的延迟时间：如果在延迟时间期间没有其它的变量访问操作，则延迟时间结束后，OPC 服务器会自动终止连接。</p> <p>这样可在变量访问间隔时间较长的情况下减少所需的连接数量。</p> <p>数据类型 UInt32，只读。</p> <p>0: 不终止 &gt;0: 终止前的空闲时间 (ms)</p>
optimizes7read	<p>优化对块的 S7 读访问。</p> <p>数据类型布尔型，读写。</p> <p>True: 优化 False: 不优化</p> <p>优化含义： 将各个变量的若干访问作业内部转换为对通信伙伴的单个数组访问。</p>
optimizes7write	<p>优化对块的 S7 写访问。</p> <p>数据类型布尔型，只读。</p> <p>True: 优化 False: 不优化</p> <p>优化含义： 将各个变量的若干访问作业内部转换为对通信伙伴的单个数组访问。</p>
defaultalarmseverity	<p>未组态报警事件的默认优先级。</p> <p>数据类型 UInt16，只读。</p> <p>1: 低优先级 ... 1000: 高优先级</p> <p>在组态中，S7 报警和 S7 诊断报警的默认报警优先级只有一个选项。</p>
defaultdiagnosisseverity	<p>未组态诊断事件的默认优先级。</p> <p>数据类型 UInt16，只读。</p> <p>1: 低优先级 ... 1000: 高优先级</p> <p>在组态中，S7 报警和 S7 诊断报警的默认报警优先级只有一个选项。</p>

## 2.7 通过 OPC UA 进行的 S7 通信

S7 连接诊断属性	含义																
events	<p>在连接伙伴上注册报警和事件。</p> <p>数据类型 UInt32，只读。</p> <p>可组合各个值</p> <table border="1"> <tr> <td>0x00000001</td><td>扫描数据项（不再支持）</td></tr> <tr> <td>0x00000002</td><td>简单报警（不再支持）</td></tr> <tr> <td>0x00000004</td><td>简单符号相关报警（不再支持）</td></tr> <tr> <td>0x00000008</td><td>Simotion TO 报警</td></tr> <tr> <td>0x00000010</td><td>连接监视报警</td></tr> <tr> <td>0x00000020</td><td>块相关报警（作为条件事件）</td></tr> <tr> <td>0x00000040</td><td>符号相关报警（作为条件事件）</td></tr> <tr> <td>0x00000080</td><td>诊断报警</td></tr> </table>	0x00000001	扫描数据项（不再支持）	0x00000002	简单报警（不再支持）	0x00000004	简单符号相关报警（不再支持）	0x00000008	Simotion TO 报警	0x00000010	连接监视报警	0x00000020	块相关报警（作为条件事件）	0x00000040	符号相关报警（作为条件事件）	0x00000080	诊断报警
0x00000001	扫描数据项（不再支持）																
0x00000002	简单报警（不再支持）																
0x00000004	简单符号相关报警（不再支持）																
0x00000008	Simotion TO 报警																
0x00000010	连接监视报警																
0x00000020	块相关报警（作为条件事件）																
0x00000040	符号相关报警（作为条件事件）																
0x00000080	诊断报警																
connectiontype	<p>S7 连接类型</p> <p>数据类型 UInt16，只读。</p> <p>2:S7D_STD_TYPE；标准连接</p> <p>3:S7D_H_TYPE；容错连接</p> <p>如果 S7 连接尚未建立，则此数据项将报告为质量“不良”，且值无效。</p>																

S7 连接诊断属性	含义
connectionstate	<p>S7 连接状态 数据类型 UInt16, 只读。</p> <p>0x11:STD_DOWN; 故意终止标准连接 0x12:STD_ABORT; 无意终止标准连接 (错误) 0x13:STD_NOT_USED; 从未建立标准连接 0x14:STD_OK; 已建立标准连接 0x20:H_OK_RED; 已建立容错连接 (冗余) 0x21:H_OK_RED_PATH_CHG; 已建立容错连接 (冗余, 提供故障切换) 0x22:H_OK_NOT_RED; 非冗余的已建立容错连接 0x23:H_ABORT; 无意终止容错连接 (错误) 0x24:H_NOT_USED; 从未建立容错连接 0x25:H_DOWN; 故意终止容错连接 如果 S7 连接尚未建立, 则此数据项将报告为质量“不良”, 且值无效。</p>
hconnectionwaystate	<p>H 连接路径的状态 数据类型 UInt16 的数组, 4 个数组元素, 只读。</p> <p>0x30:HW_PROD; 路径是生产连接 0x31:HW_STBY; 路径是备用连接 0x32:HW_ABORT; 无意终止路径 (错误) 0x33:HW_NOT_USED; 从未建立路径 0x34:HW_DOWN; 故意终止路径 0x35:HW_CN_BREAK; 无法建立路径 如果 S7 连接尚未建立, 则此数据项将报告为质量“不良”, 且值无效。</p>

## 2.7.11 S7 OPC UA 模板数据变量

通过使用 OPC UA 的 S7

协议的过程变量, 您可拥有灵活的设置选项以所需的数据格式获取设备的过程数据。

不过, 在完全可浏览的命名空间中, 不能将各种寻址选项放在一起。

即使单字节长度的数据块也有大约 40 个不同的数据格式选项 - 从 Byte 和 SByte 开始, 由这些数据类型的一个元素组成的数组、单个位、最多 8 个数组元素的位数组, 每个元素均起始于不同的位偏移。

## 2.7 通过 OPC UA 进行的 S7 通信

因此，OPC UA 服务器支持 S7 命名空间中具有所谓模板数据变量的用户。在典型的 OPC 客户端文本输入框中，只需更改几个字符，就可将这些模板转换为有效的 ItemID。

示例：

```
S7_connection1.db<db>.<o>, dw
```

将 <db> 替换为块数据块编号，将 <o> 替换为数据块中的偏移量，即可获得有效 Nodeld。

```
-> S7_connection1.db10.4, dw
```

另一个示例：

```
S7_connection1.bsend<rid>.<l>,<o>,b,<c>
```

```
-> S7_connection1.bsend43.1000,0,b,100
```

此原理的优点在于，几乎所有 OPC UA 客户端都可以使用，无需对客户端进行任何调整。

### 说明

可以在“通信设置”组态程序中启用和禁用 S7 OPC UA 模块数据变量的可用性：在“OPC 协议选择”(OPC protocol selection) 中单击“S7”旁的箭头符号。

## 浏览层级结构中的模板数据变量

在命名空间表示中相应的文件夹旁将模板数据变量进行排序以便需要时便于使用。

## 模板数据变量部分属性的特殊用法

UA 规范已规定了 OPC UA 属性的用途，无需更多说明。

数据块字节变量模板示例：

Nodeld:            S7\_connection1.db<db>.<o>,d

浏览名称:        模板字节

说明:            <db> 数据块编号的地址

                  <o> 文件内的偏移量

## 2.7.12 事件、条件和报警

### 2.7.12.1 有哪些报警？

本部分介绍 S7 报警的映射以及针对 OPC UA 事件、条件和报警的 S7 诊断事件。

OPC UA 和 S7 有以下事件和报警类型：

- 状态路径报警  
关于 S7 连接状态的报警
- 符号相关报警 (SCAN)  
允许监视与 PLC 用户程序异步的 CPU 中 I、Q、M 和 DB 区域内的位。
- 块相关报警（报警 SFB、报警 SFC）  
可确认的报警 SFB 包括：ALARM (SFB 33)、ALARM\_8 (SFB 34) 和 ALARM\_8P (SFB 35)  
以下 SFB 不可确认：NOTIFY (SFB 36) 和 NOTIFY\_8P (SFB 31)  
可确认的报警 SFC 包括：ALARM\_SQ (SFC 17) 和 ALARM\_DQ (SFC 107)  
以下 SFC 不可确认：ALARM\_S (SFC 18) 和 ALARM\_D (SFC 108)
- 诊断报警  
通过 WR\_USMSG (SFC 52) 的系统诊断 (ID 0x1000-0x79FF、0xC000-0xFFFF、0xF900-0xF9F) 用户诊断 (ID 0x8000-0xB9FF)

### 2.7.12.2 什么是事件？

设备的特点在于其硬件和软件组件的状态。通过 UA

报警，您可选择将从所有状态中选择的状态更改作为事件报告给注册用户。

事件的相关信息存储在其属性中。事件类型定义了事件所具有的属性。

事件类型具有自有的属性或继承自其它事件类型（也可能相互继承属性）的属性。

属性简单继承的选项构成了事件类型体系。UA

报警规范包括以预定义类型体系构建的多种预定义事件类型。UA

报警规范还包括对属性类型和这些属性语义的相关规定。

所有预定义事件均位于命名空间 ns="http://opcfoundation.org/UA/" 中。

### 2.7.12.3 S7 UA 报警服务器的事件有哪些？

S7 UA 报警服务器基于预定义事件类型，从预定义事件类型中派生出自己的事件类型。

为 S7 事件定义单独的事件类型。所有 S7 事件类型均位于命名空间 ns="S7TYPES:" 中。

## 2.7 通过 OPC UA 进行的 S7 通信

S7 UA 报警用来表示 S7 报警。下表显示了 S7 UA 报警服务器可报告的事件类型。除第一种事件类型外，仅在接收到映射 S7 报警后，才会发生指定事件类型的事件。

表格 2-4 S7 UA 报警的事件类型及其用法

S7 事件类型的 NodId	显示名称	通过 OPC 服务器 7.0，在 S7 系统中映射下列对象：	对于 OPC 服务器 8.0，在 S7 系统中映射下列对象：
ns=S7Types, i=14	“S7StatepathAlarmType”	1)	1)
ns=S7Types, i=40	“S7ExclusiveLimitAlarmType”	-	以下报警类别的所有与已组态块和符号相关的报警：“报警 - 高”、“报警 - 低”、“警告 - 高”或“警告 - 低”。
ns=S7Types, i=41	“S7ExclusiveDeviationAlarmType”	-	以下报警类别的所有与已组态块和符号相关的报警：“容差 - 高”或“容差 - 低”。
Ns=S7Types, i=43	“OffNormalAlarmType”	与块和符号相关的报警	<ol style="list-style-type: none"> <li>所有与未组态的块和符号相关的报警以及</li> <li>报警类别为除“容错 - 高”、“容错 - 低”、“报警 - 高”、“报警 - 低”、“警告 - 高”和“警告 - 低”以外的组态报警。</li> </ol>
Ns=S7Types, i=60	“S7DiagnosisEventType”	诊断报警	组态或未组态诊断报警

- 1) 的注意事项：显示名为“S7StatepathAlarmType”的事件类型 ns="S7TYPES:", i=14 会映射 S7 连接的反向状态（建立 S7 连接 ("Up") 为“未激活”；未建立 S7 连接 ("Down") 为“激活”）。状态只在 PC 端获得，因而在 S7 设备没有物理连接时也可进行报告。

### 2.7.12.4 S7 UA 报警服务器的事件类型体系

S7 UA 报警服务器的事件类型体系由标准事件类型体系构成，其中包含一些源于 S7 事件类型的事件类型。

可使用 OPC Scout V10 浏览事件类型体系。但是，浏览时所显示的不是元素的 NodId 而是它们的显示名。

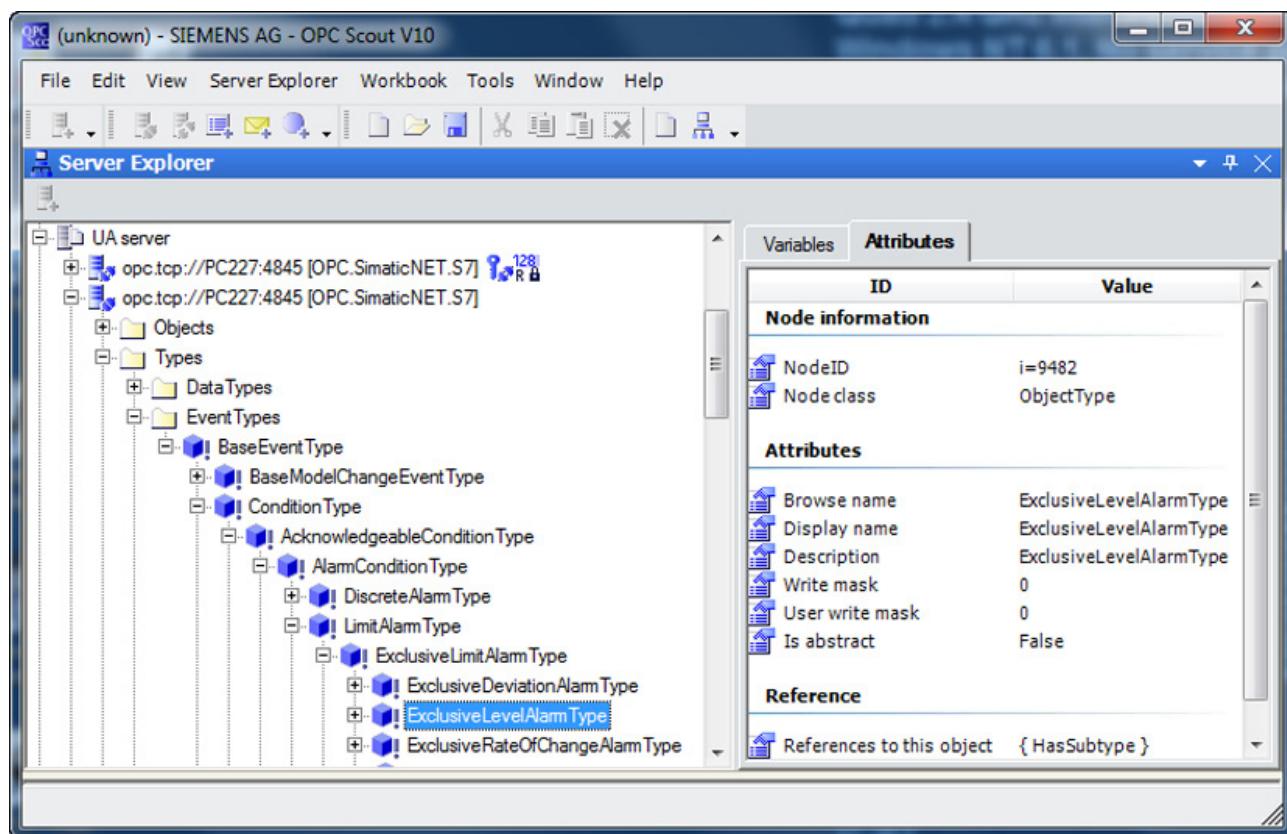


图 2-30 映射事件类型体系

事件类型继承如下：

带显示名的事件类型	继承自...
“ExclusiveLevelAlarmType”	“ExclusiveLimitAlarmType”
“ExclusiveLimitAlarmType”	“LimitAlarmType”
“LimitAlarmType”	“AlarmConditionType”
“AlarmConditionType”	“AcknowledgeableConditionType”
“AcknowledgeableConditionType”	“ConditionType”

显示名为“**BaseEventType**”的事件类型是派生出所有事件类型的源类型。

此类型可定义所有事件及其行为中使用的所有属性。

事件类型是一个数字 **NodeId**（例如 **ns="http://opcfoundation.org/UA/", i=2041;** 显示名 =“**BaseEventType**”）。

## 2.7 通过 OPC UA 进行的 S7 通信

在 S7 UA 报警中，将实例化源自 ConditionType 的事件类型。因此，S7 报警与其所有属性会映射到报警对象：条件实例。  
也可通过“数据访问”读取或监视条件实例的属性。

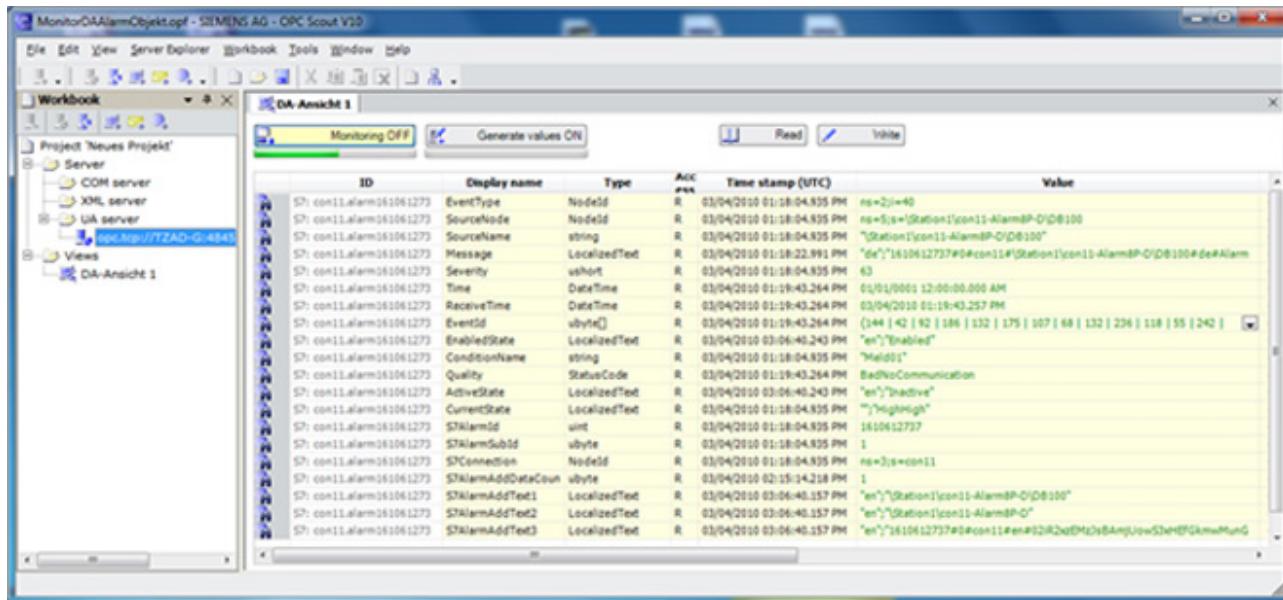


图 2-31 通过数据访问监视报警对象

以下部分介绍了编写 UA 应用程序时与用户相关的属性。

这些属性根据定义它们时使用的事件类型来分组。

引用属性时，将使用数据类型“Qualified Name”。

“Qualified Name”包含命名空间和浏览名称（有时为浏览路径）。

不过，文档的上下文中只引用“Qualified Names”的命名空间与定义类型命名空间相同的属性，因此将忽略命名空间的规范。将指定属性的浏览名称，而非“Qualified Name”。

除此之外，还会在括号中显示请求的属性（用“|”将属性与数据类型分隔开）。

大多数情况下，请求的属性为指定“Value”的数字 13。少数情况下会出现数值

1，该值指定 NodId。

## 2.7.13 标准事件类型

### 2.7.13.1 显示名称为“BaseEventType”的标准事件类型

NodeId: ns="http://opcfoundation.org/UA/", i=2041

源于: 不源于任何其它事件类型。

相关属性的浏览名称

- "EventType" (13|NodeId)
- "SourceNode" (13|NodeId)
- "SourceName" (13|String)
- "Message" (13|LocalizedText)
- "Severity" (13|UInt16)
- "Time" (13|DateTime)
- "ReceiveTime" (13|DateTime)
- "EventId" (13|ByteString)

直接源自该类型的 S7 UA 事件类型: ns="S7TYPES:", i =60, 显示名  
="S7DiagnosisEventType"。无条件实例。此事件类型的事件由源名称唯一标识。

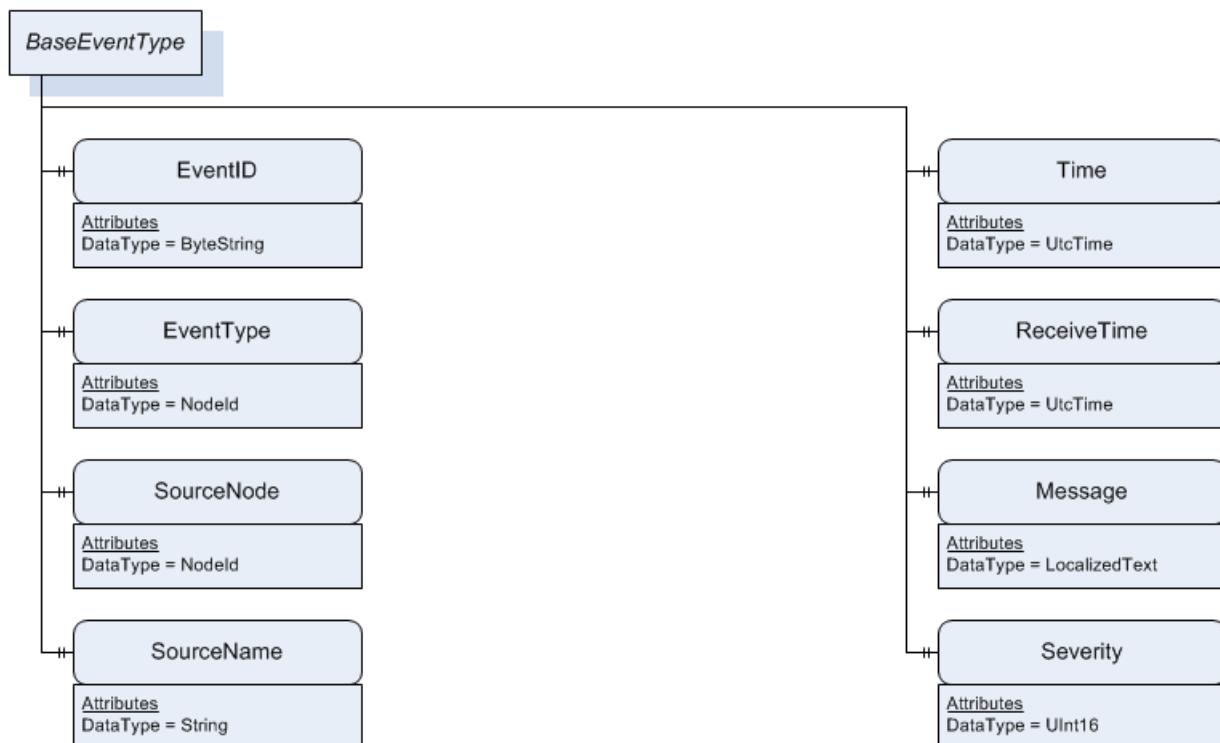


图 2-32 BaseEventType 表示示例

### EventType

始终为之前列出的 S7 事件类型之一。

### SourceNode

事件的源节点；在 S7 UA 报警中，由 ns="S7SOURCES:" 中的 NodId 或 ns="S7:" 中的 S7 连接对象指定对象。另请参见“区域树和源空间 (页 255)”部分。

### SourceName

非本地化源名称。请参见下表“源名称的构成”。

### Message

请参见下表“消息的构成”。

### Severity

有关优先级的信息，请参见下表“严重程度的构成”。

### Time

时间戳，尽可能接近过程。这由组态决定，以下选项可用：

- CPU 时间戳
- CPU 时间 + 偏移量
- PC 时间 (UTC)。

对于 PC 时间 (UTC)，“Time”与“ReceiveTime”相同。

### ReceiveTime

PC 的时间戳。

### EventId

唯一标识和引用事件的标识符 (opaque Id)。客户端需要 EventId 来实现报警确认等操作。

## 源名称的构成

S7 事件类型的 Nodeld	显示名称	OPC 服务器 7.0 的构成规则:	OPC 服务器 8.0 的构成规则:
ns=S7TYPES:, i=14	“S7StatepathAlarmType”	S7 连接名称 +“.statepath”。例如 con13.statepath	S7 连接名称 +“.statepath”。 示例： con13.statepath。 存在相应的 SourceNode。
ns= S7TYPES:, i=40	“S7ExclusiveLimitAlarmType”	S7 连接名 示例： con13	<ol style="list-style-type: none"> <li>未组态 S7 报警: S7 连接名 示例： con13</li> <li>已组态报警： 附加文本 1 存在相应的 SourceNode。</li> <li>无附加文本 1 的已组态报警： 设备路径 +“\”+ 块（或带有 SCAN 的符号名）。示例： Station1\con13-Scan- Noti- fy\S7_program(3)\DB6 04。存在相应的 SourceNode。</li> </ol>
ns= S7TYPES:, i=41	“S7ExclusiveDeviationAlarmType”	S7 连接名示例： con13	<ol style="list-style-type: none"> <li>未组态 S7 报警: S7 连接名 示例： con13</li> <li>已组态报警： 附加文本 1 存在相应的源节点。</li> <li>无附加文本 1 的已组态报警：设备路 径 +“\”+ 块（或带有 SCAN 的符号名）。 示例： Station1\con13- Scan- Noti- fy\S7_program(3)\DB6 04。存在相应的 SourceNode。</li> </ol>

## 2.7 通过 OPC UA 进行的 S7 通信

S7 事件类型的 NodeID	显示名称	OPC 服务器 7.0 的 构成规则:	OPC 服务器 8.0 的 构成规则:
ns= S7TYPES:, i=43	“OffNormalAlarmType”	S7 连接名 示例: con13	<ol style="list-style-type: none"> <li>未组态 S7 报警 S7 连接名 示例: con13</li> <li>已组态报警: 附加文本 1。 存在相应的 SourceNode。</li> <li>无附加文本 1 的已组态报警: 设备路 径 +“\”+ 块 (或带有 SCAN 的符号名)。 示例: Station1\con13- Scan- Noti- fy\S7_program(3)\DB6 04。存在相应的 SourceNode。</li> </ol>
ns= S7TYPES:, i=6 0	“S7DiagnosisEventType”	S7 连接名称 +“.diagnosis”+ 十六进制值 形式报警编号 示例: Con13.diagnosis0 xA001	<ol style="list-style-type: none"> <li>未组态报警: S7 连接名称 +“.diagnosis”+ 十六进制值 形式报警编号 示例: Con13.diagnosi s0xA001</li> <li>已组态报警: 设备路径 +“\”+ 报警标识符 示例: Station1\con13-Scan- Notify\S7_program(3)\ DB604</li> <li>组态报警: 设备路径 +“\”+ 报警标识符。 示例: Station1\con13- Scan- Noti- fy\S7_program(3)\WR_ USMSG (1)。 存在相应的 SourceNode。</li> </ol>

## 构成报警

S7 事件类型的 Nodeld	显示名称	OPC 服务器 7.0 的 构成规则:	OPC 服务器 8.0 的 构成规则:
ns= S7TYPES:, i=14	“S7StatepathAlarmType”	“statepath”	“statepath”
ns= S7TYPES:, i=40	“S7ExclusiveLimitAlarmType”	连接名 +“.alarm”+ 报警编号 示例: con13.alarm1	1. 未组态 S7 报警: S7 连接名称 +“alarm”+ 报警编号。示例: con 13.alarm1 2. 已组态报警: 报警文本
ns= S7TYPES:, i=41	“S7ExclusiveDeviationAlarmType”	连接名 +“.alarm”+ 报警编号 示例: con13.alarm1	1. 未组态 S7 报警: S7 连接名称 +“alarm”+ 报警编号。示例: con 13.alarm1 2. 已组态报警: 报警文本
ns= S7TYPES:, i=43	“OffNormalAlarmType”	连接名 +“.alarm”+ 报警编号 示例: con13.alarm1	1. 未组态 S7 报警: S7 连接名称 +“alarm”+ 报警编号。示例: con 13.alarm1 2. 已组态报警: 报警文本
ns= S7TYPES:, i=60	“S7DiagnosisEventType”	S7 连接名称 +“.diagnosis”+ 十六进制值 示例: con13.diagnosis0xA0 01	1. 未组态报警: S7 连接名称 +“.diagnosis”+ 十六进制值。示例: con13.diagnosis0xA0 01 2. 已组态报警: 报警文本进入状态和 报警文本退出状态

## 2.7 通过 OPC UA 进行的 S7 通信

## 严重程度的构成

S7 事件类型的 Nodeld	显示名称	OPC 服务器 7.0 的 构成规则:	OPC 服务器 8.0 的 构成规则:
ns= S7TYPES:, i=14	“S7StatepathAlarmType”	每个 S7 连接的报警的默认优先级。	每个 S7 连接的报警的默认优先级。
ns= S7TYPES:, i=40	“S7ExclusiveLimitAlarmType”	-	1. 优先级从每一 S7 连接的各个单独报警中检测 (如果存在), 或源自块和 符号的 S7 报警优先级 (0 到 16)。
ns= S7TYPES:, i=41	“S7ExclusiveDeviationAlarmType”	-	1. 优先级从每一 S7 连接的各个单独报警中检测 (如果存在), 或源自块和 符号的 S7 报警优先级 (0 到 16)。
ns= S7TYPES:, i=43	“OffNormalAlarmType”	每个 S7 连接的报警的默认优先级。  根据报警编号检测 的优先级 (如果存 在)。	1. 未组态 S7 报警: 每个 S7 连接的报警的默认优先级, 或通过报警编号分别获取的 优先级。  2. 组态报警: 优先级从每一 S7 连接中检测 (如果存在), 或源自块和符号的 S7 报警优先级 (0 到 16)。
ns= S7TYPES:, i=60	“S7DiagnosisEvent Type”	每个 S7 连接的报警的默认优先级, 或 基于报警编号的优 先级。	1. 每个 S7 连接的报警的默认优先级, 或 2. 基于报警编号的优先级 (如 果存在)。

表格 2- 5 S7 报警优先级严重程度转换表

S7 报警优先级	严重程度
0	1
1	63
2	125
3	188

S7 报警优先级	严重程度
4	250
5	313
6	375
7	438
8	500
9	563
10	625
11	688
12	750
13	813
14	875
15	938
16	1000

### 2.7.13.2 显示名称为“ConditionType”的标准事件类型

Nodeld: ns="http://opcfoundation.org/UA/", i=2782

源自: ns="http://opcfoundation.org/UA", i=2041, 显示名称为“BaseEventType”

相关属性的浏览名称

"" (1|Nodeld)

"ConditionName" (13|String)

"EnableState|ID" (13|Boolean)(Browse-Path)

"EnableState" (13|LocalizedText)

"Quality" (13|StatusCode)

直接源自该类型的 S7 事件类型: 无

直接或间接源自该类型的事件类型都有一个条件实例。

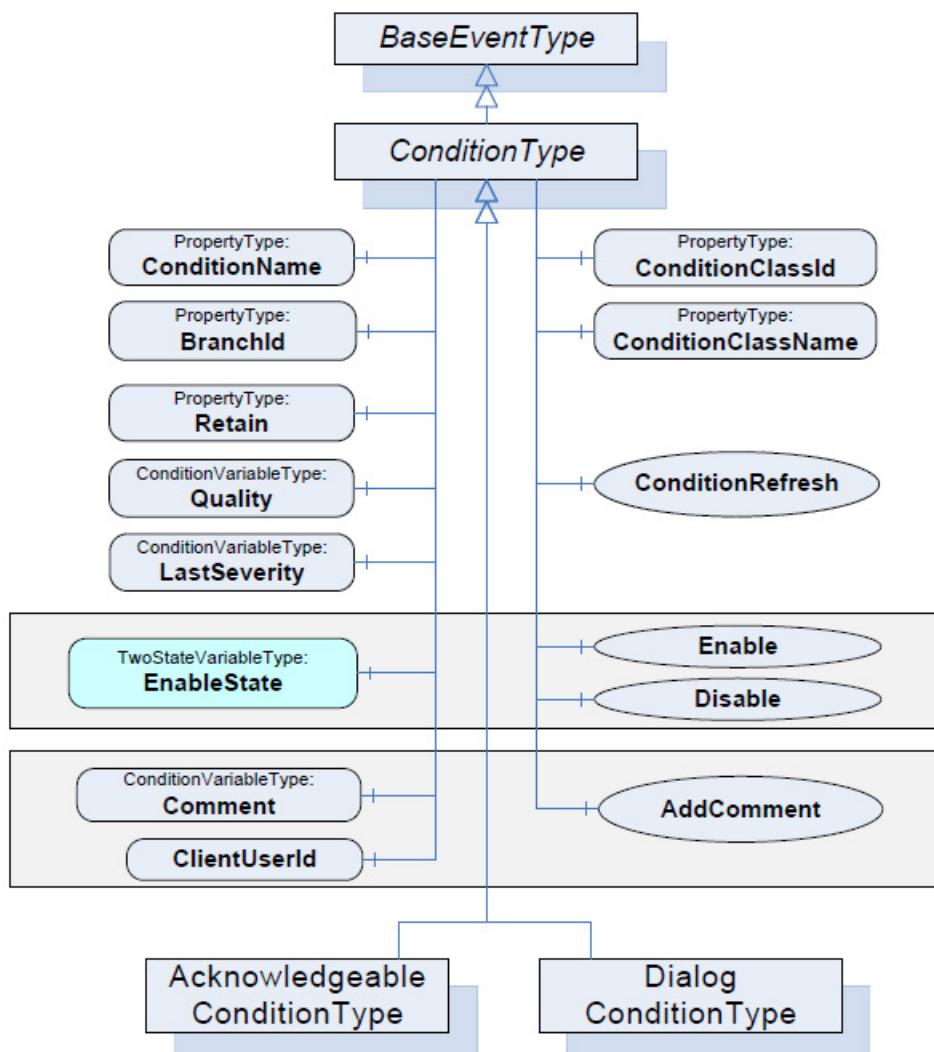


图 2-33 ConditionType 表示示例

## ConditionName

通过“SourceName”和“ConditionName”的组合进行唯一标识的条件事件。

请参见下文中的“构成条件名称”部分。

## EnableState

状态机的一部分。根据 EnableState (LocalizedText) 的 UA 报警规范，可能的值为 "de";"Enabled" 和 "de";"Disabled"。S7 UA 报警不支持禁用，所以 "en";"Enabled" 将始终显示。

## Quality

有关可能的质量值，请参见“UA 规范”。

## 条件名称的构成

S7 事件类型的 Nodeld	显示名称	OPC 服务器 7.0 的 构成规则:	OPC 服务器 8.0 的 构成规则:
ns=S7TYPES:, i=14	“S7StatepathAlarmType”	“statepath”	“statepath”
ns= S7TYPES:, i=40	“S7ExclusiveLimitAlarmType”	“alarm”+ 报警编号。 示例: alarm1	<p>1. 未组态 S7 报警: “alarm”+ 报警编号 示例: alarm1</p> <p>2. 组态报警:  块相关报警: 用 S7_a_type 属性标记的生成功能块中输入变量的名称。信号 2 到 8 后面跟“,”和信号编号。 示例: Alar01, 或 Alar01,</p> <p>符号相关 报警: 符号名称 示例: Scan01</p>

## 2.7 通过 OPC UA 进行的 S7 通信

S7 事件类型的 Nodeld	显示名称	OPC 服务器 7.0 的 构成规则:	OPC 服务器 8.0 的 构成规则:
ns= S7TYPES:, i=41	“S7ExclusiveDeviationAlarmType”	“alarm”+ 报警编号。 示例: alarm1	<p>1. 未组态 S7 报警: “alarm”+ 报警编号 示例: alarm1</p> <p>2. 组态报警: 块相关报警: 用 S7_a_type 属性标记的生成功能块中 输入变量的名称。信号 2 到 8 后面跟“,”和信号编号。 示例: Alar01, 或 Alar01,</p> <p>符号相关 报警: 符号名称 示例: Scan01</p>
ns= S7TYPES:, i=43	“OffNormalAlarmType”	“alarm”+ 报警编号。 示例: alarm1	<p>1. 未组态 S7 报警: “alarm”+ 报警编号 示例: alarm1</p> <p>2. 组态报警: 块相关报警: 用 S7_a_type 属性标记的生成功能块中 输入变量的名称。信号 2 到 8 后面跟“,”和信号编号。 示例: Alar01, 或 Alar01,</p> <p>符号相关 报警: 符号名称 示例: Scan01</p>

### 2.7.13.3 显示名称为“AcknowledgeableConditionType”的标准事件类型

NodeId: ns:"http://opcfoundation.org/UA/", i=2881

源自: ns="http://opcfoundation.org/UA", i=2782, 显示名称为“ConditionType”

相关属性的浏览名称

"AckedState|Id" (13|Boolean)

"AckedState" (13|LocalizedText)

直接源自该类型的 S7 事件类型: 无

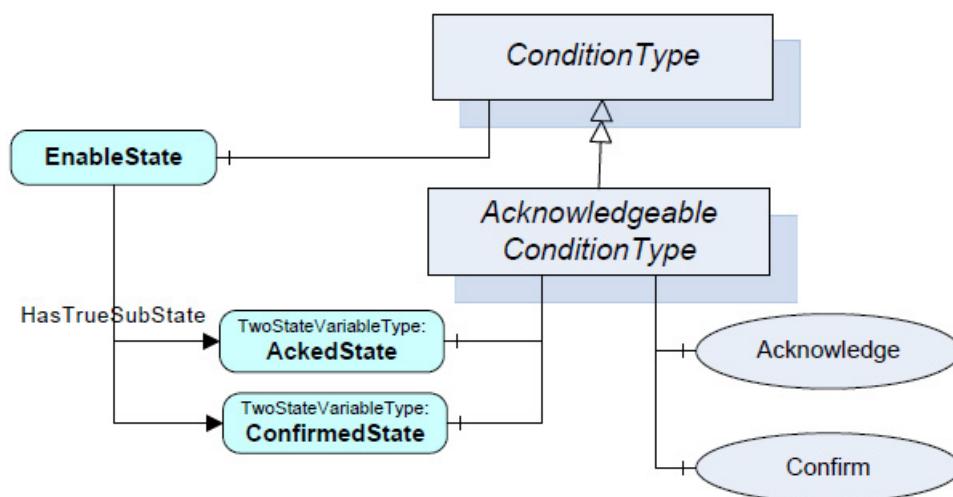


图 2-34 AcknowledgeableConditionType 表示示例

### AckedState

状态机的一部分。可能的值为 "en", Acknowledged" 和 "en", "Unacknowledged"。

AckedState 中的每次值更改都会报告一个事件。报告为

AckedState“zh”,“未确认”的事件都必须使用确认功能确认。请参见 STEP 7 中有关需要确认的块中断的介绍。

### 2.7.13.4 显示名称为“AlarmConditionType”的标准事件类型

NodeId: ns:"http://opcfoundation.org/UA/", i=2915

源自: ns="http://opcfoundation.org/UA", i=2881, 显示名称为

“AcknowledgeableConditionType”

相关属性的浏览名称

"ActiveState|Id" (13|Boolean)

"ActiveState" (13|LocalizedText)

直接源自该类型的 S7 事件类型: 无

根据 ActiveState (LocalizedText) 的 UA 报警规范，可能的值为 "en";"Active" 和 "en";"Inactive"。

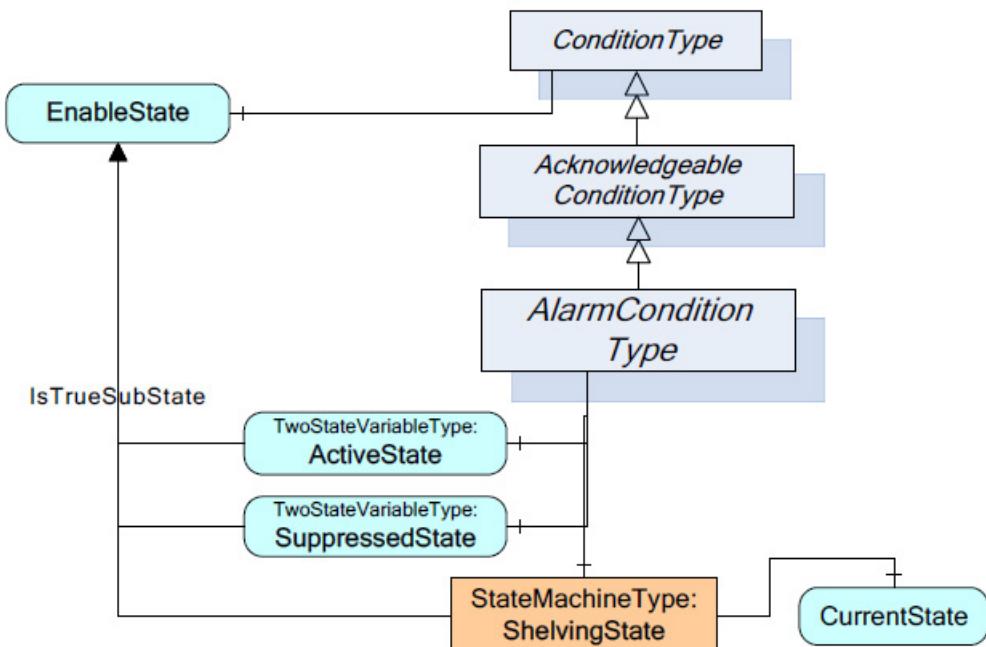


图 2-35 AlarmConditionType 表示示例

## ActiveState

状态机的一部分。在 S7 UA 报警中，ActiveState 由块相关和符号相关报警控制。

如果报警触发信号的值为“1”，将报告

AckedState“激活”，如果值为“0”，将报告“未激活”。ActiveState 中的每次值更改都会报告一个事件。

## 事件类型显示名称为“S7StatepathAlarmType”的 ActiveState

在此，未建立 S7 连接时达到“激活”状态。建立 S7 连接后达到“未激活”状态。

### 2.7.13.5 显示名称为“ExclusiveLimitAlarmType”的标准事件类型

NodeId: ns: "http://opcfoundation.org/UA/", i=9341

间接源自: ns="http://opcfoundation.org/UA/", i=2915, 显示名称为  
“AlarmConditionType”

相关属性的浏览名称:

"LimitState|CurrentState" (13|LocalizedText)

直接源自该类型的 S7 UA 事件类型: 无

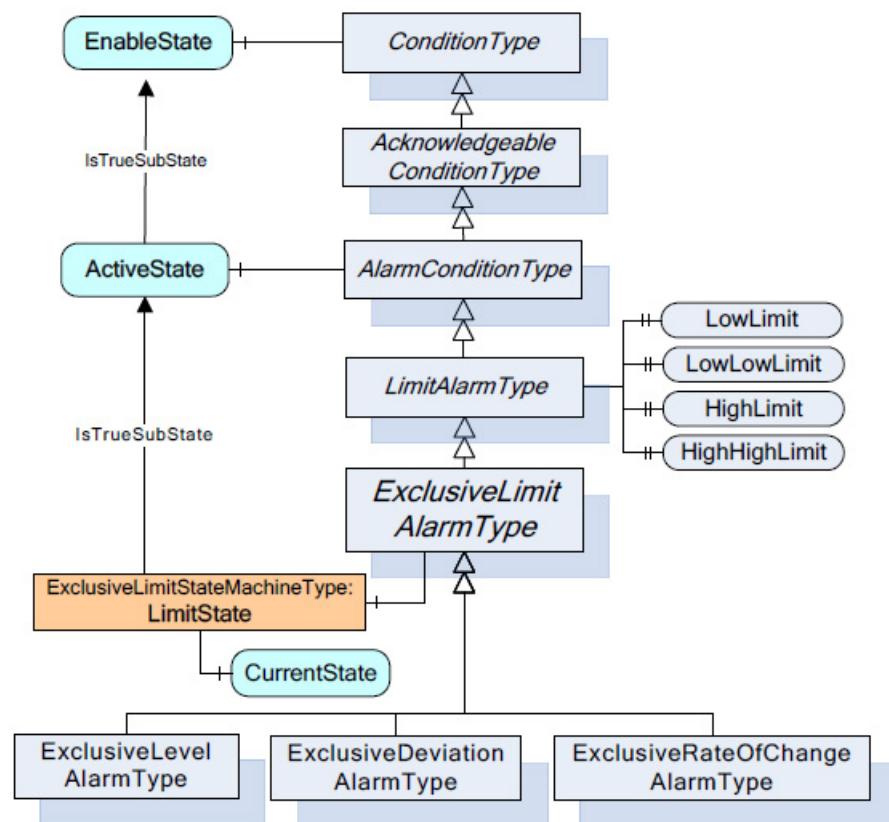


图 2-36 ExclusiveLimitAlarmType 表示示例

#### LimitState|CurrentState

可能的值为“High”、“HighHigh”、“Low”和“LowLow”。

## 2.7 通过 OPC UA 进行的 S7 通信

### 2.7.13.6 显示名称为“ExclusiveLevelAlarmType”的标准事件类型

Nodeld: ns: "http://opcfoundation.org/UA/", i=9482

源自: ns="http://opcfoundation.org/UA/", i=9341, 显示名称为“ExclusiveLimitAlarm Type”

相关属性的浏览名称: 无

直接源自该类型的 S7 UA 事件类型:

ns="S7TYPES:", i=40, 显示名称为“S7ExclusiveLevelAlarm Type”

选择 OPC 服务器 8.0

并在报警的报警类别中做下列设置时, 将生成显示名称为“S7ExclusiveLevelAlarm Type”的事件类型

“报警 - 高” (LimitState|CurrentState="HighHigh") 、

“报警 - 低” (LimitState|CurrentState="LowLow") 、

“警告 - 高” (LimitState|CurrentState="High") 和

“警告 - 低” (LimitState|CurrentState="Low") 。

### 2.7.13.7 显示名称为“ExclusiveDeviationAlarmType”的标准事件类型

Nodeld: ns: "http://opcfoundation.org/UA/", i=9764

源自: ns="http://opcfoundation.org/UA/", i=9341, 显示名称为“ExclusiveLimitAlarm Type”

相关属性的浏览名称: 无

直接源自该类型的 S7 UA 事件类型:

ns=S7Types, i =41, 显示名称为“S7ExclusiveDeviationAlarm Type”

选择 OPC 服务器 8.0 并在报警的报警类别中进行“容差 - 高”或“容差 -

低”设置时, 将生成显示名称为“S7ExclusiveDeviationAlarm Type”的事件类型。

### 2.7.13.8 显示名称为“OffNormalAlarmType”的标准事件类型

Nodeld: ns:"http://opcfoundation.org/UA/", i=10637

间接源自: ns="http://opcfoundation.org/UA/", i=2915, 显示名称为“AlarmCondition Type”

相关属性的浏览名称: 无

直接源自该类型的 S7 UA 事件类型:

ns=S7TYPES:, i = 43, 显示名称为“S7OffNormalAlarm Type”

ns=S7TYPES:, i =14, 显示名称为“S7StatepathAlarm Type”

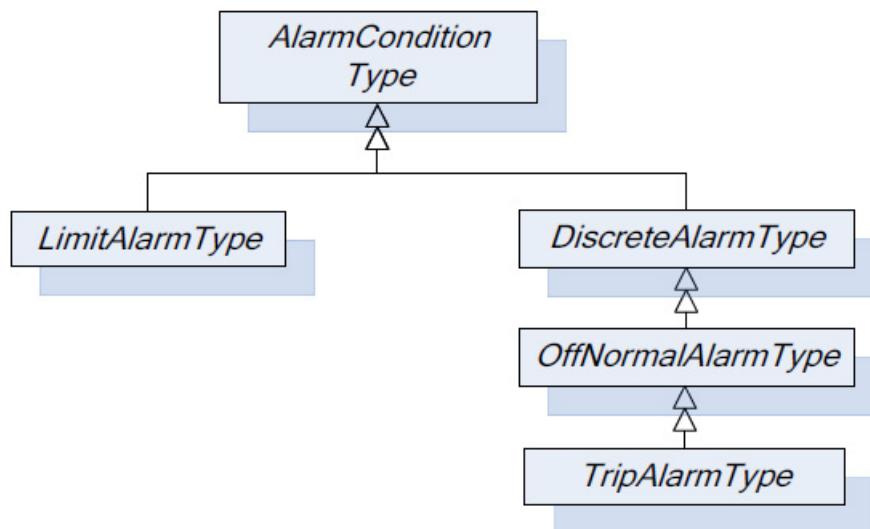


图 2-37 OffNormalAlarm Type 表示示例

选择 OPC 服务器 7.0 或 OPC 服务器 8.0 并且报警中报警类别的设置不是“容差 - 高”、“容差 - 低”、“报警 - 高”、“报警 - 低”、“警告 - 高”和“警告 - 低”时，将生成显示名称为“S7OffNormalAlarm Type”的事件类型。

## 2.7.14 S7 事件类型

### 2.7.14.1 显示名称为“S7StatepathlAlarmType”的 S7 事件类型

NodId: ns="S7TYPES:", i=14

间接源自: ns="http://opcfoundation.org/UA/", i=2915, 显示名称为“AlarmCondition Type”

相关属性的浏览名称:

"S7Connection" (13|NodId)

该事件类型会映射 S7 连接的反向状态（建立 S7 连接 ("Up") 时为“未激活”；未建立 S7 连接 ("Down") 时为“激活”）。状态只在 PC 端获得，因而在 S7 设备没有物理连接时也可进行报告。

#### S7Connection

指定接收 S7 报警的连接的 NodId。

示例: ns="S7:", s="connectionname"。

### 2.7.14.2 显示名称为“S7ExclusiveLevelAlarmType”的 S7 事件类型

Nodeld: ns="S7TYPES:", i=40

间接源自: ns="http://opcfoundation.org/UA/", i=9482, 显示名称为  
“ExclusiveLevelAlarm Type”

相关属性的浏览名称:

"S7AlarmId" (13|UInt32)  
"S7AlarmSubId" (13|Byte)  
"S7Connection" (13|Nodeld)  
"S7Time" (13|DateTime)  
"S7AlarmAddDataCount" (13|Byte)  
"S7AlarmAddData1|DataType" (13|Byte)  
"S7AlarmAddData1|Data" (13|ByteString)  
"S7AlarmAddData2|DataType" (13|Byte)  
"S7AlarmAddData2|Data" (13|ByteString)  
"S7AlarmAddData3|DataType" (13|Byte)  
"S7AlarmAddData3|Data" (13|ByteString)  
"S7AlarmAddData4|DataType" (13|Byte)  
"S7AlarmAddData4|Data" (13|ByteString)  
"S7AlarmAddData5|DataType" (13|Byte)  
"S7AlarmAddData5|Data" (13|ByteString)  
"S7AlarmAddData6|DataType" (13|Byte)  
"S7AlarmAddData6|Data" (13|ByteString)  
"S7AlarmAddData7|DataType" (13|Byte)  
"S7AlarmAddData7|Data" (13|ByteString)  
"S7AlarmAddData8|DataType" (13|Byte)  
"S7AlarmAddData8|Data" (13|ByteString)  
"S7AlarmAddData9|DataType" (13|Byte)  
"S7AlarmAddData9|Data" (13|ByteString)  
"S7AlarmAddData10|DataType3|Byte)  
"S7AlarmAddData10|Data" (13|ByteString)  
"S7AlarmAddText1" (13|LocalizedText)  
"S7AlarmAddText2" (13|LocalizedText)  
"S7AlarmAddText3" (13|LocalizedText)  
"S7AlarmAddText4" (13|LocalizedText)  
"S7AlarmAddText5" (13|LocalizedText)  
"S7AlarmAddText6" (13|LocalizedText)  
"S7AlarmAddText7" (13|LocalizedText)  
"S7AlarmAddText8" (13|LocalizedText)

通过 OPC 服务器 8.0，该 S7 事件类型可映射报警类别为“报警 - 高”、“报警 - 低”、“警告 - 高”或“警告 - 低”的所有已组态块相关和符号相关报警。该 S7 事件类型不能用于 OPC 服务器 7.0。

### S7AlarmId

S7 报警的报警编号。

### S7AlarmSubId

S7 报警的触发信号编号。值范围为 1 到 8。

### S7Connection

指定接收 S7 报警的连接的 NodId。

示例：ns="S7:", s="connectionname"。

### S7AlarmAddData1|Data

生成 S7 报警的关联值。

报警以 ByteString 格式显示。1 <= n <=10

### S7AlarmAddData8|DataType

关联值的 S7 数据类型。因为不能明确映射，所以无法将 S7 数据类型映射到 UA 数据类型。

### S7AlarmAddDataCount

S7 报警的实际关联值数。

### S7AlarmAddTextn

1 <= n < =8。组态中的附加文本。

### 2.7.14.3 显示名称为“S7ExclusiveDeviationAlarmType”的 S7 事件类型

Nodeld: ns="S7TYPES:", i=41

间接源自: ns="http://opcfoundation.org/UA/", i=9764, 显示名称为“ExclusiveDeviation Alarm Type”

相关属性的浏览名称:

- "S7AlarmId" (13|UInt32)
- "S7AlarmSubId" (13|Byte)
- "S7Connection" (13|Nodeld)
- "S7AlarmAddDataCount" (13|Byte)
- "S7Time" (13|DateTime)
- "S7AlarmAddData1|DataType" (13|Byte)
- "S7AlarmAddData1|Data" (13|ByteString)
- "S7AlarmAddData2|DataType" (13|Byte)
- "S7AlarmAddData2|Data" (13|ByteString)
- "S7AlarmAddData3|DataType" (13|Byte)
- "S7AlarmAddData3|Data" (13|ByteString)
- "S7AlarmAddData4|DataType" (13|Byte)
- "S7AlarmAddData4|Data" (13|ByteString)
- "S7AlarmAddData5|DataType" (13|Byte)
- "S7AlarmAddData5|Data" (13|ByteString)
- "S7AlarmAddData6|DataType" (13|Byte)
- "S7AlarmAddData6|Data" (13|ByteString)
- "S7AlarmAddData7|DataType" (13|Byte)
- "S7AlarmAddData7|Data" (13|ByteString)
- "S7AlarmAddData8|DataType" (13|Byte)
- "S7AlarmAddData8|Data" (13|ByteString)
- "S7AlarmAddData9|DataType" (13|Byte)
- "S7AlarmAddData9|Data" (13|ByteString)
- "S7AlarmAddData10|DataType" (13|Byte)
- "S7AlarmAddData10|Data" (13|ByteString)
- "S7AlarmAddText2" (13|LocalizedText)
- "S7AlarmAddText3" (13|LocalizedText)
- "S7AlarmAddText4" (13|LocalizedText)
- "S7AlarmAddText5" (13|LocalizedText)
- "S7AlarmAddText6" (13|LocalizedText)
- "S7AlarmAddText7" (13|LocalizedText)
- "S7AlarmAddText8" (13|LocalizedText)

通过 OPC 服务器 8.0，该 S7 事件类型可映射报警类别为“容差 - 高”或“容差 - 低”的所有已组态块相关和符号相关报警。请参见有关“S7ExclusiveLevelAlarm Type”的注意事项。

#### 2.7.14.4 显示名称为“S7OffNormalAlarmType”的 S7 事件类型

Nodeld: ns="S7TYPES:", i=43

间接源自: ns="http://opcfoundation.org/UA/", i=10637, 显示名称为  
“OffNormalAlarm Type”

相关属性的浏览名称:

"S7AlarmId" (13|UInt32)  
"S7AlarmSubId" (13|Byte)  
"S7Connection" (13|Nodeld)  
"S7AlarmAddDataCount" (13|Byte)  
"S7Time" (13|DateTime)  
"S7AlarmAddData1|DataType" (13|Byte)  
"S7AlarmAddData1|Data" (13|ByteString)  
"S7AlarmAddData2|DataType" (13|Byte)  
"S7AlarmAddData2|Data" (13|ByteString)  
"S7AlarmAddData3|DataType" (13|Byte)  
"S7AlarmAddData3|Data" (13|ByteString)  
"S7AlarmAddData4|DataType" (13|Byte)  
"S7AlarmAddData4|Data" (13|ByteString)  
"S7AlarmAddData5|DataType" (13|Byte)  
"S7AlarmAddData5|Data" (13|ByteString)  
"S7AlarmAddData6|DataType" (13|Byte)  
"S7AlarmAddData6|Data" (13|ByteString)  
"S7AlarmAddData7|DataType" (13|Byte)  
"S7AlarmAddData7|Data" (13|ByteString)  
"S7AlarmAddData8|DataType" (13|Byte)  
"S7AlarmAddData8|Data" (13|ByteString)  
"S7AlarmAddData9|DataType" (13|Byte)  
"S7AlarmAddData9|Data" (13|ByteString)  
"S7AlarmAddData10|DataType" (13|Byte)  
"S7AlarmAddData10|Data" (13|ByteString)  
"S7AlarmAddText2" (13|LocalizedText)  
"S7AlarmAddText3" (13|LocalizedText)  
"S7AlarmAddText4" (13|LocalizedText)  
"S7AlarmAddText5" (13|LocalizedText)  
"S7AlarmAddText6" (13|LocalizedText)  
"S7AlarmAddText7" (13|LocalizedText)  
"S7AlarmAddText8" (13|LocalizedText)

该事件类型可通过 OPC 服务器 7.0 映射块相关和符号相关报警。通过 OPC 服务器 8.0，该事件类型可映射所有未组态的块相关和符号相关报警以及报警类别为“容错 - 高”、“容错 - 低”、“报警 - 高”、“报警 - 低”、“警告 - 高”和“警告 - 低”之外的组态报警。请参见有关“S7ExclusiveLevelAlarm Type”的注意事项。

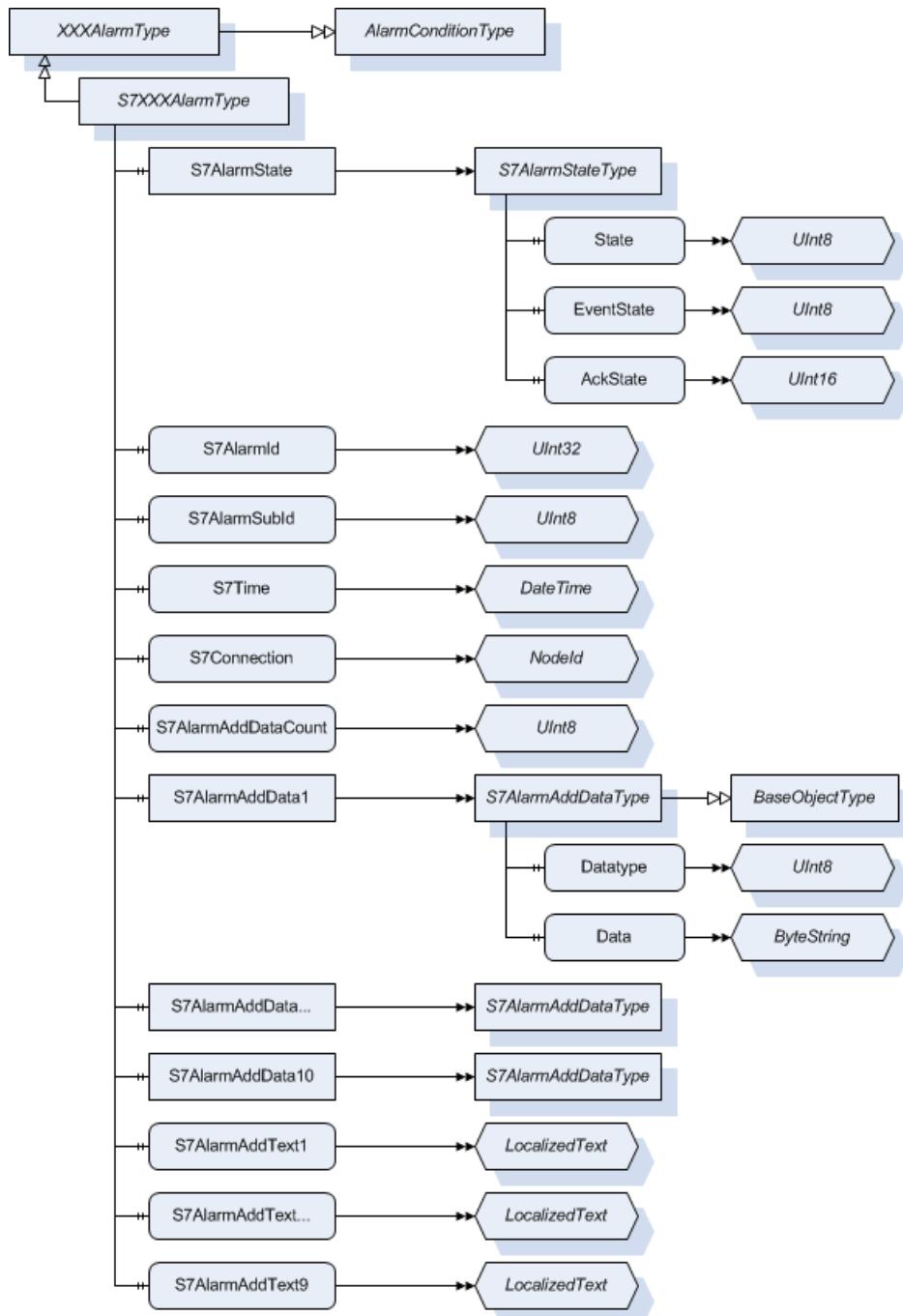


图 2-38 S7OffNormalAlarm 类型和其它 S7XXXAlarm 类型中 S7 特定属性的表示示例

### 2.7.14.5 显示名称为“S7DiagnosisEventType”的 S7 事件类型

Nodeld: ns="S7TYPES:", i=60

间接源自: ns="http://opcfoundation.org/UA/", i=2041, 显示名称为“BaseEventType”

相关属性的浏览名称:

"S7DiagnosisId" (13|UInt32)

"S7Connection" (13|Nodeld)

"S7DiagnosisData" (13|ByteString)

"S7Time" (13|DateTime)

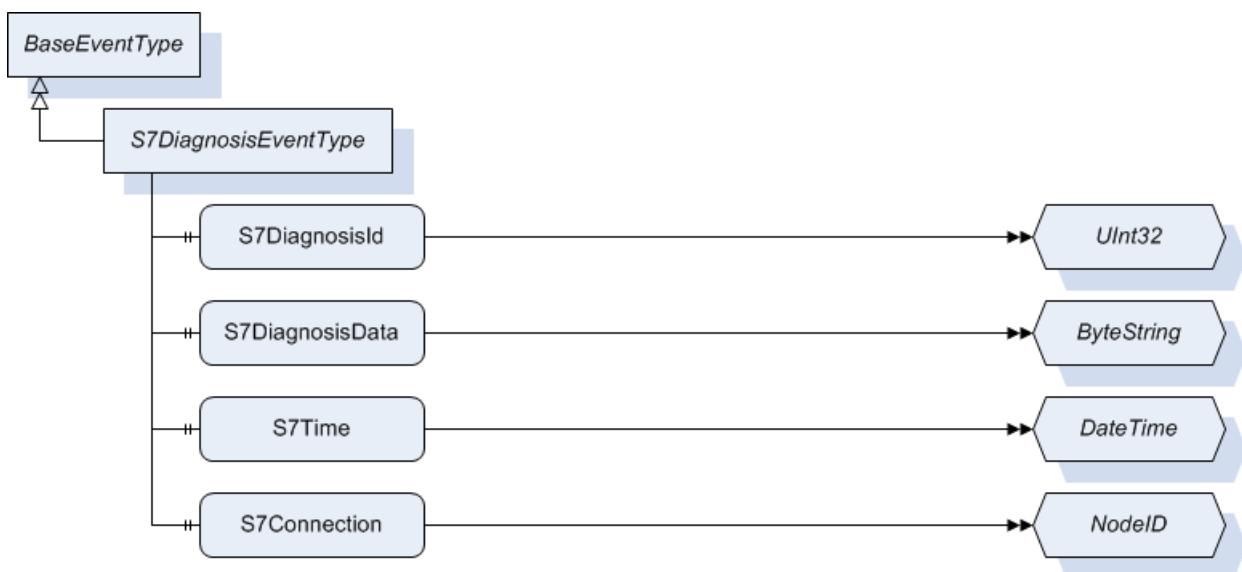


图 2-39 S7DiagnosisEvent 类型表示示例

#### S7DiagnosisId

诊断报警的报警 ID。

#### S7Connection

指定接收诊断报警的连接的 Nodeld。

示例: ns="S7:", s="connectionname"。

## S7DiagnosisData

诊断数据

### 说明

有关 SFC 52 (S7 诊断数据) 的详细说明, 请参见“S7-300/400 的 STEP 7 系统和标准功能”文档。

## 2.7.15 区域树和源空间

OPC 服务器 7.0 中没有区域树和源空间。对于 OPC 服务器 8.0, 区域树和源空间包含已组态 S7 报警的 SourceNode。未组态 S7 报警没有 SourceNode。

S7 UA 报警可建立区域树以向区域分配 SourceNode。

区域树的节点为映射工厂区域的特殊 UA 文件夹。这些特殊的 UA 文件夹称为区域节点。

工厂区域定义为所有块相关和符号相关消息的“附加文本 2”, 例如附加文本“machineroom\tank”通过相应 UA 文件夹中的 NodId 定义高级别区域“machineroom”和低级别区域“tank”

- ns="S7AREAS:", s="machineroom"。
- ns="S7AREAS:", s="machineroom\tank"。

块相关和符号相关报警中的“附加文本 1”可生成一个源节点。例如, 假设“附加文本 1”为“超压传感器”, 则具有下列 NodId 的源节点将对应此附加文本

- ns="S7SOURCES:", s="overpressure sensor"。

“附加文本 1”可包含 "\", 不过, 这不会通向多个 SourceNode。

## 2.7 通过 OPC UA 进行的 S7 通信

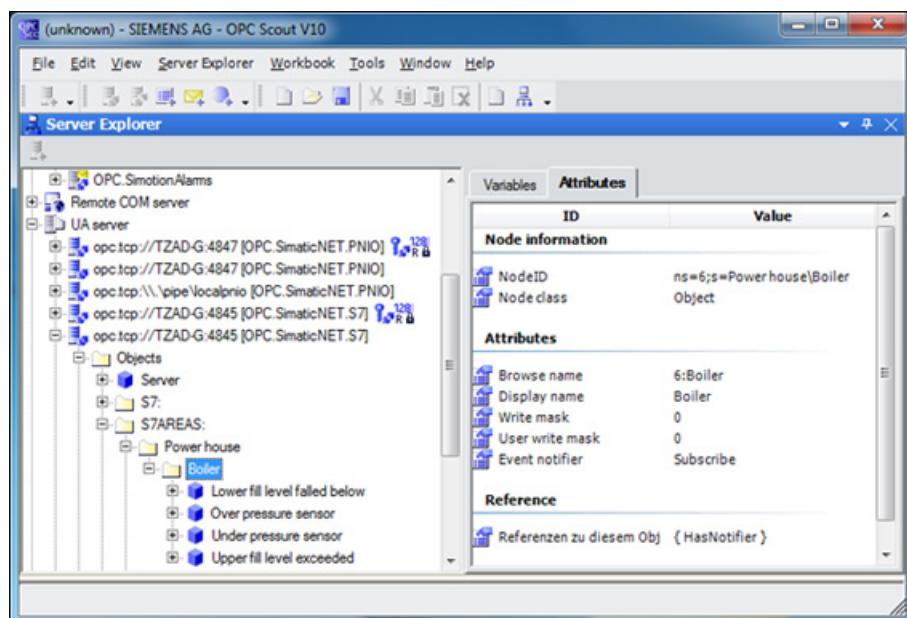
没有必要指定“附加文本 1”和“附加文本 2”。如果这些不存在，则 STEP 7 工厂树中的路径将替代“附加文本 2”，STEP 7 工厂树中的路径 + 报警实例块或符号名称（带有 SCAN）将替代“附加文本 1”。例如，在 AS 设备“Station1”CPU“CPU 416-3”的“S7\_program(1)”报警实例数据块 DB404 中，报警将生成带有次级区域“CPU 416-3”的区域“Station1”以及带有对应 UA 文件夹 Nodeld 的“S7\_program(1)”。

- ns="S7AREAS:", s="Station"
  - ns="S7AREAS:", s="Station1\CPU 416-3"
  - ns="S7AREAS:", s="Station1\CPU 416-3\S7-program(1)"
- 和 SourceNode
- ns="S7SOURCES:", "Station1\CPU 416-3\S7-program(1)\DB404"

诊断报警不含“附加文本 1”和“附加文本 2”。与报警类似，STEP 7 工厂树中的路径将替代附加文本 2，STEP 7 工厂树中的路径 + 报警标识符将替代附加文本

- 1。例如，在上文所示的“S7:program(1)”中，带有报警标识符的诊断报警可生成现有 UA 文件夹和 SourceNode
- ns="S7SOURCES:", "Station1\CPU 416-3\S7-program(1)\WR\_USMSG (1)"

通过 OPC Scout V10 可浏览区域树和源空间。



在左侧窗口选择显示名称为“con13 Scan Notify”（`NodId ns="S7AREAS:", s=" con13-Scan-Notify"`）的区域节点。

右侧窗口将显示其属性；由此可以看到，区域节点拥有参考“HasNotifier”。

## 2.7.16 接收事件

### EventItems 和 EventNotifier

从服务器接收当前事件的唯一方法就是在订阅中生成一个或多个受监视的事件项。

对于事件，EventNotifier 属性 (12) 将受到监视。EventNotifier

仅引用带有“HasNotifier”参考的对象。在 OPC 服务器 8.0 的 S7 UA

报警中，这些是服务器节点 `ns="http://opcfoundation.org/UA/", i=2253`，即 S7

连接的所有节点，例如，`ns="S7:"、s="S7_connection_1"` 和 `ns="S7AREAS:"`

中的所有区域节点。服务器节点 EventNotifier 和 S7 连接节点是报告未组态 S7 报警事件的唯一节点。

生成受监视事件项时，必须指定要返回的属性。

如果不指定属性，则即便发生事件，也无法确定是哪一个。

要将某一属性包括在要返回的属性列表中，必须为各项属性指定：

- 已定义事件类型 (Typeid) 的 NodId、
- 属性的浏览名称与可能的浏览路径以及
- AttributId

在“S7 UA 报警服务器的事件类型体系 (页 230)”部分中，您可以找到已定义事件类型的 NodId、浏览名称以及浏览路径和 AttributId（如果适用）。

---

### 说明

并非所有报警状态的更改都通过 OPC UA 接口上的事件传递。

如果报警状态快速变化，事件可能不报告某些状态变化，只发送上一状态和当前状态。

这取决于“PublishingInterval”和“SamplingInterval”等 OPC 参数，以及组态和计算机性能。

---

### 选择属性

事件不需要返回所有的属性。“零”用于返回事件不具有的属性。

对于事件本身不具备或由于浏览名称等的不规范而导致无效的属性，S7 OPC UA 报警无法区分。

## 使用过滤条件

如果监视区域节点的 EventNotifier，将报告这个区域发生的所有事件。视特定系统而定（可从与区域空间结合的源空间获取一组潜在可用的 EventSource），可以报告非常多的事件。这就需要对事件进行过滤。

### 2.7.17 UA 报警方法

#### Enable()/Disable()

报警禁用的客户端中不会生成事件。即使这样，S7 控制器的报警也将一直受到监视，因为控制器的报警无法单独启用或禁用。

#### AddComment()

可为每个报警实例存储一个注释。

#### ConditionRefresh()

报告未确认报警的所有活动。

#### Acknowledge()

报警的确认将传送到 S7 控制器。不管怎样，确认状态中的更改通过 S7 控制器显式返回 OPC 服务器，然后在 OPC 端报警将改为已确认状态并触发相应事件。

## 2.8 通过 OPC UA 进行的优化 S7 通信

### 2.8.1 通过 OPC UA 进行的优化 S7 通信的属性

借助 OPC UA 和包含 S7-1200 和 S7-1500 类型新站的工业以太网，SIMATIC NET OPC 服务器可支持 S7 通信。

SIMATIC NET 的 S7OPT OPC UA 服务器具有以下特性：

- 变量服务  
通过标准访问和优化访问对 S7 变量进行访问和监视。  
通过符号寻址即可实现优化访问。
- S7 CPU 保护等级概念  
为受保护的连接建立以及对 S7-1200 和 S7-1500 站的访问设置密码
- OPC UA 事件、条件和报警  
处理 PLC 报警

## 2.8.2 支持优化 S7 协议的 SIMATIC NET OPC UA 服务器

### 简介

以下部分所述为通过工业以太网和 OPC UA 对优化 S7 协议进行组态。

### 组态

在“通信设置”组态程序的“OPC 协议选择”(OPC protocol selection) 目录中选择“OPC UA”（在“优化 S7”(S7 optimized) 行）激活 S7OPT OPC UA 服务器。由于优化 S7 协议只适用于 OPC UA，因此不显示“优化 S7”(S7 optimized) 复选标记。因此，启用 S7OPT UA 服务器不会对其它 S7 COM OPC 数据访问服务器的进程内/进程外组态产生影响。

## 2.8 通过 OPC UA 进行的优化 S7 通信



图 2-40 “通信设置”组态程序中为优化 S7 协议选择 OPC UA 的窗口

### 说明

只能通过符号寻址对 S7-1200 站和 S7-1500 站的 S7 变量进行优化访问。

### 说明

使用 STEP 7 Professional (TIA 门户) 或符号编辑器创建符号时，允许使用以下字符：  
 A-Z、a-z、0-9、\_、-  
 、^、!、#、\$、%、&、'、/、(、)、<、>、=、?、~、+、\*、'、`、:、|、@、[、]、{、}、"  
 。使用 STEP 7 创建符号时，还应记住，如果符号文件中的符号同时具有  
<symbolname> 和 <symbolname>[<索引>] 形式，则解析数组时会出现问题。

## 优点/缺点

使用 S7OPT OPC UA 服务器时，仅支持“优化 S7”的进程外模式。必须保持 S7OPT OPC UA 服务器进程的运行以维持准备接收状态。即使所有 OPC UA 客户端都已注销，S7OPT OPC UA 服务器也不会更不能退出运行。

以下优点弥补了这方面的不足：

- 不再需要 COM/DCOM 组态。
- 高速、安全的通信
- PLC 报警和数据访问只需要一个 OPC 服务器。

---

## 说明

如果激活了 S7 协议的 S7OPT OPC UA 服务器，则当首个 OPC UA 客户端登录到 S7OPT OPC UA 服务器时，会立即建立永久组态的 S7 连接。如果尚未连接任何 OPC UA 客户端（例如，计算机启动后），则不会建立永久的 S7 连接。

---

---

## 说明

可在 SIMATIC STEP 7 Professional (TIA 门户) 中通过设置“请求时建立连接”(Establish connection on demand) 对“优化 S7”协议的 S7 连接进行组态。

首次访问变量需要此连接，首次建立该连接，之后只有实际访问时才会建立连接。

也就是说首次访问变量时可能需要更长的时间，建立连接期间可能会发出各种错误信号。

---

### 2.8.3 如何寻址 S7OPT OPC UA 服务器？

#### 服务器 URL

对于本地二进制 TCP 协议，OPC 客户端有两种对服务器进行寻址的方法：

- 直接寻址：
  - `opc.tcp://<hostname>:55105`
  - 或
  - `opc.tcp://<IP-Adresse>:55105`
  - 或
  - `opc.tcp://localhost:55105`

S7OPT OPC UA 服务器使用端口 55105。

- 也可以使用 OPC UA 的发现服务找到 S7OPT OPC UA 服务器。  
发现服务器的位置如下所示：
  - `opc.tcp://<主机名称>:4840`
  - 或
  - `opc.tcp://<IP 地址>:4840`

发现服务器使用端口 4840（用于 TCP 连接）。

#### IPv6 地址

IPv6 地址也可以用作 IP 地址。地址必须位于括号中，例如  
`[fe80:e499:b710:5975:73d8:14]`

#### 端点和安全模式

SIMATIC NET S7OPT OPC UA 服务器支持本地二进制 TCP 协议的端点，并允许使用证书进行验证以及进行加密传送。

已寻址主机上的“发现”服务将用信号通知服务器的端点，换句话说就是服务器的安全要求和协议支持。

S7OPT OPC UA 服务器的服务器 URL“opc.tcp://<主机名称>:55105”提供以下端点：

- 安全模式“SignAndEncrypt”下的端点：  
必须进行签名和加密才能与服务器进行通信。  
通过交换证书和输入密码对通信进行保护。  
除安全模式外，还显示安全策略 Basic128Rsa15。
- 安全模式“无”(none) 下的端点  
此模式下，服务器不需要任何安全功能（安全策略“无”(None)）。

有关安全功能的详细信息，请参见“对 OPC UA 接口进行编程 (页 586)”部分。

OPC 基金会的 UA 规范中对安全策略“Basic128Rsa15”和“无”(None) 进行了介绍，网址为：

<http://opcfoundation.org/UA> (<http://opcfoundation.org/UA>) > "Specifications" > "Part 7"

有关更多的详细信息，请参见以下 Internet 页面：

OPC 基金会 (<http://www.opcfoundation.org/profilereporting/index.htm>)  
> "Security Category" > "Facets" > "Security Policy"

## OPC Scout V10 的 OPC UA 发现

在 OPC Scout V10 中，您可以打开“OPC UA 发现”(OPC UA Discovery) 对话框并在 OPC Scout V10 的导航区域中输入 OPC UA 端点。

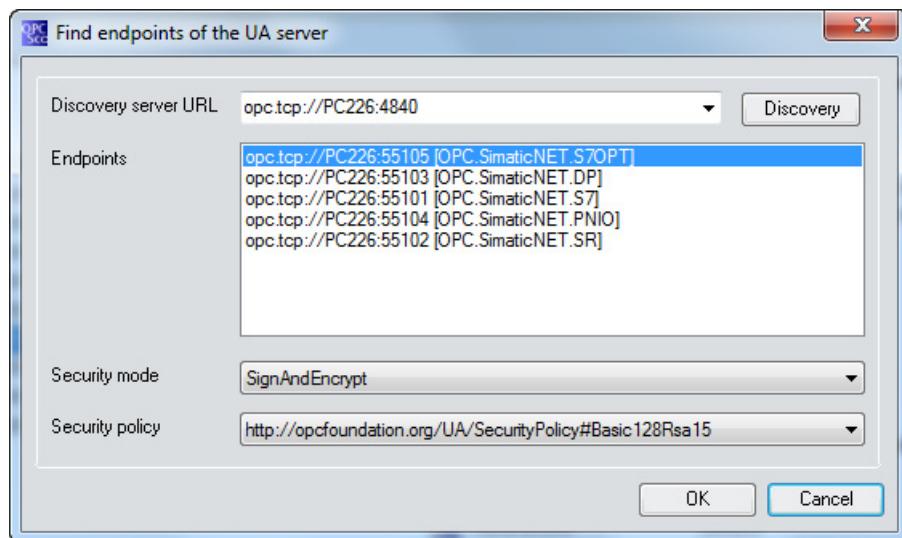


图 2-41 OPC Scout V10 的“查找 UA 服务器端点”对话框

## 2.8 通过 OPC UA 进行的优化 S7 通信

可通过 OPC UA 发现服务查找 S7OPT OPC UA 服务器。 相关条目，请参见上述“服务器 URL”。

OPC Scout V10 包含了一系列 OPC UA 端点。

然后，被寻址主机上的“发现”服务会用信号通知已注册的 OPC UA 服务器、其端口和安全模式。

更多详细信息，请参见 OPC Scout V10 的在线帮助。

### 2.8.4 S7OPT OPC UA 服务器提供了哪些命名空间？

S7OPT OPC UA 服务器提供了以下命名空间：

命名空间索引	“标识符”（命名空间 URI）/注释
0	“ <a href="http://opcfoundation.org/UA/">http://opcfoundation.org/UA/</a> ” (由 OPC 基金会指定)
1	“urn:Siemens.Automation.SimaticNET.S7OPT:(GUID)” 本地高速 S7OPT OPC UA 服务器的唯一标识符。
2	“S7OPTTYPES:” S7OPT 特定对象类型的定义。
3	“S7OPT:” 本地高速 S7OPT OPC UA 服务器的名称。
4	“S7OPTSOURCES:” 报警事件来源的标识符。
5	“S7OPTAREAS:” 报警层级结构区域的标识符。
6	“SYM:” 可选服务器，使用 ATI-S7OPT 符号；取决于项目工程组态和 PC 站组态（可浏览，也可与 OPC UA 一起使用）。 或者，此处也可使用在符号的参数分配中指定的前缀（“通信设置”）。

命名空间索引 0 和 1 保留，其意义由 OPC 基金会指定。

将剩余命名空间索引分配至标识符（命名空间 URI），这一操作必须由指定该标识符的客户端通过 OPC UA

会话开始处的数据变量“NamespaceArray”来完成。

标识符“S7OPTTYPES:”、“S7OPT:”、“S7OPTAREAS:”和“S7OPTSOURCES:”始终与 S7OPT OPC UA 服务器一起存在。

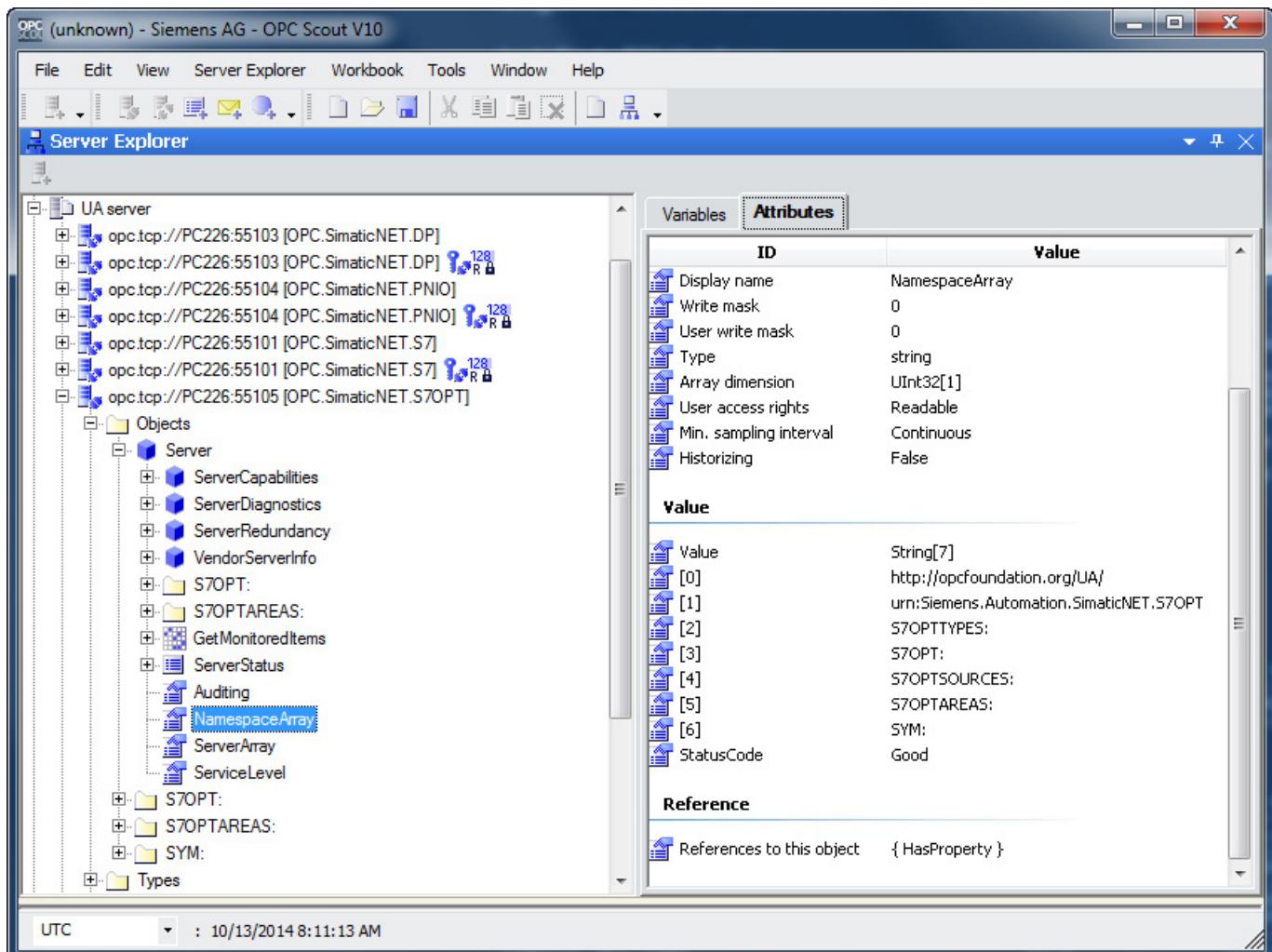


图 2-42 使用 OPC Scout V10 的浏览功能显示 S7OPT OPC UA 的命名空间

## 2.8.5 NodId

### S7OPT 过程变量的标识

NodId 借助下列元组来标识 S7OPT 过程变量：

- 命名空间索引
- 标识符（字符串）

## 示例

- **NodId:**
  - 命名空间 URI:  
S7OPT:  
(= 命名空间索引 3)
  - 标识符:  
S7\_connection\_1.m.10, b

## OPC UA 命名空间如何运作?

OPC UA 视图不仅显示自动化对象，同时还显示对象的各种属性。OPC UA 可访问对象及其子对象。

- 诸如数据变量、方法及某种程度上的事件都是“优化 S7”连接对象的子对象。特性和属性可更详细地定义对象。

合格的 **NodId** 标识符在 OPC UA 中非常重要。每次单独访问对象、子对象、特性及属性均使用其 **NodId**。

OPC UA 提供显示名称等内容以支持本地语言。

也就是说，可通过不同方式浏览相同对象（例如在 OPC UA 客户端指定的各种语言环境下），每次都显示同一个 **NodId**。显示名称的选择与相关 **NodId** 类似。整个命名空间的文本均采用英语。

## S7OPT 过程变量的语法

OPC UA 中引入了一种优化语法。所有 OPC UA 对象的 **NodId** 均具有以下结构：

<connectionobject>."<subobject>".<property>

子对象可包含其它子对象。

无法解析的 **NodId** 会被拒绝，并会显示错误消息。所有 **NodId** 中的字母“A-Z”均不区分大小写。

## 符号对象表示法

OPC UA 规范建议对地址空间的层级描述使用统一的符号表示法。

本文档中使用以下符号：

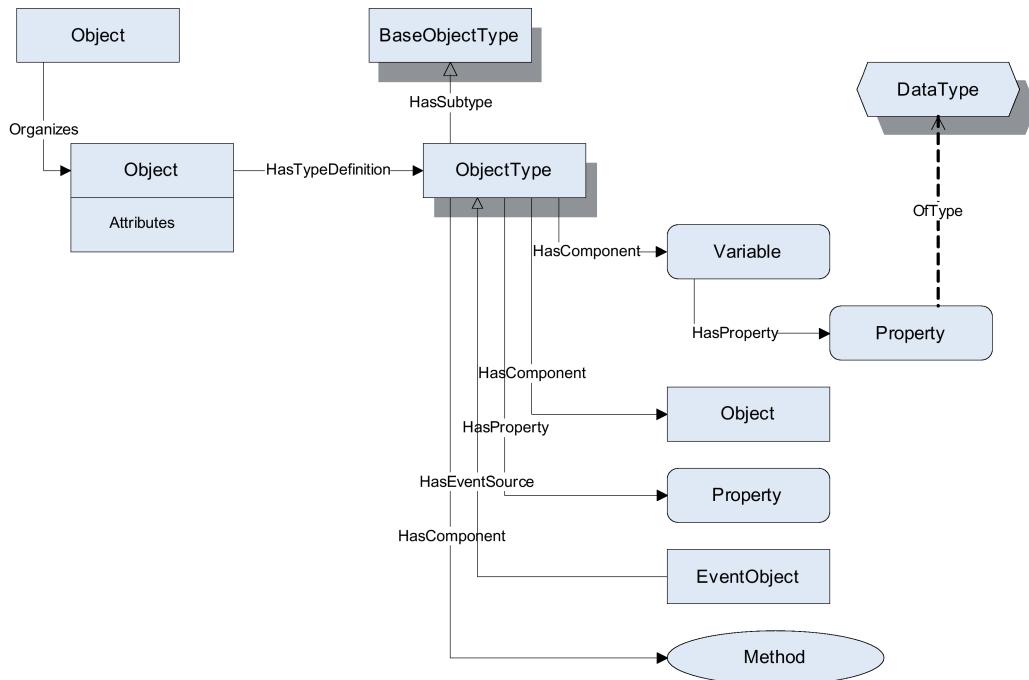


图 2-43 OPC UA 地址空间的符号

## 2.8.6 连接对象

### S7OPT 连接对象

存在以下 S7OPT 连接对象：

- **S7 生产连接**

优化 S7 通信采用 S7 连接，用于可编程控制器之间数据交换，在 STEP 7 Professional (TIA Portal) 中进行组态。

- **CpuSubscription 对象**

此对象用于支持直接对 CPU 上的数据变更进行循环监控，而无需轮询。S7-1200/S7-1500 站支持的 CpuSubscriptions 数量有限。

#### 说明

有关 CPU 组态限制的信息，请参见相关设备手册的“技术规范”部分。

### CpuSubscription 对象

<连接对象>="cpusubscription"

### 说明

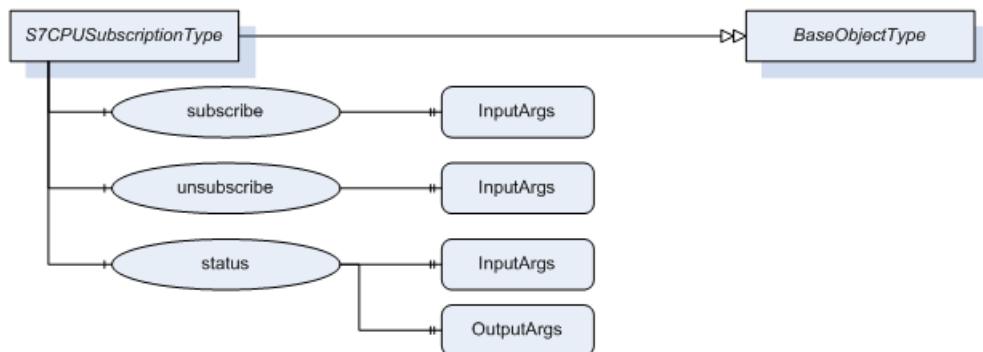
保留连接名称“CpuSubscription”，不得将其用作 S7 连接的名称。STEP 7 Professional (TIA Portal) 在创建连接组态的过程中可防止此类情况的发生。

在 S7-1200 和 S7-1500 站中，S7OPT OPC UA 服务器允许对 CPU 上的数据变更进行直接监控，而无需对 S7 连接执行循环读取作业。使用这些 CpuSubscriptions 对象，可大大减少 CPU 的通信负载。

CpuSubscriptions 的数量取决于所使用的 S7 站的类型，必须严格限制。对象的数量限制在几百个（几千字节），且 S7OPT OPC UA 服务器正在运行时不能对其进行查询。这不仅限于 S7OPT OPC UA 服务器，其它应用程序同样适用。

为此，S7OPT OPC UA 服务器为其 UA 客户端提供了使用 OPC UA 方法的选项，以在首选的数据变量中查询 CpuSubscriptions，获取这些 CpuSubscriptions 的当前状态。

这些方法并不局限于单个 S7 连接。将 OPCUA\_NodeIds 假设为数据变量的参数，基于这些数据变量，可识别连接到 CPU 的路径。符号的 OPCUA\_NodeIds 通常不包含连接名称，对于这样的符号，其数据变量也可包含在 CpuSubscriptions 中。



### CpuSubscription 方法的语法

命名空间 URI: S7OPT: (命名空间索引: 3)

*cpusubscription.subscribe(OpcaUa\_UInt32 cycletime, OpcaUa\_NodeId[] nodeids)*  
*cpusubscription.unsubscribe(OpcaUa\_NodeId[] nodeids)*  
*cpusubscription.status(OpcaUa\_NodeId[] nodeids, OpcaUa\_Status[] status)*

### 示例：

```
cpusubscription.subscribe(100, {S7OPT_1.db10.1,b|PCStation1.PLC1500.optDB.Byte})  
cpusubscription.unsubscribe({S7OPT_1.db10.1,b|PCStation1.PLC1500.optDB.Byte})  
cpusubscription.status({S7OPT_1.db10.1,b|PCStation1.PLC1500.optDB.Byte}, Status)
```

### OPC UA 方法的说明

- **subscribe()**

为多个数据变量创建 CpuSubscription。 (如果发生错误，可使用相应的 OPC UA 错误代码，另请参见“OPC UA 服务器的消息 (页 605)”部分)

- InputArgument1: “cycletime”；将 PLC 的数据更改以信号形式发送到 S7OPT OPC UA 服务器的时间间隔。

OPC UA“UInt32”数据类型的数据变量，指定以 ms 为单位，最小 100 ms 且只能为 100 ms 的整数倍

- InputArgument2: “nodeids”；指定使用 CpuSubscription 访问选项创建一个还是多个 Nodeld。

OPC UA“OpcUa\_Nodeld[]”数据类型的数据变量

### 说明

如果已为 Nodeld 创建一个 CpuSubscription，则要根据已更改的循环时间修改 CpuSubscription。

- **unsubscribe()**

删除 CpuSubscription。 (如果发生错误，可使用相应的 OPC UA 错误代码，另请参见“OPC UA 服务器的消息 (页 605)”部分)

- InputArgument1: “nodeids”；可显示 Nodeld 列表  
OPC UA“OpcUa\_Nodeld[]”数据类型的数据变量

- **status()**

返回 CpuSubscription 的状态。 (如果发生错误，可使用相应的 OPC UA 错误代码，另请参见“OPC UA 服务器的消息 (页 605)”部分)

- InputArgument1: “nodeids”；可显示 Nodeld 列表  
OPC UA“OpcUa\_Nodeld[]”数据类型的数据变量  
- OutputArgument1: “status”  
OPC UA“OpcUa\_Status[]”数据类型的数据变量

CpuSubscription 方法调用可添加或删除 S7OPT OPC UA 服务器中内部 CpuSubscription 管理的已传递 Nodeld，与连接伙伴的连接状态无关。  
因此，(成功) 完成方法调用并不代表可以在 CPU 上实际生成 CpuSubscription。

如果使用了错误的 **NodeId**，这些函数将返回参数错误，已传送的 **NodeId** 也无法传送至 **CpuSubscription** 管理。

### 说明

如果超出了可用的 **CpuSubscriptions**

或数据变量的数量限制，则方法返回参数的评估中不会提供任何关于在 CPU 上成功创建 **CpuSubscription** 的信息。

为此，需要通过调用状态方法来检查支持或不支持的方法是否成功完成，如果出现错误，则会通过调整或减少数据变量/**CpuSubscriptions** 的数量来响应。

成功建立连接或添加/删除 **CpuSubscription** 之后，S7OPT OPC UA 服务器会对相关连接上当时已注册的所有 **CpuSubscription** 进行优化，并通过连接伙伴对其重新注册或注销。

如果通过连接伙伴注册成功，则该服务器会在数据更改后自动返回更改数据。可将 S7OPT OPC UA 服务器上建立的具有相同或更高更新速率的监视项映射至使用的 **CpuSubscription**，而不再对这些监视项进行轮询。

如果通过连接伙伴注册未成功，则会以标准步骤对映射到该 **CpuSubscription** 的 **MonitoredItem** 进行轮询，而不使用 **CpuSubscription**。该 **CpuSubscription** 的状态将设置为“故障”(Bad)，可使用状态方法读取其状态，另请参见“OPC UA 服务器的消息 (页 605)”部分。

**CpuSubscription** 对所有 UA 客户端均有效，也就是说 UA 客户端可以删除由其它客户端建立的 **CpuSubscription**。最后一个 UA 客户端完成注销后，将删除所有此时仍然存在的 **CpuSubscription**。

为了能够以最佳方式充分利用 **CpuSubscription** 服务，在建立并使用 **CpuSubscription** 时应记住以下几点：

- 仅为最重要的数据变量注册 **CpuSubscription**
- 若要监视该数据区内大量的小数据变量，则可以注册更大的连续区域。
- 在可能的情况下，对那些不常变更但其变更需要快速响应的数据变量进行注册
- **CpuSubscription** 的循环时间要与 **MonitoredItem** 的更新时间相匹配。

对于 OPC 客户端来说，**CpuSubscriptions** 的使用是透明的。无论哪种情况（无 **CpuSubscription**、已成功注册 **CpuSubscription**、未成功注册 **CpuSubscription**），都将通过轮询或 **CpuSubscription** 监视已注册为 **MonitoredItem** 的数据变量的数据变更。

S7OPT OPC UA 服务器不会影响连接伙伴执行确认处理的顺序或发送的数据变更信号。可同时对读取作业和数据变更消息进行确认。因此，S7OPT OPC UA 服务器始终保持将其存储器更新为所接收到的最新值。

## 2.8.7 连接名称

### S7 连接的连接名称

连接名称是指在 STEP 7 Professional (TIA Portal) 中组态的名称，用于标识连接。在 STEP 7 Professional (TIA Portal) 中，此名称称为“本地 ID”。在 OPC 服务器内，本地 ID 是唯一的。

#### 连接类型

OPC 服务器支持 S7 连接的连接类型。

#### S7 连接名称允许使用哪些字符？

对于 <connectionname>，可使用数字“0-9”、大小写字母字符“A-z”以及特殊字符。

连接名称的开头和结尾不允许使用空格，不允许使用特殊字符“.”、“|”、“\”、“/”、“[”、“]”。

连接名称的长度最多可为 24 个字符。名称不区分大小写。

不允许使用 ASCII 码大于 126 及小于 30 的可打印字符。

连接名称“SYSTEM”和“cpusubscription”保留，不得使用。

#### S7 连接名称示例：

典型示例包括：

- S7\_connection\_1
- S7OPT OPC connection

## 2.8.8 S7OPT 生产连接对象的结构和功能

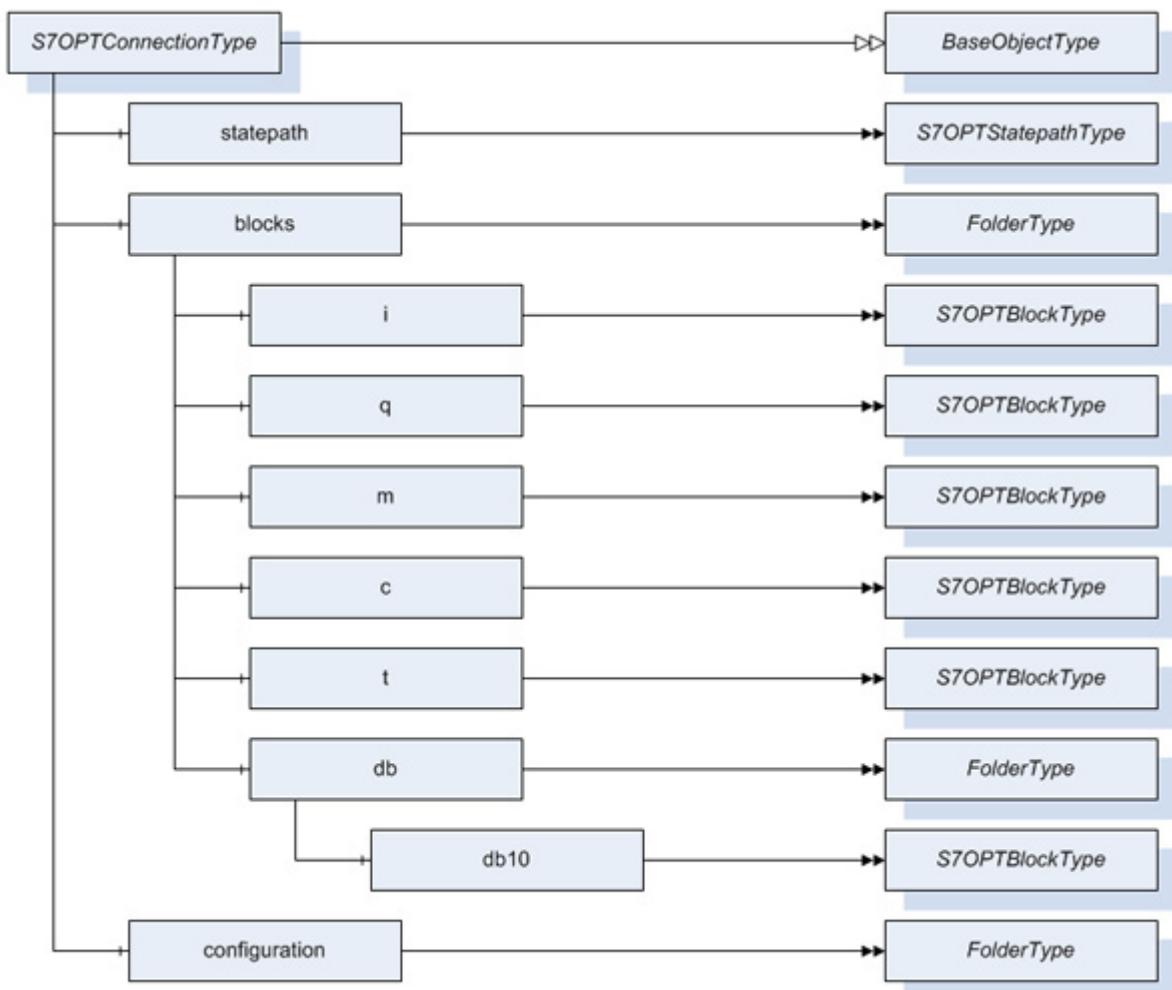
### 什么是 S7OPT 连接对象？

所有生产协议特定对象都将分配到连接。对于“优化 S7”，这些是与通信伙伴之间的连接（S7 连接）。CpuSubscription 对此是个例外。

## 2.8.9 S7OPT 连接对象的类型定义

### S7OPT 连接对象的类型定义

对于可通过 S7 生产连接使用的对象和功能，将定义特定的 OPC UA 对象类型：



对象的 OPC UA 命名空间将显示此类型的实例。可在“类型”(Type)下构造化读取类型本身。

## 2.8.10 S7OPT 连接信息对象

### S7OPT 特定信息数据变量

可通过 S7OPT 特定数据变量获取优化 S7 通信和已建立连接的相关信息。

可获取以下信息：

- 优化访问时 S7 连接的状态

### S7OPT 特定信息变量的语法

Nodeld:

命名空间索引：3

*<connectionobject>. <informationparameter>*

### 说明

*<informationparameter>:=“statepath”*

alarm	属于 statepath 对象的 statepath 报警实例。		
statepath	与伙伴设备之间的通信连接状态 该变量的值以数字形式读出，能通过额外读取相关 Enumstring {UNKNOWN, DOWN, UP, RECOVERY, ESTABLISH} 将其分配到文本。 UA“MultistateDiscreteType”类型的变量，只读		
0	UNKNOWN	为将来的扩展保留	
1	DOWN	未建立连接	
2	UP	已建立连接	
3	RECOVERY	未建立连接。正在尝试建立连接。	
4	ESTABLISH	为将来的扩展保留	

## 2.8.11 S7OPT 特定信息变量和返回值的示例

以下示例说明了 S7OPT 特定信息变量名称的语法。

### statepath 报警优先级

- Nodeld:
  - 命名空间 URI:  
S7OPT: (命名空间索引: 3)
  - 标识符:  
S7\_connection\_1.statepath.alarm.Severity

可能的返回值: 500

### 可能的通信连接状态文本

- Nodeld:
  - 命名空间 URI:  
S7OPT: (命名空间索引: 3)
  - 标识符:  
S7\_connection\_1.statepath.statepath.EnumStrings

可能的返回值: { UNKNOWN | DOWN | UP | RECOVERY | ESTABLISH }

### 数值形式的通信连接状态

- Nodeld:
  - 命名空间 URI:  
S7OPT: (命名空间索引: 3)
  - 标识符:  
S7\_connection\_1.statepath.statepath

可能的返回值: 3 (RECOVERY) - 当前正在建立连接。

## 2.8.12 变量服务

### 2.8.12.1 标准访问

#### 变量服务调用标准访问的作用？

调用标准访问的变量服务可对可编程控制器中的 S7 变量进行访问并监视。S7 变量以地址对象短名称的形式进行寻址。访问类型取决于 S7 工具的表示法。

#### 可编程控制器中的对象

S7OPT OPC UA 服务器支持以下对象：

- 数据块（标准访问）
- 背景数据块和多重背景数据块（标准访问）
- 输入
- 输出
- 位存储器
- 定时器（仅限 S7-1500）
- 计数器（仅限 S7-1500）
- UDT

并非每种 S7 可编程控制器都支持所有对象类型。

---

#### 说明

此外，也可以使用符号对变量服务进行标准访问。

有关此主题的所有必要信息，请参见《调试 PC 站》手册

---

#### 变量服务的语法

#### 过程变量的语法

S7OPT OPC UA Nodeld 过程变量的简化语法：

命名空间 URI: S7OPT: (命名空间索引: 3)

## 典型语法

有三个选项：

```
<connectionname>.<S7OPTobject>.<address>{,<S7OPTtype>{,<quantity>}}  

<connectionname>.<S7OPTtimerobject>.<address>  

{,<S7OPTtimertype>{,<quantity>}}  

<connectionname>.<S7OPTcounterobject>.<address>  

{,<S7OPTcountertype>{,<quantity>}}
```

## 说明

命名空间 URI: S7OPT: (命名空间索引: 3)

### **<connectionname>**

协议特定的连接名称。连接名称是在组态过程中指定的。  
其后的分隔符为句点 (“.”)。

### **<S7OPTobject>**

指定 S7 上的区域类型。可能值如下：

参数	含义
db<no>	数据块或实例数据块。数据块中 S7OPT 变量的标识符。在 S7 通信中，不区分实例数据块和正常数据块。 因此，无需为省事而分配附加标识符。  <no> 数据块或实例数据块的编号，无前导零。 只能使用这些数据变量对 S7-1200/S7-1500 上的非优化数据块进行寻址。 对于优化数据块中的数据变量，运行期间无法创建任何地址信息。 因此需使用符号访问来访问这些优化数据变量。
m	位存储器
i	输入 可读写 为简化起见，仅使用英语标识符。
q	输出 可读写 为简化起见，仅使用英语标识符。

其后的分隔符为句点 (\*.\*)。

#### <S7OPTtimerobject>

类型标识符	S7 数据类型	OPC UA 数据类型	说明
t	TIMER	UInt16	后面的地址信息为定时器编号。

#### <S7OPTcounterobject>

类型标识符	S7 数据类型	OPC UA 数据类型	描述
c	COUNTER	UInt16	后面的地址信息是计数器编号。

#### <address>

要寻址的第一个变量的字节地址。不支持前导零（如 001）。

字节地址的取值范围为 0 至

65534。根据所使用设备及类型的不同，地址的实际可用值可能会较低。

#### <S7OPTtype>

S7 数据类型将转换为 OPC UA 服务器上对应的 OPC UA 数据类型。

下表列出了类型标识符以及可用于表示变量值的对应 OPC UA 数据类型。

类型标识符	S7 数据类型	OPC UA 数据类型	描述
b	BYTE	字节	8 位（位字符串）
	USINT	字节	8 位（无符号）
c	CHAR	SByte	8 位（字符）
	SINT	SByte	8 位（有符号）
w	WORD	UInt16	16 位（位字符串）
	UINT	UInt16	16 位（无符号）
dw	DWORD	UInt32	32 位（位字符串）

类型标识符	S7 数据类型	OPC UA 数据类型	描述
	UDINT	UInt32	32 位 (无符号)
lw	LWORD	UInt64	64 位 (位字符串)； 仅适用于 S7-1500
	ULINT	UInt64	64 位 (无符号)； 仅适用于 S7-1500
i	INT	Int16	16 位 (有符号)
di	DINT	Int32	32 位 (有符号)
li	LINT	Int64	64 位 (有符号)； 仅适用于 S7-1500
r	REAL	浮点	浮点 (4 字节) IEEE 754
lr	LREAL	双精度	浮点 (8 字节) IEEE 754
dt	DATE_TIME	UtcTime	日期和时钟， 值范围, 从 1990 年 1 月 1 日开始； 仅适用于 S7-1500
ldt	LDT	UtcTime	日期和时钟, 精确到纳秒 值范围, 从 1990 年 1 月 1 日开始； 仅适用于 S7-1500
dtl	DTL	UtcTime	日期和时钟, 精确到纳秒 值范围 1970 年 1 月 1 日 – 2553 年 12 月 31 日； 仅适用于 S7-1500
日期	DATE	UtcTime	日期和时间 (8 字节), 时间始终为 00:00:00, 值范围从 1990 年 1 月 1 日开始。  CPU 数据类型“DATE”映射 (无符号 , 16 位)。
t	TIME	Int32	有符号时间值, 单位为毫秒

类型标识符	S7 数据类型	OPC UA 数据类型	描述
lt	LTIME	Int64	有符号时间值，单位为纳秒；仅适用于 S7-1500
tod	TOD	UInt32	时钟，从午夜开始，0 到 86399999 ms
ltod	LTOD	UInt64	时钟，从午夜开始，0 到 86399999999999 ns；仅适用于 S7-1500
s5bcd	S5TIME	UInt16	CPU 数据类型“S5TIME”到 UInt16（无符号，16 位）的映射，值范围限制为 0 至 9990000 ms。*)；仅适用于 S7-1500
x<bitaddress>	BOOL	布尔	位（布尔） 除区域内的字节偏移量外，还必须在字节中指定 <bitaddress>。 值范围是 0 到 7
s<stringlength>	STRING	字符串	还必须指定为字符串保留的 <stringlength>。 值范围是 1 到 254 写入时还可写入较短的字符串，使传送的数据长度始终为保留的字符串长度（字节）加 2 个字节。不必要的字节用值 0 填充。 读写字符串和字符串数组在内部映射到读写字节数组。 在 S7 上，字符串必须用有效值初始化。

\*) 请参见下表“S7 数据类型“s5bcd”的时间基准和值范围”

### 说明

如果未指定“<S7OPTTyp>”，类型标识符“<b>”将被 S7OPT OPC UA 服务器用作默认值。

**说明**

“DTL”S7 数据类型只能由 SIMATIC NET OPC UA 服务器读取。

**说明**

S7 数据类型“Byte[]”通过 SIMATIC NET OPC UA 服务器映射到字节字符串。

**S7 数据类型“s5bcd”的时间基准和值范围**

数据类型“s5bcd”时间变量的值范围采用 BCD 编码。值范围请参见下表：

位号	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
含义（符号）	0	0	x	x	t	t	t	t	t	t	t	t	t	t	t	t
<b>说明：</b>																
“0”的含义	不相关															
“x”的含义	指定时间基准															
	位 13 和 12								时间基准（毫秒）							
	00								10							
	01								100							
	10								1000							
	11								10000							
“t”的含义	BCD 编码的时间值（0 至 999）															

**示例：**

位号	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
值	0	0	0	1	0	0	0	0	0	1	1	1	0	1	0	1

位 0-11 指定值为 075。位 12 和 13 指定时间基准为 100 ms。

$75 * 100 \text{ ms} = 7500 \text{ ms}$ 。以 100 ms 为基准递减时钟 75 次。

时间变量（数据类型 s5bcd）的 OPC 数据类型为字 (UInt16)。

写入时，相应限制值范围。

**<S7OPTTimertype>**

S7 定时器数据类型将转换为 OPC UA 服务器上对应的 OPC UA

数据类型。下表列出了类型标识符以及可用于表示变量值的对应 OPC UA 数据类型。

类型标识符	S7 数据类型	OPC UA 数据类型	说明
tbcd	TIMER	UInt16	BCD 编码 如果未指定 <S7timertype>, 则用作默认值。
tda	TIMER	UInt16[2]	十进制时间基准和时间值

**S7OPTtimerobject“t”和 S7OPTtimertype“tbcd”的时间基准和值范围:**

类型“tbcd”的 S7 定时器变量的值范围采用 BCD 编码。

时间基准（用于写入）取自下表所示值范围：

位号	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
含义（符号）	0	0	x	x	t	t	t	t	t	t	t	t	t	t	t	t
<b>说明：</b>																
“0”的含义	不相关															
“x”的含义	指定时间基准															
	位 13 和 12								时间基准（毫秒）							
	00								10							
	01								100							
	10								1000							
	11								10000							
“t”的含义	BCD 编码的时间值（0 至 999）															

**示例：**

位号	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
值	0	0	0	1	0	0	0	0	0	1	1	1	0	1	0	1

位 0-11 指定值为 075。位 12 和 13 指定时间基准为 100。

$75 * 100 = 7500 \text{ ms}$ 。以 100 ms 为基准递减时钟 75 次。

**S7 定时器数据类型“tda”的时间基准和值范围:**

数据类型：由两个字组成的字段 {以毫秒为单位的时间基准 UInt16 | 时间值 UInt16}

时间基准 [ms]	10, 100, 1 000, 10 000
时间值	0 ... 999
时间基准 [ms]	10 ms: 0 ... 9 990 100 ms: 0 ... 99 900 ms 1 000 ms: 0 ... 99 9000 10,000 ms: 0 ... 9 990 000

“TDA”对象无法输入 <quantity>。

**示例：**

示例：将值 {100|50} 写入定时器“t.3,tda”，“tda”以值  $50 * 100 \text{ ms} = 5000$  初始化定时器 3，以 100 ms 为基准递减时钟 50 次。

**<S7OPTcountertype>**

类型标识符	S7 数据类型	OPC UA 数据类型	描述
c	COUNTER	UInt16	S7 的值范围：0 到 999，十进制编码。如果未指定 <S7OPTcountertype>，则用作默认值。

**<quantity>**

要寻址的某一类型变量的数目，从“address”参数中指定的偏移量开始（取值范围为 1 到 65 535）。

指定多个数组元素将导致形成相应类型的数组，即使只对一个数组元素进行寻址也是如此。

分隔符为逗号（“,”）。

对于数据类型“x”，写访问的数量只能输入 8 的倍数。之后位地址必须为零。

对数据类型“x”，不限制读访问数量的输入。位地址的取值范围之后可为 0 到 7。

**示例：**

命名空间 URI: *S7OPT*: (命名空间索引: 3)

*S7-OPC-1.db1.10,x0,64*, 访问权限 RW

*S7-OPC-1.db1.10,x3,17*, 访问权限 R

**非优化 S7OPT OPC UA 变量服务的过程变量示例**

以下示例说明了变量服务的变量名称的语法。

**非优化数据模块 DB 单字节**

命名空间 URI: *S7OPT*: (命名空间索引: 3)

*S7\_connection-1.db5.12,b*

表示 *S7\_connection-1* 中数据块 5 的数据字节 12。

**非优化数据块 DB, 字数组**

命名空间 URI: *S7OPT*: (命名空间索引: 3)

*S7\_connection-1.db5.10,w,9*

表示 *S7\_connection-1* 中数据块 5 自字节地址 10 起的 9 个数据字。

**非优化数据块 DB, 字符串数组**

命名空间 URI: *S7OPT*: (命名空间索引: 3)

*S7\_connection-1.db100.50,s32,3*

表示 *S7\_connection-1* 中数据块 100 自字节地址 50 起长度为 32 的 3 个字符串。

**输入 0**

命名空间 URI: *S7OPT*: (命名空间索引: 3)

*S7\_connection-1.i.0*

表示 *S7\_connection-1* 中的输入字节 0。

## 输出 0 位 0

命名空间 URI: *S7OPT*: (命名空间索引: 3)

*S7-connection-1.q.0,x0*

表示 *S7\_connection-1* 中的输出地址 0, 位 0。

## 可读存储器位数组

命名空间 URI: *S7OPT*: (命名空间索引: 3)

*S7-connection-1.m.3, x4,12 // 访问权限 R*

表示 *S7\_connection-1* 中自位存储器地址 3 起的 12 位和自位地址 4 起的 12 位, 只读。

## BCD 编码定时器 22

命名空间 URI: *S7OPT*: (命名空间索引: 3)

*S7\_connection-1.t.22*

表示 *S7\_connection-1* 中的定时器 22, TBCD 默认值。

### 2.8.12.2 优化访问

#### 变量服务调用优化访问的作用?

调用优化访问的变量服务可对可编程控制器中的 S7 变量进行优化访问和监视。对 S7 变量进行符号寻址。访问类型取决于 S7 符号的表示法。

#### 可编程控制器中的对象

*S7OPT OPC UA* 服务器支持以下对象:

- 数据块 (优化访问)
- 实例数据块 (优化访问)

---

#### 说明

结合 S7-1200 (自 V4.0 起) 和 S7-1500 站运行通过 OPC UA 进行的优化 S7 通信。

---

## 优化变量服务的语法

### 符号寻址的语法

S7OPT OPC UA Nodeld 过程变量的简化语法:

命名空间 *URI*: *SYM*: (命名空间索引: 4)

### 符号语法

<stationname>. <CPUname>{ . <folder> } . <symbol>

#### 说明

命名空间 *URI*: *SYM*: (命名空间索引: 4)

**<stationname>**

在 STEP 7 Professional (TIA Portal) 中组态期间指定的站名称。其后的分隔符为句点 (\*.\*)。

**<CPU\_name>**

在 STEP 7 Professional (TIA Portal) 中指定的 S7 CPU 名称。其后的分隔符为句点 (\*.\*)。

**<folder>**

S7 CPU 中的数据块或数据结构的名称。在 STEP 7 Professional (TIA Portal) 中指定的名称。其后的分隔符为句点 (\*.\*)。

文件夹结构需根据数据块中的数据结构进行相应扩展。

**<symbol>**

S7 CPU 中的变量名称。在 STEP 7 Professional (TIA Portal) 的数据块或符号表中指定的变量名称。

S7 数据类型	OPC UA 数据类型	描述
BYTE	字节	8 位 (位字符串)
USINT	字节	8 位 (无符号)
CHAR	SByte	8 位 (字符)
SINT	SByte	8 位 (有符号)
WORD	UInt16	16 位 (位字符串)
UINT	UInt16	16 位 (无符号)

S7 数据类型	OPC UA 数据类型	描述
DWORD	UInt32	32 位 (位字符串)
UDINT	UInt32	32 位 (无符号)
LWORD	UInt64	64 位 (位字符串)； 仅适用于 S7-1500
ULINT	UInt64	64 位 (无符号)； 仅适用于 S7-1500
INT	Int16	16 位 (有符号)
DINT	Int32	32 位 (有符号)
LINT	Int64	64 位 (有符号)； 仅适用于 S7-1500
REAL	浮点	浮点 (4 字节) IEEE 754
LREAL	双精度	浮点 (8 字节) IEEE 754
DATE_TIME	UtcTime	日期和时钟， 值范围，从 1990 年 1 月 1 日开始； 仅适用于 S7-1500
LDT	UtcTime	日期和时钟，精确到纳秒 值范围，从 1990 年 1 月 1 日开始； 仅适用于 S7-1500
DTL	UtcTime	日期和时钟，精确到纳秒 值范围 1970 年 1 月 1 日 – 2553 年 12 月 31 日； 仅适用于 S7-1500
DATE	UtcTime	日期和时间 (8 字节)，时间始终为 00:00:00， 值范围从 1990 年 1 月 1 日开始。 <b>CPU</b> 数据类型“DATE”映射 (无符号，16 位)。
TIME	Int32	有符号时间值，单位为毫秒
LTIME	Int64	有符号时间值，单位为纳秒； 仅适用于 S7-1500

S7 数据类型	OPC UA 数据类型	描述
TOD	UInt32	时钟, 从午夜开始, 0 到 86399999 ms
LTOD	UInt64	时钟, 从午夜开始, 0 到 86399999999999 ns; 仅适用于 S7-1500
S5TIME	UInt16	CPU 数据类型“S5TIME”到 UInt 16 (无符号, 16 位) 的映射, 值范围限制为 0 至 9990000 ms。*); 仅适用于 S7-1500
BOOL	布尔	位 (布尔) 除区域内的字节偏移量外, 还必须在字节中指定 <bitaddress>。 值范围是 0 到 7
STRING	字符串	还必须指定为字符串保留的 <stringlength>。 值范围是 1 到 254 写入时还可写入较短的字符串, 使传送的数据长度始终为保留的字符串长度 (字节) 加 2 个字节。不必要的字节用值 0 填充。 读写字符串和字符串数组在内部映射到读写字节数组。 在 S7 上, 字符串必须用有效值初始化。

---

#### 说明

“DTL”S7 数据类型只能由 SIMATIC NET OPC UA 服务器读取。

---

#### 说明

使用用户自定义数据类型 (UDT) 时, SIMATIC NET OPC UA 服务器仅支持对 UDP 子元素的访问。无法访问整个 UDT。

---

### 说明

S7 数据类型“Byte[]”通过 SIMATIC NET OPC UA 服务器映射到字节字符串。

---

## 优化 S7OPT OPC UA 变量服务的过程变量示例

以下示例说明了变量服务的变量名称的语法。

### 优化数据块 DB 单字节

命名空间 URI: SYM: (命名空间索引: 4)

*S7-1500-Station\_1.S7-1500.datatypes\_optimized.byte*

表示站“S7-1500-Station\_1”中 CPU“S7-1500”上数据块“datatypes\_optimized”的 S7 数据变量“byte”(字节)。

在 STEP 7 Professional

项目中，用户为站、CPU、数据块和变量分配唯一的名称来设定语法规则。在 TIA Portal OPC 服务器的“符号组态”(Symbol configuration) 对话框中选择 OPC 地址空间的可用符号。

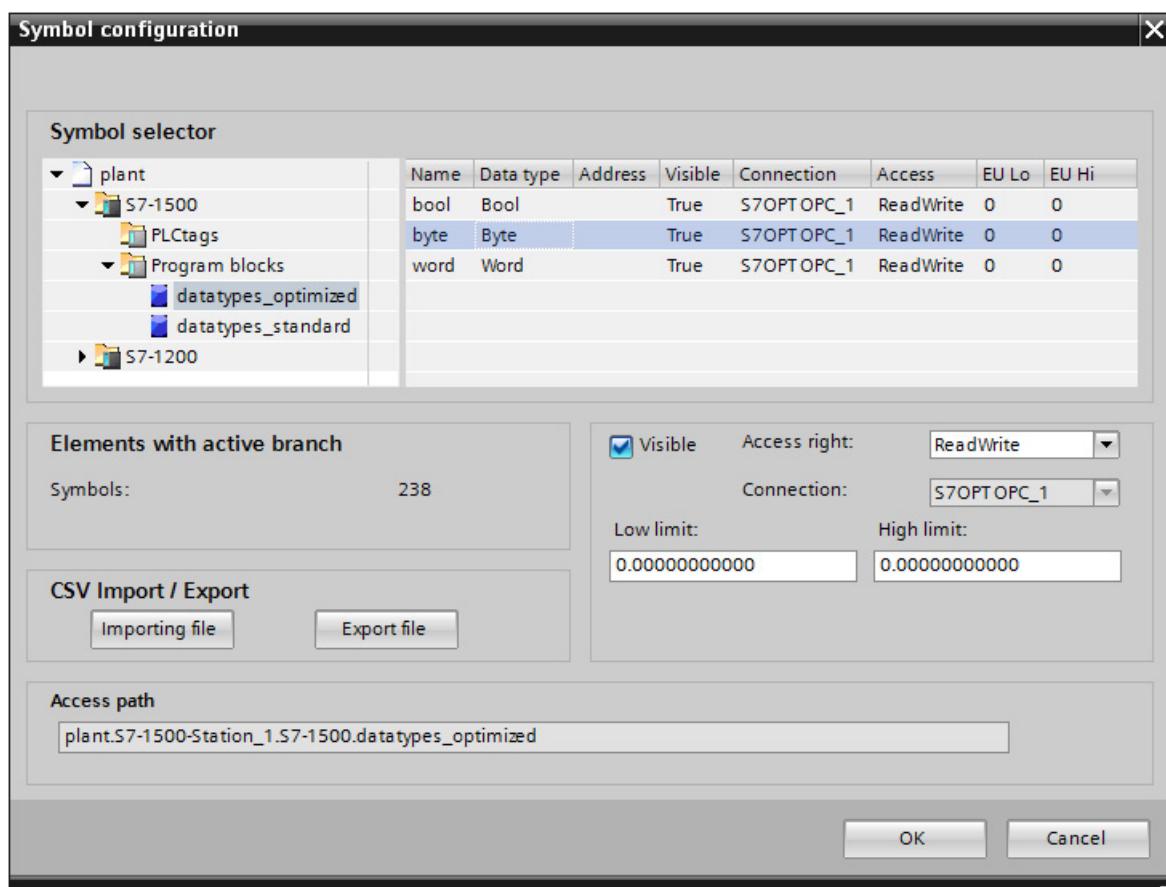


图 2-44 STEP 7 Professional (TIA Portal) 中 OPC 服务器的符号组态

## 2.8.13 S7 连接的块信息对象

### 2.8.13.1 长度信息

#### S7OPT 连接对象下的块对象

在运行期间对 S7 可编程控制器中块结构的类型和大小进行计算。只能获取 S7-1200/S7-1500 站非优化块与此相关的详细信息。在 S7 连接对象下，以下块对象会为 OPC UA 客户端提供这方面的信息。

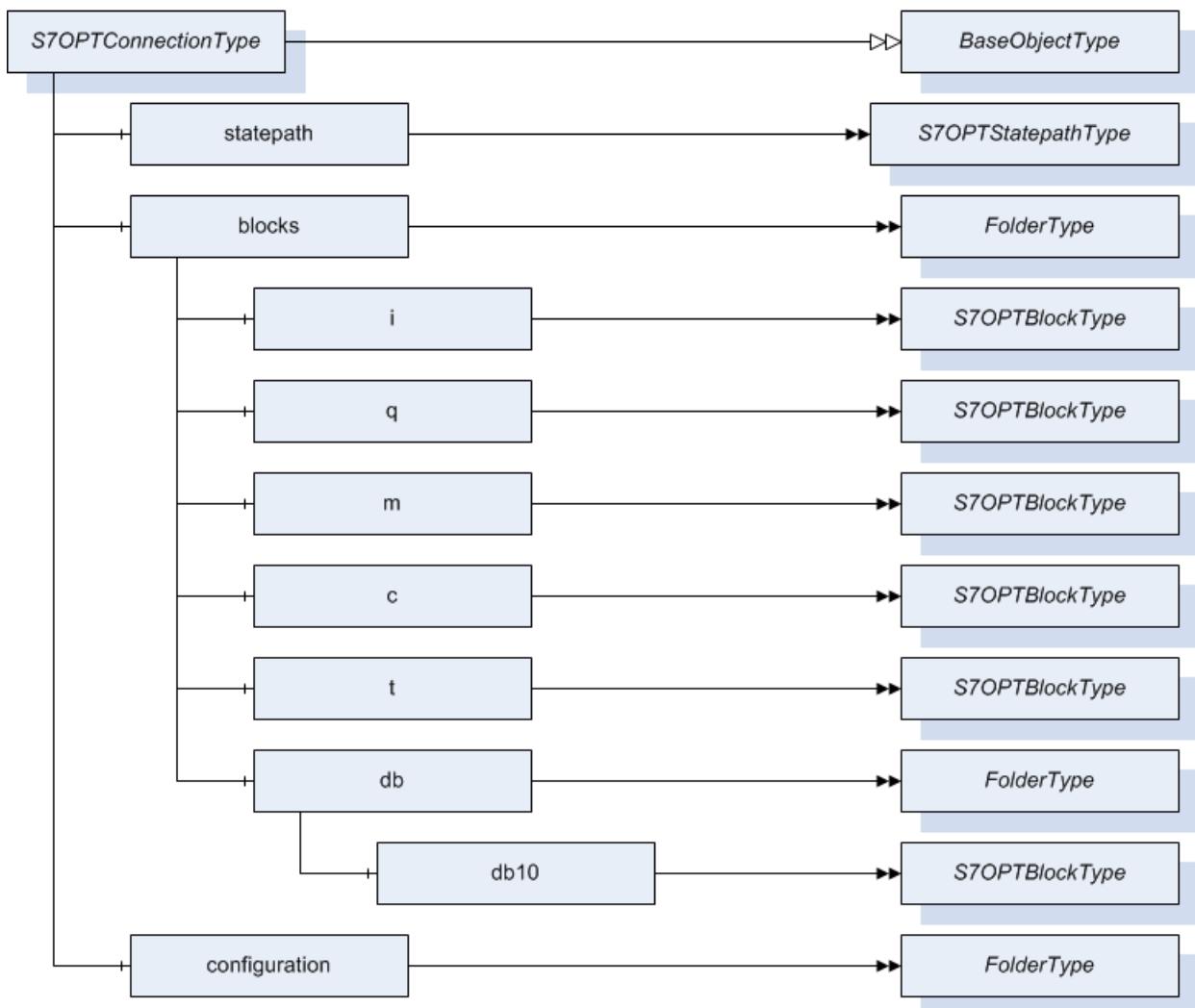


图 2-45 S7 连接对象下的块对象

每个块对象的 OPC UA 属性会提供有关块长度/大小的信息。

**示例：**

命名空间 URI: S7OPT: (-->命名空间索引: 3)

*S7-connection\_1.db10.length* //块 DB10 的长度属性 (长度单位为字节)

**说明**

长度信息字段将显示标准访问调用的数据块。

**2.8.13.2 模板对象****块对象的模板对象**

对于浏览过程中显示的各个块对象，将生成模板数据变量，其 **NodeId** 可用作用户进一步自定义数据对象的模板。

相关块对象的模板数据变量采用标准数据类型 B (或 c、或 tbcd)，起始地址始终为 0 (标准访问的数据块)。

如果此项在连接伙伴中不可访问，则相应的访问结果和质量标准指示相关信息。

**示例：**

命名空间 URI: S7OPT: (命名空间索引: 3)

*S7\_connection\_1.db10.0,b*

*S7\_connection\_1.m.0,b*

**2.8.13.3 诊断和组态信息****S7OPT 连接对象的属性**

通常，由 STEP 7 Professional (TIA Portal) 组态工具对 S7 连接的属性进行组态。运行过程中，评估某些组态参数会非常有用。

为 OPC UA 提供了部分组态参数作为 S7OPT 连接对象的属性：

**诊断和组态信息的语法**

命名空间 URI: S7OPT: (-->命名空间索引: 3)

*<connectionname>.<S7OPTconnectionproperty>*

```
<S7OPTconnectionproperty>:= "vfd"|"cp"|"remoteaddress"|"connect"|"modelversionid"
                                |
                                "fastconnectionstatereturnenable"|"connecttimeout"|
                                "timeout"|"abortconnectionafter"|"optimizebuffer"|
                                "defaultalarmseverity"
```

S7 连接诊断属性（只读）	说明
vfd	连接所分配到的 OPC 服务器的名称。对于在 TIA Portal 中组态的 STEP 7 连接，名称通常为“OPC Server_1”。 数据类型字符串，只读。
cp	连接所分配到的接口参数分配的名称。 数据类型字符串，只读。
remoteaddress	连接伙伴的地址。 数据类型字符串，只读。 连接伙伴的地址是一个数据缓冲区，其数据长度取决于连接类型。 为使其更便于管理，以供用户进行评估，数据缓冲区以字符串格式显示。 IP 地址 (ISO-on-TCP) 格式：“ddd.ddd.ddd.ddd”（每组 1-3 个十进制数） 或使用 MAC 地址
modelversionid	连接伙伴的类型和版本 数据类型为 UInt32，只读。
connect	连接建立的类型。 数据类型 UInt32，只读。
	1 主动，仅根据需要建立连接，如果在等待时间过后还不使用，则终止连接。
	2 主动，连接永久保持为已建立状态。

S7 连接诊断属性 (只读)	说明
fastconnectionstatereturnenable	如果连接中断，则快速返回写入/读取作业。 数据类型布尔型，读写。 <b>True:</b> 启用 (Enabled) <b>False:</b> 禁用 (Disabled)
connecttimeout	连接建立超时 (ms) 数据类型为 UInt32，读写。 0: 未超时 >0: 超时 (ms)
timeout	生产通信作业超时 (ms)。 数据类型为 UInt32，读写。 0: 未超时 >0: 超时 (ms)
abortconnectionafter	自动终止连接。 自动终止连接的延迟时间： 如果在延迟时间期间没有其它的变量访问操作，则延迟时间结束后，OPC 服务器会自动终止连接。 这样可在变量访问间隔时间较长的情况下减少所需的连接数量。 数据类型为 UInt32，读写。 0: 不终止 >0: 终止前的空闲时间 (ms)
optimizebuffer	使用标准访问数据块通信块的优化缓冲区的大小 数据类型为 UInt16，读写。 0: 不优化 >0: 优化缓冲区的大小 (字节)
defaultalarmseverity	statepath 报警的默认优先级。 数据类型为 UInt16，读写。 1: 低优先级 ... 1000: 高优先级

## 2.8.14 S7OPT OPC UA 模板数据变量

借助 OPC UA S7OPT

协议的过程变量，您可以通过灵活的设置选项获取所需数据格式的设备“非优化”过程数据。

不过，在完全可浏览的命名空间中，不能将各种寻址选项放在一起。

即使通过标准访问调用的单字节长度的数据块也有大约 40 个不同的数据格式选项 - 从 **Byte** 和 **SByte** 开始，由这些数据类型的一个元素组成的数组、单个位、**最多 8** 个数组元素的位数组，每个元素均起始于不同的位偏移。

因此，OPC UA 服务器支持 S7OPT 命名空间中具有所谓模板数据变量的用户。

在典型的 OPC 客户端文本输入框中，只需更改几个字符，就可将这些模板转换为有效的 **ItemID**。

**示例：**

```
S7_connection1.db<db>.<o>, dw
```

将 **<db>** 替换为块数据块编号，将 **<o>** 替换为数据块中的偏移量，即可获得有效 **NodeID**。

```
-> S7_connection1.db10.4, dw
```

此原理的优点在于，几乎所有 OPC UA  
客户端都可以使用，无需对客户端进行任何调整。

### 说明

以在“通信设置”组态程序中启用和禁用 OPC UA S7OPT 模块数据变量：在“OPC 协议选择”(OPC protocol selection) 中单击“优化 S7” (S7 optimized) 旁的箭头符号。

### 浏览层级结构中的模板数据变量

在命名空间表示中相应的文件夹旁将模板数据变量进行排序以便需要时便于使用。

## 模板数据变量部分属性的特殊用法

UA 规范已规定了 OPC UA 属性的用途，无需更多说明。

数据块模板字节变量示例：

Nodeld: S7\_connection1.db<db>.<o>,b

浏览名称： 模板字节

说明： <db> 数据块的地址

<o> 数据块内的偏移量

## 2.8.15 OPC UA 事件、条件和报警

### 2.8.15.1 有哪些 OPC UA 报警？

S7OPT OPC UA 服务器支持以下 OPC UA 事件和报警类型：

- 状态路径报警  
与 S7 连接状态相关的报警
- 一致性报警  
与一致性报警组态相关的报警
- 程序报警（块相关 PLC 报警）  
“Program\_Alarm”报警块可从信号变化生成带或不带强制确认的 PLC 报警， 报警中最多可带有十个关联值。

---

#### 说明

S7OPT OPC UA 服务器仅支持 S7-1500 的程序报警。

---

---

#### 说明

输出程序报警之前，需要在 SIMATIC STEP 7 Professional (TIA Portal) 中组态这些报警。它们在编程编辑器中创建并在报警编辑器中进行处理。  
请参见系统手册“SIMATIC STEP 7 Professional”或 SIMATIC STEP 7 Professional (TIA Portal) 中的信息系统。

---

### 2.8.15.2 什么是 OPC UA 事件？

设备的特点在于其硬件和软件组件的状态。通过 UA 报警，您可选择将从状态中选择的状态更改作为事件报告给注册用户。事件的相关信息存储在其属性中。事件类型定义了事件所具有的属性。

事件类型具有自有的属性或继承自其它事件类型（也可能相互继承属性）的属性。

属性简单继承的选项构成了事件类型体系。UA

报警规范包括以预定义类型体系构建的多种预定义事件类型。UA

报警规范还包括对属性类型和这些属性语义的相关规定。

所有预定义事件均位于命名空间 ns="http://opcfoundation.org/UA/" 中。

### 2.8.15.3 S7OPT OPC UA 服务器支持哪些 S7OPT 事件类型？

#### S7OPT OPC UA

服务器基于预定义事件类型，并且可以从预定义事件类型中派生出自己的事件类型。为 S7OPT 事件定义单独的事件类型。所有 S7OPT 事件类型均位于命名空间 ns="S7OPTTYPES:" 中。

S7OPT UA 报警用来表示 PLC 报警。下表显示了 S7OPT OPC UA

服务器可报告的事件类型。除前两种 S7OPT

事件类型外，仅在接收到已组态的程序报警后，才会发生指定事件类型的事件。

S7OPT 事件类型的 Nodeld 和显示名称	S7OPT 事件类型的含义
ns= S7OPTTYPES:, i=14 “S7OPTStatepathAlarmType”	“S7OPTStatepathAlarmType”可映射 S7 连接的反向状态（建立 S7 连接（“Up”）时为“未激活”；未建立 S7 连接（“Down”）时为“激活”）。状态只在 PC 端获得，因而在 PLC 没有物理连接时也可进行报告。
ns= S7OPTTYPES:, i=15 “S7OPTConsistencyAlarmType”	“S7OPTConsistencyAlarmType”显示不一致的报警组态。

S7OPT 事件类型的 NodeId 和显示名称	S7OPT 事件类型的含义
ns= S7OPTTYPES:, i=43 “S7OPTOffNormalAlarmType”	带或不带强制确认的已组态程序报警。 此类型的程序报警具有一个状态或条件。 这样可以采用状态“进入”或“离开”。
ns= S7OPTTYPES:, i=61 “S7OPTInfoReportEventType”	仅可用作信息的已组态程序报警。 要使用这些报警 必须在 SIMATIC STEP 7 Professional (TIA Portal) 的报警编辑器中选择“仅信息”(Information only) 列的复选框。

#### 2.8.15.4 S7OPT OPC UA 服务器的事件类型体系

S7OPT OPC UA 服务器的事件类型体系由标准事件类型体系构成，其中包含一些源于 S7OPT 事件类型的事件类型。

可使用 OPC Scout V10 浏览事件类型体系。

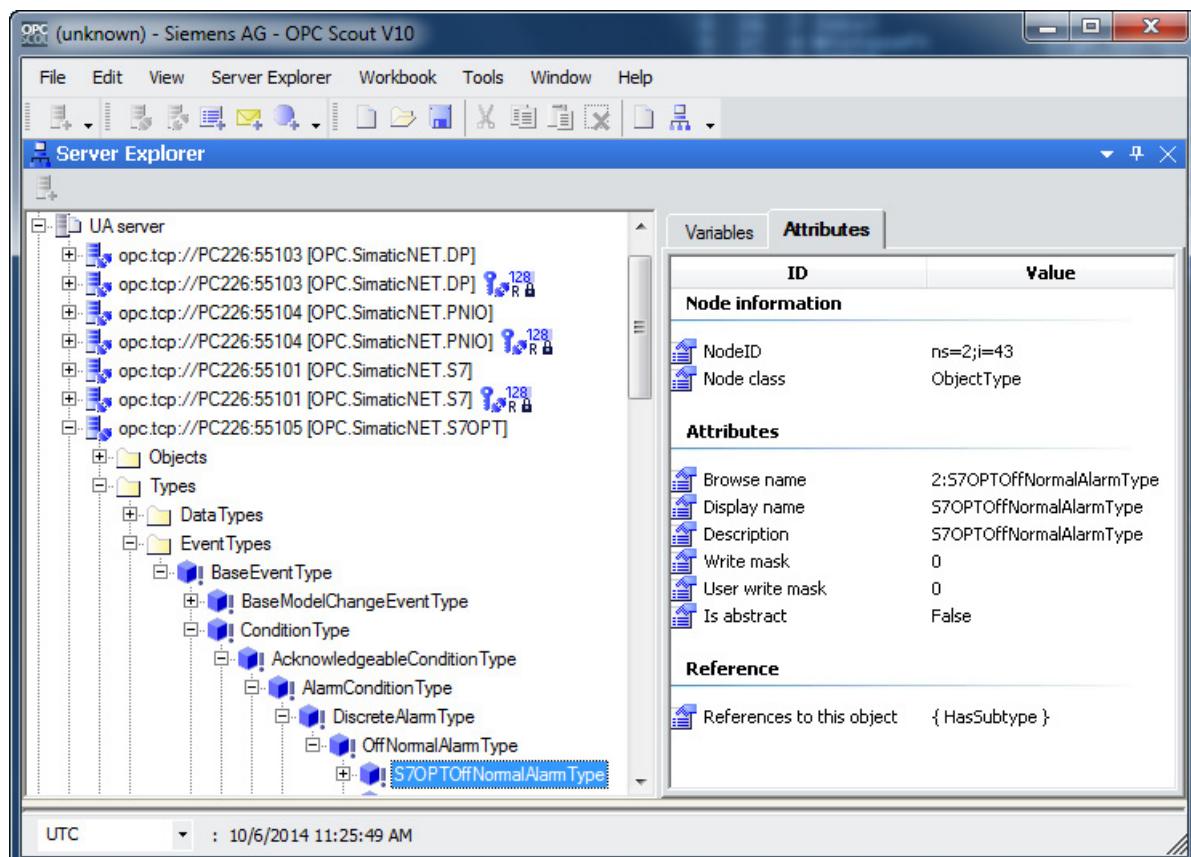


图 2-46 映射标准事件类型体系

标准事件类型的继承如下：

带以下显示名称的标准事件类型	继承自...
“ConditionType”	“BaseEventType”
“AcknowledgeableConditionType”	“ConditionType”
“AlarmConditionType”	“AcknowledgeableConditionType”
“DiscreteAlarmType”	“AlarmConditionType”
“OffNormalAlarmType”	“DiscreteAlarmType”
“SystemOffNormalAlarmType”	“OffNormalAlarmType”

显示名称为“BaseEventType”的标准事件类型是派生出所有事件类型的源类型。

此类型可定义所有事件及其行为中使用的所有属性。

事件类型具有一个数字 **NodeId**（例如

`ns="http://opcfoundation.org/UA/", i=2041; 显示名称 = "BaseEventType" )。`

S7OPT 事件类型从标准事件类型的继承如下：

带以下显示名称的 S7OPT 事件类型	继承自...
“S7OPTStatepathAlarmType”	“SystemOffNormalAlarmType”
“S7OPTConsistencyAlarmType”	“SystemOffNormalAlarmType”
“S7OPTOffNormalAlarmType”	“OffNormalAlarmType”
“S7OPTInfoReportEventType”	“BaseEventType”

在 S7OPT-UA 报警中，S7OPT

事件类型“S7OPTStatepathAlarmType”、“S7OPTConsistencyAlarmType”和“S7OPTOffNormalAlarmType”将实例化。因此，PLC 报警与其所有属性都会映射到报警对象。

也可通过“数据访问”读取或监视相关实例的属性。

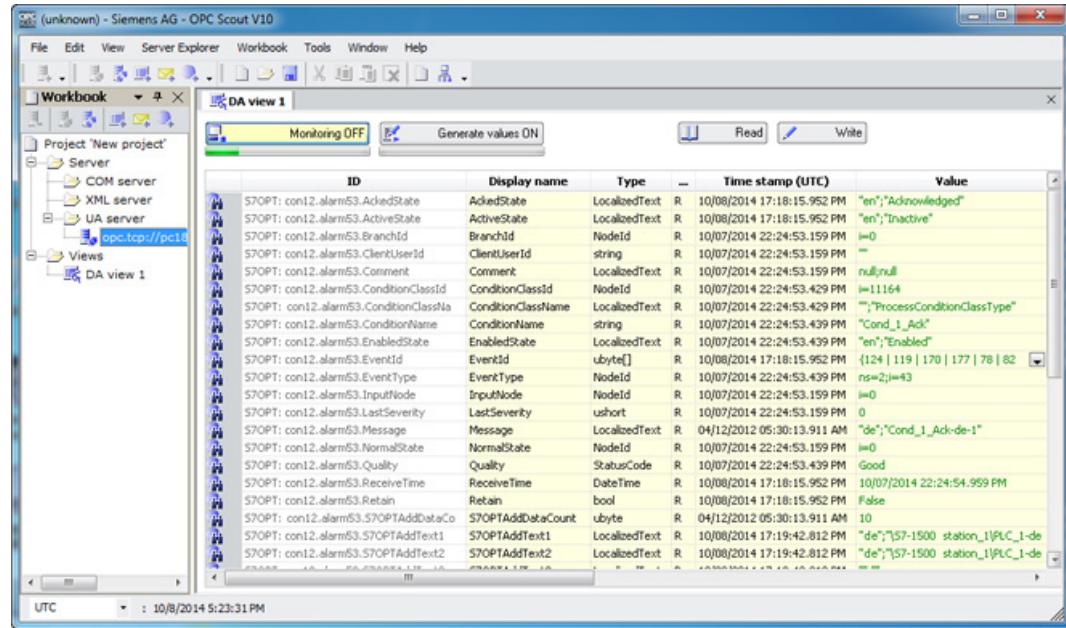


图 2-47 通过数据访问监视报警对象

“S7OPT 事件类型 (页 309)”和“区域树和源空间 (页 316)”部分介绍了编写 UA 应用程序时与用户相关的属性。这些属性根据定义它们时使用的事件类型来分组。引用属性时，将使用数据类型“QualifiedNames”。“QualifiedNames”包含命名空间和浏览名称（有时为浏览路径）。

不过，文档的上下文中只引用“QualifiedNames”的命名空间与定义类型命名空间相同的属性，因此将忽略命名空间的规范。将指定属性的浏览名称，而非“QualifiedNames”。

除此之外，还会在括号中显示请求的属性（用“|”将属性与数据类型分隔开）。大多数情况下，请求的属性为指定“Value”的数字 13。偶尔会出现数值 1，该值指定 Nodeld。

## 2.8.16 标准事件类型及其属性的使用

### 2.8.16.1 显示名称为“BaseEventType”的标准事件类型

Nodeld: ns="http://opcfoundation.org/UA/", i=2041

源于：不源于任何其它事件类型。

相关属性的浏览名称

- “EventId”(13|ByteString)
- “EventType”(13|Nodeld)
- “SourceNode”(13|Nodeld)
- “SourceName”(13|String)
- “Time”(13|DateTime)
- “ReceiveTime”(13|DateTime)
- “Message”(13|LocalizedText)
- “Severity”(13|UInt16)

直接源自该类型的 S7OPT 事件类型: ns="S7OPTTYPES:", i =61, 显示名称 =“S7OPTInfoReportEventType”。

此事件类型的事件由 SourceName 唯一标识。

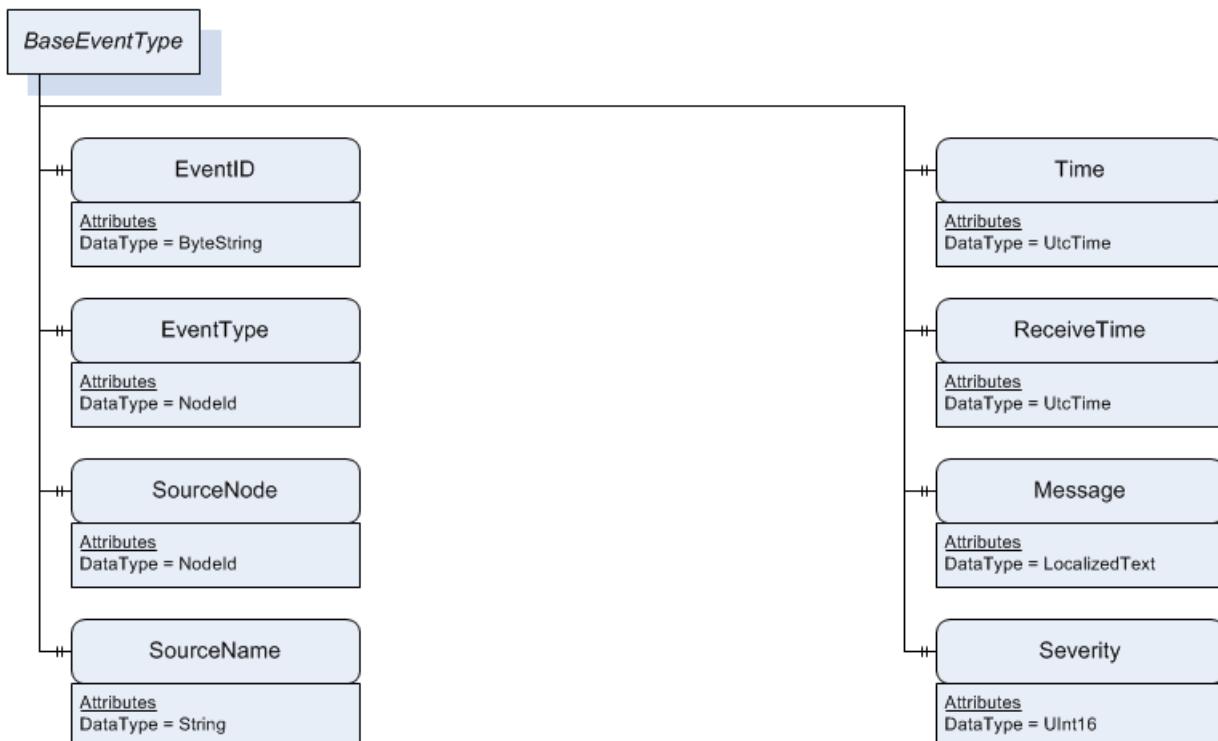


图 2-48 BaseEventType 的表示

**EventId**

唯一标识和引用事件的标识符。客户端需要 EventId 来实现报警确认等操作。

**EventType**

始终为之前列出的 S7OPT 事件类型之一。

**SourceNode**

事件的源节点；在 S7OPT 报警中，由 ns=“S7OPTSOURCES:”中的 Nodeld 指定的对象或 ns=“S7OPT:”中的 S7 连接对象。另请参见“区域树和源空间 (页 316)”部分。

**SourceName**

请参见“构成 SourceName、Message 和 Severity (页 302)”部分。

**Time**

时间戳，尽可能接近过程。这由组态决定，以下选项可用：

- PLC 时间戳
- PLC 时间 + 偏移量
- PC 时间 (UTC)

对于 PC 时间 (UTC)，“Time”与“ReceiveTime”相同。

**ReceiveTime**

PC 的时间戳。

**Message**

请参见“构成 SourceName、Message 和 Severity (页 302)”部分。

**Severity**

请参见“构成 SourceName、Message 和 Severity (页 302)”部分。

### 2.8.16.2 构成 SourceName、Message 和 Severity

某些属性名称由 STEP 7 项目设置。

STEP 7 项目中使用的名称（如 S7 连接名称、站名称、PLC 名称）用于构成属性。

#### SourceName 的构成

S7OPT 事件类型的 NodeId 和显示名称	SourceName 的构成规则
ns= S7OPTTYPES:, i=14 “S7OPTStatepathAlarmType”	S7 连接名称 +“.statepath” 示例: connection2.statepath 存在相应的 SourceNode。
ns= S7OPTTYPES:, i=15 “S7OPTConsistencyAlarmType”	S7 连接名称 +“.statepath” 示例: connection2.statepath 存在相应的 SourceNode。
ns= S7OPTTYPES:, i=43 “S7OPTOffNormalAlarmType”	已组态的程序报警: SourceName 包含 站名称（如“station_1”） +“\ PLC 名称（如“PLC_1516”） +“\ PLC 报警的背景数据块的符号名称（如“motor1”） 示例: “station_1\PLC_1516\motor1”
ns= S7OPTTYPES:, i=61 “S7OPTInfoReportEventType”	已组态的程序报警: SourceName 包含 站名称（如“station_1”） +“\ PLC 名称（如“PLC_1516”） +“\ PLC 报警的背景数据块的符号名称（如“motor1”） 示例: “station_1\PLC_1516\motor1”

## Message 的构成

S7OPT 事件类型的 Nodeld	Message 的构成规则
ns= S7OPTTYPES:, i=14 “S7OPTStatepathAlarmType”	“statepath”
ns= S7OPTTYPES:, i=15 “S7OPTConsistencyAlarmType”	“consistency”
ns= S7OPTTYPES:, i=43 “S7OPTOffNormalAlarmType”	已组态的程序报警： 程序报警的报警文本。 报警文本可包含动态参数（变量和文本列表）。
ns= S7OPTTYPES:, i=61 “S7OPTInfoReportEventType”	已组态的程序报警： 程序报警的报警文本。 报警文本可包含动态参数（变量和文本列表）。

## 严重程度的构成

S7OPT 事件类型的 Nodeld 和显示名称	Severity 的构成规则
ns= S7OPTTYPES:, i=14 “S7OPTStatepathAlarmType”	每个 S7 连接检测到的默认优先级。
ns= S7OPTTYPES:, i=15 “S7OPTConsistencyAlarmType”	每个 S7 连接检测到的默认优先级。
ns= S7OPTTYPES:, i=43 “S7OPTOffNormalAlarmType”	已组态的程序报警： 源于程序报警的 PLC 报警优先级（0 到 16）。
ns= S7OPTTYPES:, i=61 “S7OPTInfoReportEventType”	已组态的程序报警： 源于程序报警的 PLC 报警优先级（0 到 16）。

表格 2-6 PLC 报警优先级 - 严重程度转换表

PLC 报警优先级	严重程度
0	1
1	63
2	125

PLC 报警优先级	严重程度
3	188
4	250
5	313
6	375
7	438
8	500
9	563
10	625
11	688
12	750
13	813
14	875
15	938
16	1000

### 2.8.16.3 显示名称为“ConditionType”的标准事件类型

NodeId: ns:"http://opcfoundation.org/UA/", i=2782

源自: ns="http://opcfoundation.org/UA", i=2041, 显示名称为“BaseEventType”

相关属性的浏览名称

"" (1|NodeId)

“ConditionName”(13|String)

“EnableState|ID”(13|Boolean)(Browse-Path)

“EnableState”(13|LocalizedText)

“Quality”(13|StatusCode)

直接源自该类型的 S7OPT 事件类型: 无

直接或间接源自该类型的事件类型都有一个条件实例。

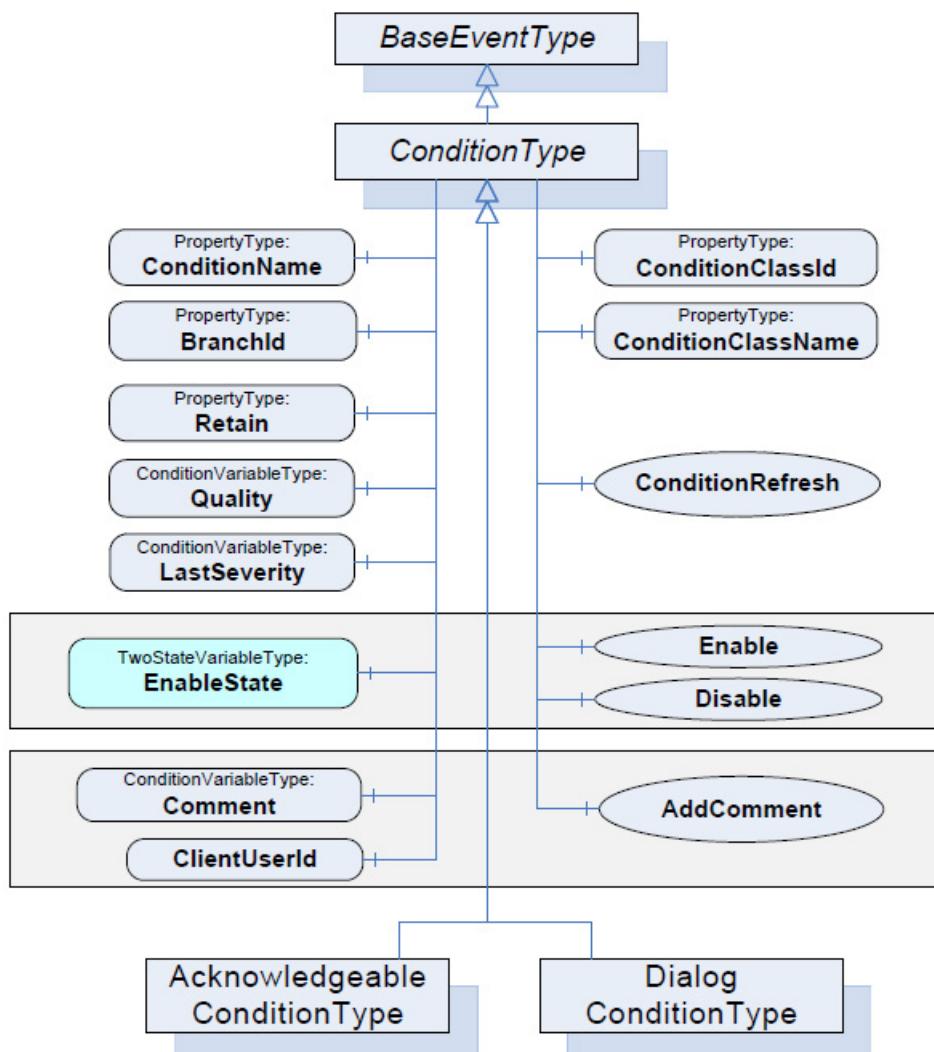


图 2-49 ConditionType 的表示

**ConditionName**

通过“SourceName”和“ConditionName”的组合进行唯一标识的条件事件。

请参见下面的“ConditionName 的构成”部分。

**EnableState**

状态机的一部分。根据 UA 报警规范，EnableState 的可能值为“zh”,“已启用”和“zh”;“已禁用”。

**Quality**

有关可能的质量值，请参见“UA 规范”。

**2.8.16.4 ConditionName 的构成**

某些属性名称由 STEP 7 项目设置。

S7OPT 事件类型的 NodeId 和显示名称	ConditionName 的构成规则
ns=S7OPTTYPES:, i=14 “S7OPTStatepathAlarmType”	“statepath”
ns= S7OPTTYPES:, i=15 “S7OPTConsistencyAlarmType”	“consistency”
ns= S7OPTTYPES:, i=43 “S7OPTOffNormalAlarmType”	已组态的程序报警： 程序报警的名称 示例： “MyAlarm”
ns= S7OPTTYPES:, i=61 “S7OPTInfoReportEventType”	已组态的程序报警： 程序报警的名称 示例： “MyAlarm”

### 2.8.16.5 显示名称为“AcknowledgeableConditionType”的标准事件类型

NodeId: ns:"http://opcfoundation.org/UA/", i=2881

源自: ns="http://opcfoundation.org/UA", i=2782, 显示名称为“ConditionType”

相关属性的浏览名称

“AckedState|Id”(13|Boolean)

“AckedState”(13|LocalizedText)

直接源自该类型的 S7OPT 事件类型: 无

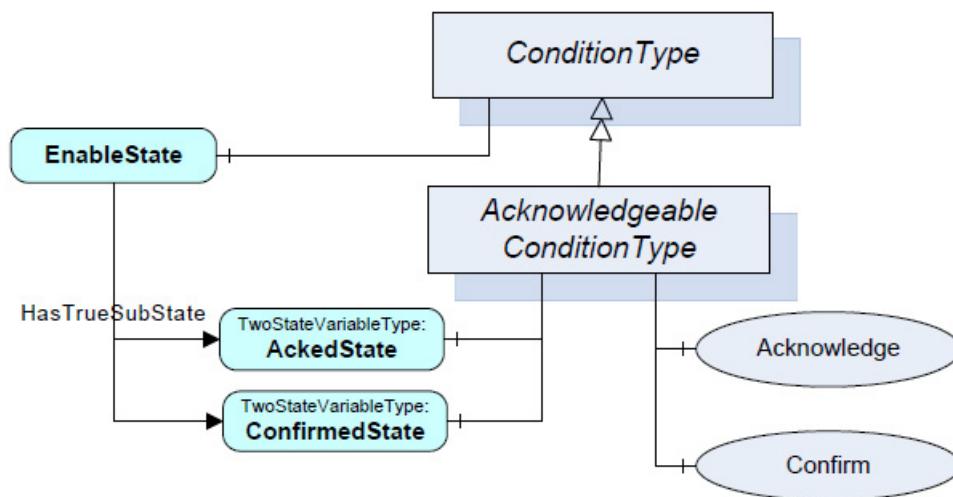


图 2-50 AcknowledgeableConditionType 的表示

### AckedState

状态机的一部分。可能的值为“zh”;“已确认”和“zh”;“未确认”。AckedState 中的每次值更改都会报告一个事件。报告为

AckedState“zh”,“未确认”的事件都必须使用确认功能确认。请参见系统手册“SIMATIC STEP 7 Professional”或 SIMATIC STEP 7 Professional (TIA Portal) 中的信息系统。

### 2.8.16.6 显示名称为“AlarmConditionType”的标准事件类型

NodeId: ns:"http://opcfoundation.org/UA/", i=2915

源自: ns="http://opcfoundation.org/UA", i=2881, 显示名称为

“AcknowledgeableConditionType”

相关属性的浏览名称

“ActiveState|Id”(13|Boolean)

“ActiveState”(13|LocalizedText)

直接源自该类型的 S7OPT 事件类型: 无

根据 UA 报警规范，ActiveState 的可能值为“zh”;“激活”和“zh”;“未激活”。

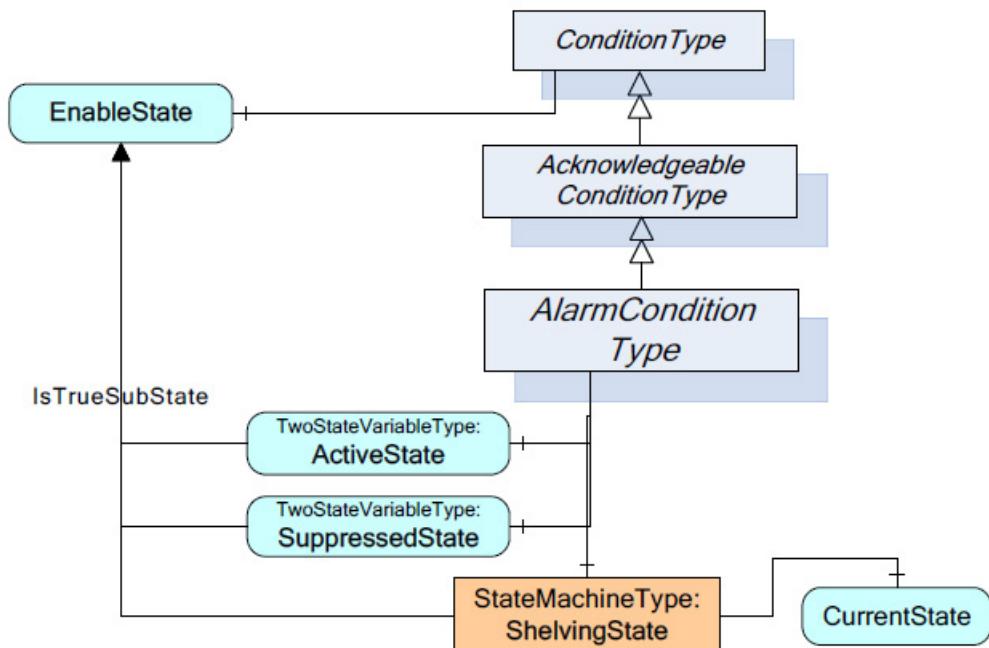


图 2-51 AlarmConditionType 的表示

## ActiveState

状态机的一部分。在 S7OPT UA 报警中，ActiveState 由 PLC 报警控制。

如果报警触发信号的值为“1”，将报告

ActiveState“激活”，如果值为“0”，将报告“未激活”。ActiveState 中的每次值更改都会报告一个事件。

### 事件类型显示名称为“S7OPTStatepathAlarmType”的 ActiveState

在此，未建立 S7 连接时达到“激活”状态。建立 S7 连接后达到“未激活”状态。

#### 2.8.16.7 显示名称为“OffNormalAlarmType”的标准事件类型

Nodeld: ns="http://opcfoundation.org/UA/", i=10637

间接源自：

ns="http://opcfoundation.org/UA/", i=2915, 显示名称为“AlarmConditionType”

相关属性的浏览名称：无

直接源自该类型的 S7OPT 事件类型：

ns=S7OPTTYPES:, i =14, 显示名称=“S7OPTStatepathAlarmType”

ns=S7OPTTYPES:, i=15, 显示名称=“S7OPTConsistencyAlarmType”

ns=S7OPTTYPES:, i=43, 显示名称=“S7OPTOffNormalAlarmType”

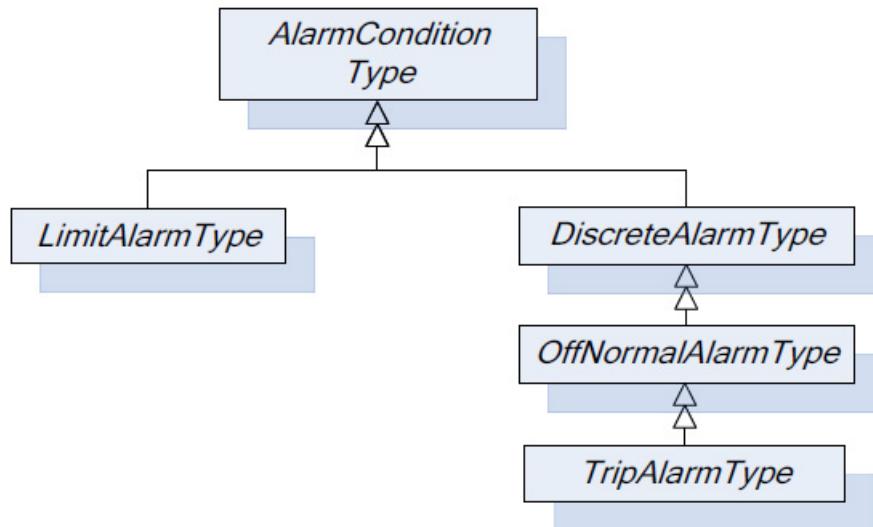


图 2-52 报警类型体系中的 OffNormalAlarmType 布局

## 2.8.17 S7OPT 事件类型

### 2.8.17.1 显示名称为“S7OPTStatepathAlarmType”的 S7OPT 事件类型

Nodeld: ns="S7OPTTYPES:", i=14

间接源自:

ns="http://opcfoundation.org/UA/", i=2915, 显示名称为“AlarmConditionType”

相关属性的浏览名称:

“S7OPTConnection”(13|Nodeld)

该 S7OPT 事件类型可映射 S7 连接的反向状态（建立 S7

连接 (“Up”) 时为“未激活”；未建立 S7 连接 (“Down”或“RECOVERY”) 时为“激活”）。

状态只在 PC 端获得，因而在 PLC 没有物理连接时也可进行报告。

在“激活”状态下，不会再收到 PLC 报警。

#### S7OPTConnection

指定接收 PLC 报警的连接的 Nodeld。

示例: ns="S7OPT:", s="connectionname"。

### 2.8.17.2 显示名称为“S7OPTConsistencyAlarmType”的 S7OPT 事件类型

Nodeld: ns="S7OPTTYPES:", i=15

间接源自:

ns="http://opcfoundation.org/UA/", i=2915, 显示名称为“AlarmConditionType”

相关属性的浏览名称:

“S7OPTConnection”(13|Nodeld)

S7OPT 事件类型“S7OPTConsistencyAlarmType”不需要确认。它会显示 PLC 与 PC 站之间不一致的报警组态。这些不一致的 PLC 报警不会报告给 OPC 客户端。

这种情况下，会针对此 PLC 将条件实例“consistency”设置为“激活”（一次）。

建议首先更新 PLC 上的报警组态，然后更新 PC 站上的报警组态。

#### S7OPTConnection

指定接收 PLC 报警的连接的 Nodeld。

示例: ns="S7OPT:", s="connectionname"。

### 2.8.17.3 显示名称为“S7OPTOffNormalAlarmType”的 S7OPT 事件类型

NodeId: ns="S7OPTTYPES:", i=43

间接源自: ns="http://opcfoundation.org/UA/", i=10637, 显示名称为  
“OffNormalAlarmType”

相关属性的浏览名称:

“S7OPTAlarmId”(13|UInt32)  
“S7OPTConnection”(13|NodeId)  
“S7OPTTime”(13|DateTime)  
“S7OPTDisplayClass”(13|UInt16)  
“S7OPTAlarmClass”(13|String)  
“S7OPTInfoText”(13|LocalizedText)  
“S7OPTAddDataCount”(13|Byte)  
“S7OPTAddData1|Datavalue”(13|Variant)  
“S7OPTAddData2|Datavalue”(13|Variant)  
“S7OPTAddData3|Datavalue”(13|Variant)  
“S7OPTAddData4|Datavalue”(13|Variant)  
“S7OPTAddData5|Datavalue”(13|Variant)  
“S7OPTAddData6|Datavalue”(13|Variant)  
“S7OPTAddData7|Datavalue”(13|Variant)  
“S7OPTAddData8|Datavalue”(13|Variant)  
“S7OPTAddData9|Datavalue”(13|Variant)  
“S7OPTAddData10|Datavalue”(13|Variant)  
“S7OPTAddText1”(13|LocalizedText)  
“S7OPTAddText2”(13|LocalizedText)  
“S7OPTAddText3”(13|LocalizedText)  
“S7OPTAddText4”(13|LocalizedText)  
“S7OPTAddText5”(13|LocalizedText)  
“S7OPTAddText6”(13|LocalizedText)  
“S7OPTAddText7”(13|LocalizedText)  
“S7OPTAddText8”(13|LocalizedText)  
“S7OPTAddText9”(13|LocalizedText)

此 S7OPT

事件类型可映射带或不带强制确认的已组态“程序报警”，报警中最多可带有十个关联值。

## 2.8 通过 OPC UA 进行的优化 S7 通信

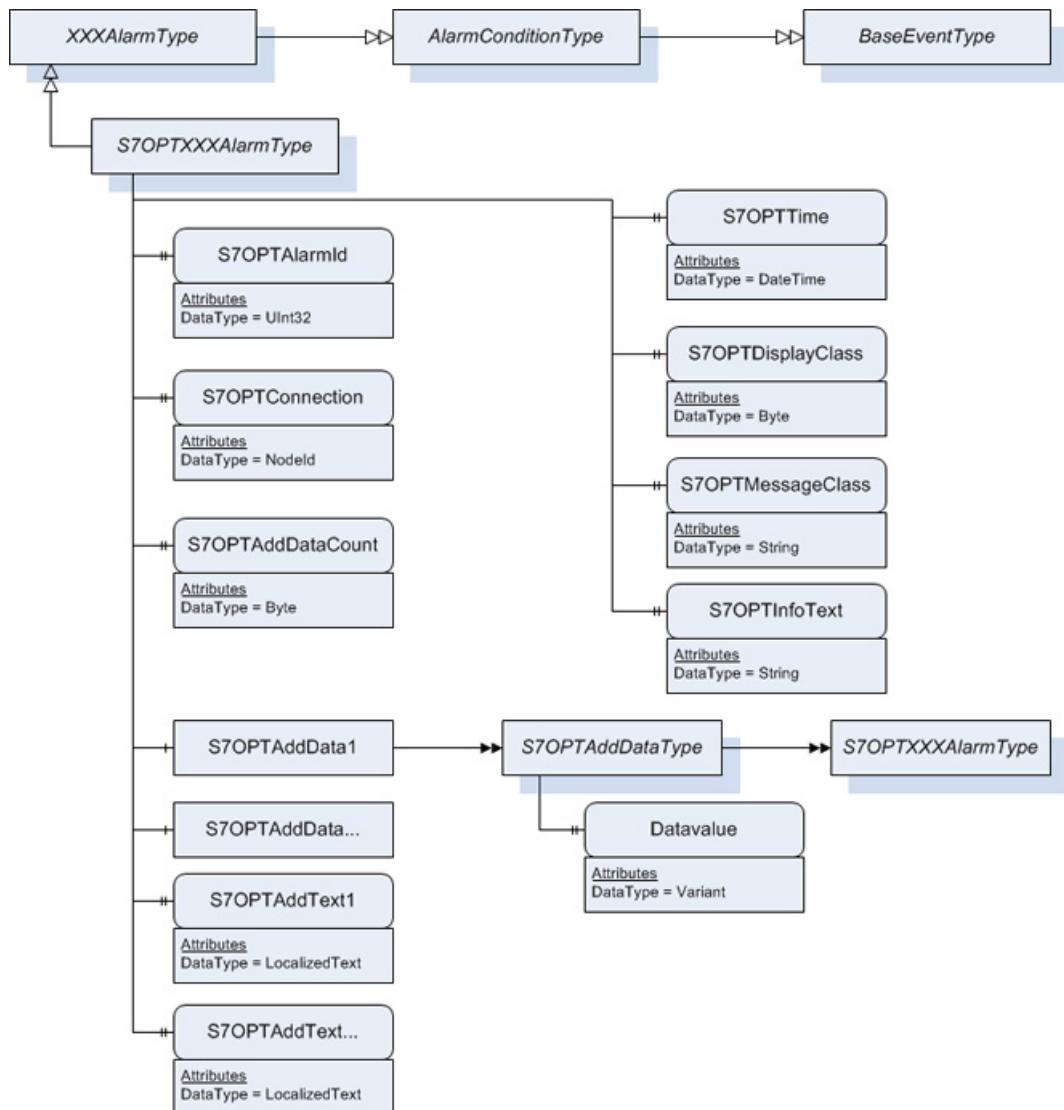


图 2-53 S7OPTOffNormalAlarmType 的 S7 特定属性的表示

### S7OPTAlarmId

PLC 报警的报警编号。 编号在整个 PLC 中是唯一的。

### S7OPTConnection

指定接收 PLC 报警的连接的 NodId。

示例： ns="S7OPT:", s="connectionname"。

**S7OPTTime**

该时间戳指示 PLC 上发生 PLC 报警的时间（PLC 时间）。

**S7OPTDisplayClass**

显示 PLC 报警的等级。

**S7OPTAlarmClass**

PLC 报警的报警等级。 报警等级可确定报警是否需要确认。

**S7OPTInfoText**

组态中 PLC 报警的信息文本。 信息文本可包含动态参数（变量和文本列表）。

**S7OPTAddDataCount**

PLC 报警的实际关联值数。

**S7OPTAddData[n]||Datavalue**

生成 PLC 报警的关联值。

$1 \leq n \leq 10$ 。

**S7OPTAddText[n]**

$1 \leq n \leq 9$ 。组态中 PLC 报警的附加文本。

#### 2.8.17.4 显示名称为“S7OPTInfoReportEventType”的 S7OPT 事件类型

Nodeld: ns="S7OPTTYPES:", i=61

间接源自: ns="http://opcfoundation.org/UA/", i=2041, 显示名称为“BaseEventType”

相关属性的浏览名称:

- “S7OPTInfoReportId”(13|UInt32)
- “S7OPTInfoReportName”(13|String)
- “S7OPTConnection”(13|Nodeld)
- “S7OPTTime”(13|DateTime)
- “S7OPTDisplayClass”(13|UInt16)
- “S7OPTAlarmClass”(13|String)
- “S7OPTInfoText”(13|LocalizedText)
- “S7OPTAddDataCount”(13|Byte)
- “S7OPTAddData1|Datavalue”(13|Variant)
- “S7OPTAddData2|Datavalue”(13|Variant)
- “S7OPTAddData3|Datavalue”(13|Variant)
- “S7OPTAddData4|Datavalue”(13|Variant)
- “S7OPTAddData5|Datavalue”(13|Variant)
- “S7OPTAddData6|Datavalue”(13|Variant)
- “S7OPTAddData7|Datavalue”(13|Variant)
- “S7OPTAddData8|Datavalue”(13|Variant)
- “S7OPTAddData9|Datavalue”(13|Variant)
- “S7OPTAddData10|Datavalue”(13|Variant)
- “S7OPTAddText1”(13|LocalizedText)
- “S7OPTAddText2”(13|LocalizedText)
- “S7OPTAddText3”(13|LocalizedText)
- “S7OPTAddText4”(13|LocalizedText)
- “S7OPTAddText5”(13|LocalizedText)
- “S7OPTAddText6”(13|LocalizedText)
- “S7OPTAddText7”(13|LocalizedText)
- “S7OPTAddText8”(13|LocalizedText)
- “S7OPTAddText9”(13|LocalizedText)

此 S7OPT 事件类型可映射仅作信息用途、带十个关联值的已组态程序报警。

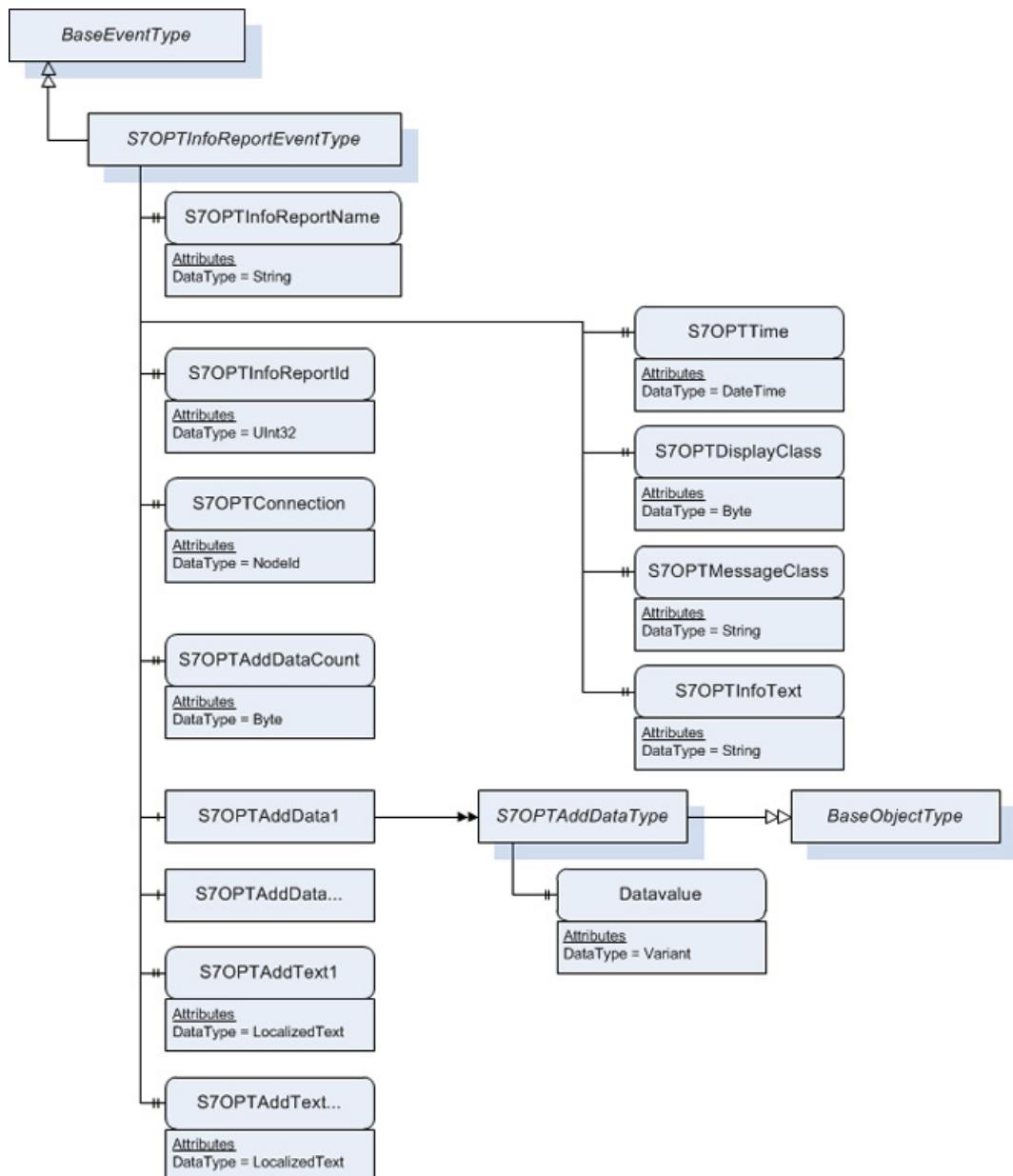


图 2-54 S7OPTInfoReportEventType 的 S7 特定属性的表示

### S7OPTInfoReportId

PLC 报警的报警编号。 编号在整个 PLC 中是唯一的。

### S7OPTInfoReportName

由“inforeport”+ PLC 报警的报警编号构成。

示例： inforeport56

### S7OPTConnection

指定接收 PLC 报警的连接的 NodeId。

示例： ns="S7OPT:", s="connectionname"。

### S7OPTTime

该时间戳指示 PLC 上发生 PLC 报警的时间。

### S7OPTDisplayClass

显示 PLC 报警的等级。

### S7OPTAlarmClass

PLC 报警的报警等级。 报警等级可确定报警是否需要确认。

### S7OPTInfoText

组态中 PLC 报警的信息文本。 信息文本可包含动态参数（变量和文本列表）。

### S7OPTAddDataCount

PLC 报警的实际关联值数。

### S7OPTAddData[n]|Datavalue

生成 PLC 报警的关联值。

1 <= n <=10。

### S7OPTAddText[n]

1 <= n < =9。 组态中 PLC 报警的附加文本。

## 2.8.18 区域树和源空间

对于 S7OPT OPC UA 服务器，区域树和源空间包含已组态 PLC 报警的 SourceNode。

S7 UA 报警可建立区域树以向区域分配 SourceNode。

区域树的节点为映射工厂区域的特殊 UA 文件夹。这些特殊的 UA 文件夹称为区域节点。

STEP 7 项目的路径名称被用作区域树。由项目导航内的以下组件构成：

- 站名称
- PLC 名称
- 组

用户自定义组可对现有客户工厂进行拓扑映射。它们可以插入 STEP 7 项目的项目树中。

站名称和 PLC 名称为永久性链接，无法将其分开。组只能插入到前后位置。

站名称会自动插入到站名称之前。

示例：

```

plant1 (组)
    station_1 (站名称)
        PLC_1516 (PLC 名称)
            house1 (组)
                motor1 (PLC 报警的背景数据块)
            house2 (组)
                motor2 (PLC 报警的背景数据块)

```

在该示例中，区域树和 SourceNode 的衍生方式如下：

区域树的节点：

- ns="S7OPTAREAS:", s="plant1"。
- ns="S7OPTAREAS:", s="plant1\station\_1"。
- ns="S7OPTAREAS:", s="plant1\station\_1\PLC\_1516"。
- ns="S7OPTAREAS:", s="plant1\station\_1\PLC\_1516\house1"。
- ns="S7OPTAREAS:", s="plant1\station\_1\PLC\_1516\house2"。

SourceNode 的 SourceName 由以下部分组成：

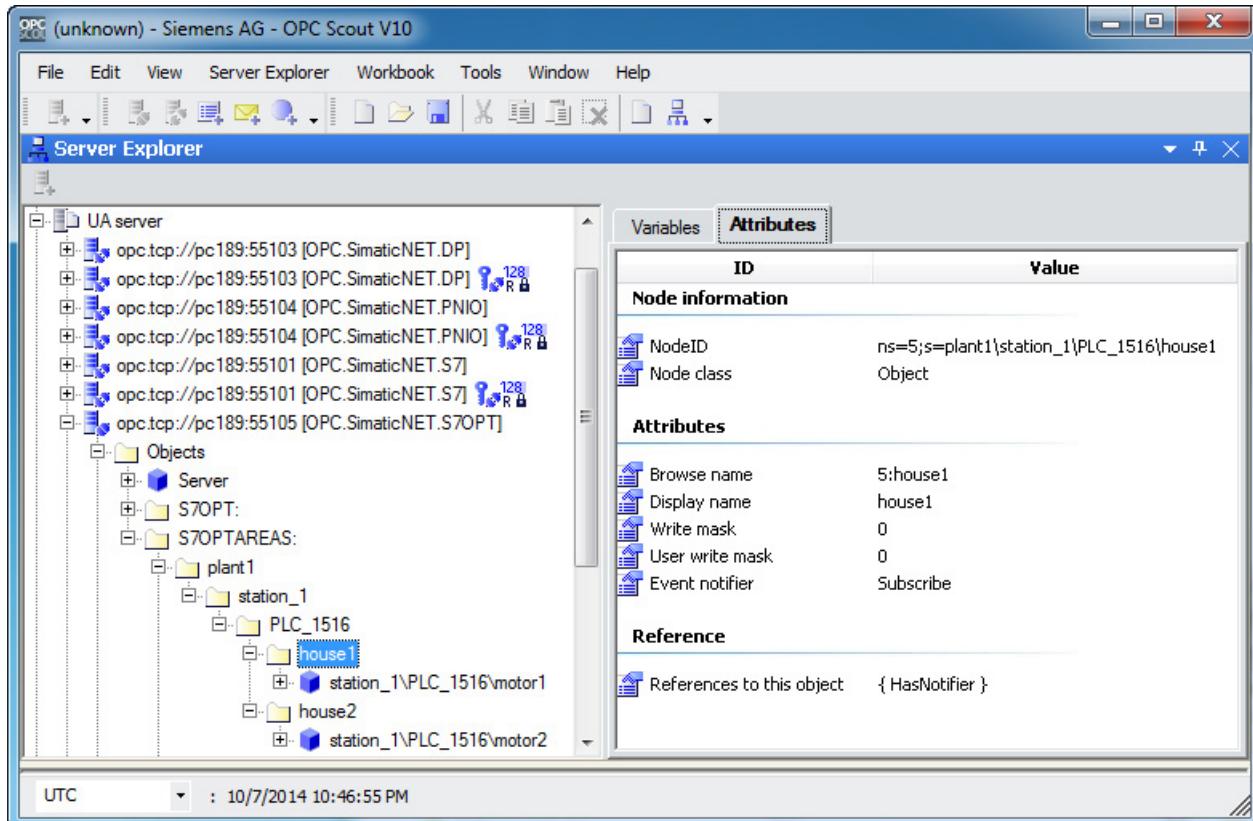
- 站名称 +“\”
- PLC 名称（已组态 PLC 的名称）+“\”
- PLC 报警的背景数据块的符号名称

## 2.8 通过 OPC UA 进行的优化 S7 通信

根据示例：

- ns="S7OPTSOURCES:", s="station\_1\PLC\_1516\motor1"。
- ns="S7OPTSOURCES:", s="station\_1\PLC\_1516\motor2"。

可使用 OPC 客户端（如 OPC Scout V10）浏览区域树和源空间。



在左侧窗口选择显示名称为“house1”（NodeID

ns="S7OPTAREAS:", s="plant1\station\_1\PLC\_1516\house1"）的区域节点。右侧窗口将显示其属性；由此可以看到，区域节点拥有参考“HasNotifier”。

## 2.8.19 接收事件

### EventItems 和 EventNotifier

从服务器接收当前事件的唯一方法就是在订阅中生成一个或多个受监视的事件项。对于事件，EventNotifier 属性 (12) 将受到监视。EventNotifier

仅引用带有“HasNotifier”参考的对象。在 S7OPT OPC UA 报警中，这些是服务器节点 ns="http://opcfoundation.org/UA/"，i=2253，即 S7

连接的所有节点，例如，ns="S7OPT:"，s="S7\_connection\_1"和 ns="S7OPTAREAS:"中的所有区域节点。

如果监视区域节点的

EventNotifier，将报告这个区域发生的所有事件。视特定系统而定（可从与区域空间结合的源空间获取一组潜在可用的 EventSource），可以报告非常多的事件。

这使得选择事件非常必要。

### 过滤事件

除了选择合适的事件项外，在 OPC UA 接口还可以使用特定属性和值过滤事件。

---

#### 说明

事件项的选择和过滤条件的设置可以确定 OPC 客户端是否能收到某个事件。

---

### 选择属性

生成受监视事件项时，必须指定要返回的属性。如果不指定属性，则即便发生事件，也无法确定是哪一个。要将某一属性包括在要返回的属性列表中，必须为各项属性指定：

- 已定义事件类型 (Typeid) 的 Nodeld、
- 属性的浏览名称与可能的浏览路径以及
- Attributeld

在“S7OPT OPC UA 服务器的事件类型体系

(页 297)”部分中，您可以找到已定义事件类型的 Nodeld、浏览名称以及浏览路径和 Attributeld (如果适用)。

事件不需要返回所有的属性。“零”用于返回事件不具有的属性。

对于事件本身不具备或由于浏览名称等的不规范而导致无效的属性，S7OPT OPC UA 报警无法区分。

### 说明

并非所有报警状态的更改都通过 OPC UA

接口上的事件传递。如果报警状态快速变化，事件可能不报告某些状态变化，只发送上一状态和当前状态。这取决于 OPC 参数（如“PublishingInterval”）以及组态和计算机性能。

## 2.8.20 UA 报警方法

### Enable()/Disable()

报警“禁用”的客户端中不会生成事件。即使这样，PLC 的报警也将一直受到监视，因为控制器的报警无法单独“启用”或“禁用”。

#### 注意

#### 禁用事件

采用这些方法时，请注意，一个客户端可以禁用其它客户端的事件。

### AddComment()

可为每个报警实例存储一个注释。

### ConditionRefresh()

报告所有激活或未确认的报警（属性“Retain”= true）。

### Acknowledge()

报警的确认将传送到 PLC。不管怎样，确认状态中的更改都会通过 PLC 显式返回 OPC 服务器，然后在 OPC 端报警将改为已确认状态并触发相应事件。

## 2.9 开放式通信服务 (SEND/RECEIVE)

取决于通信网络，有两个版本的通信网络 (SEND/RECEIVE):

- 通过工业以太网的开放式通信服务 (SEND/RECEIVE)  
协议 ID: SR
- 通过 PROFIBUS 的开放式通信服务 (SEND/RECEIVE)  
协议 ID: FDL

### 2.9.1 通过工业以太网的开放式通信服务 (SEND/RECEIVE)

通过工业以太网的开放式通信服务 (SEND/RECEIVE) 也称为 SEND/RECEIVE 协议。  
允许与 S5 和 S7 设备以及第三方设备进行通信。

#### 通过工业以太网的开放式通信服务 (SEND/RECEIVE) 的属性

- 使用 SIMATIC S5 处理块和 S7 函数块进行通信
- 可在两个 PC 站之间建立链接。
- 支持用于访问伙伴设备对象的 WRITE 和 FETCH 函数。
- 通过 SEND 和 RECEIVE 模式快速访问大型数据包。
- 在数据包内进行交替访问。
- 显示和监视连接状态

#### 2.9.1.1 适用于 SR 协议的功能强大的 SIMATIC NET OPC 服务器

##### 简介

以下部分将介绍一种满足更高性能要求的 SR 协议组态变型。

要使用这种变型，请在进程外 OPC 服务器上加载所有底层 SR  
协议库和用作进程内服务器的 COM 服务器。协议在 OPC

服务器的进程中进行处理，这就避免了在进程与多协议模式之间进行切换所需的额外执行  
时间。不过，OPC 客户端与 OPC 服务器之间的进程切换仍然必不可少。

## 2.9 开放式通信服务 (SEND/RECEIVE)

### 组态

在“通信设置”组态程序中将“SR”协议选作唯一协议可隐式启用此高速变型（如果选择其它协议或 OPC UA 接口，将会失去所述的性能优势）：



图 2-55 “通信设置”组态程序中用于选择 SR 协议的窗口

也可选择“符号”(Symbols)。

### 说明

使用符号编辑器创建符号时，允许使用以下字符：A-Z、a-z、0-9、\_、-、^、!、#、\$、%、&、'、/、(、)、<、>、=、?、~、+、\*、'、`、:、|、@、[、]、{、}、"。使用 STEP 7 创建符号时，还应记住，如果符号文件中的符号同时具有 `<symbolname>` 和 `<symbolname>[<索引|<]` 形式，则解析数组时会出现问题。

## 优点/缺点

使用功能强大的 SR OPC 服务器的缺点为只能采用 SR 单协议模式。

但另一方面，则具有以下优点：

- 与多协议模式相比，性能更高。
- 组态简单。
- 多个客户端可同时使用服务器。
- OPC 服务器的稳定性不取决于客户端。

### 2.9.1.2 协议 ID

SEND/RECEIVE 协议的协议 ID 是“SR”。

### 2.9.1.3 连接名称

连接名称是在 STEP 7 中组态的名称，用于标识连接。

OPC 服务器支持以下连接类型：

- ISO 传输连接
- ISOonTCP 连接
- TCP 连接

## 连接名称示例

典型示例：

*SR\_Connection*

### 2.9.1.4 变量服务

变量服务允许直接访问和监视可编程控制器中的变量。

变量采用符号寻址，变量名称的表示法面向编程工具。对于直读访问，OPC 服务器将所需的地址信息传送到接收器并由接收器发回所需数据。在写访问期间，OPC 服务器将地址信息连同要写入的值一起传送。

#### 说明

所需变量必须与伙伴设备的组态一致。否则，访问特定区域时将导致通信错误。

注册变量时，OPC 服务器仅检查语法。

根据伙伴的组态，无法确定伙伴设备上的变量是否有效。

### SEND/RECEIVE 变量服务中过程变量的语法

```
SR: [<connectionname>]<area>{,}<type><address>{,<quantity>}
```

#### 说明

##### SR

用于访问过程变量的 SEND/RECEIVE 协议。

##### <connectionname>

协议特定的连接名称。连接名称在组态中指定。

##### <area>

要寻址的对象。

DB $nn$	数据块编号 nn
Q	输出
I	输入
M	位存储器
P	外围（适用于 S5 CPU）
PII	外围（适用于 S7 CPU）
RS	系统区域
IA	绝对起始地址
DX $nn$	已扩展的数据块

DE $n$	外部存储器中的数据块
QB	已扩展的 I/O

**<type>**

数据类型

数据类型（格式标识符）将转换成 OPC 服务器上对应的 OLE 数据类型。

格式标识符	说明	OLE 数据类型	Visual Basic 类型
X	位（布尔）。 指定位号（0 到 7）。  注意：仅适用于 I、Q、M、P、PII、DB 和 QB 对象（只读权限）	VT_BOOL	布尔
B 或 BYTE	字节（无符号 8 位）  注意：仅对 I、Q、M、P、PII 和 QB 对象可用	VT_UI1	字节
CHAR	字节（有符号 8 位）  注意：仅对 I、Q、M、P、PII 和 QB 对象可用	VT_I1	整型
W 或 WORD	字（无符号 16 位）	VT_UI2	长整型
INT	字（有符号 16 位）	VT_I2	整型
D 或 DWORD	双字（无符号 32 位）	VT_UI4	双精度
DINT	双字（有符号 32 位）	VT_I4	长整型
REAL	浮点， IEEE 表示	VT_R4	单精度
S5REAL	浮点， S5 表示	VT_R4	单精度

**<address>**

区域中变量的地址。

视区域而定，指定的地址必须是一个字节地址或字地址。

访问以下区域时，指定的地址将解释为字地址： DB $n$

字地址:

- $\text{DX}nn$
- $\text{DE}nn$

对于其它区域, 地址为字地址。对于数据类型 X, <address> 的语法为 <byte-offset.bit> 位的值范围为 0 到 7。

字节地址中字节偏移量的值范围为 0 到 65534, 字地址中字节偏移量的值范围为 0 到 32767。视设备和类型额定, 实际可用的地址值可能较少。

字地址和字节地址的寻址方式如下:

- 字地址: <wordnumber>{.bitnumber}
- 字节地址: <bytenumber>{.bitnumber}

#### <quantity>

要寻址的某一类型的变量数目, 从地址参数中指定的地址开始 (值范围为 0 到 65535)。

对数据类型 X, 数量的输入为:

- 写访问 (WRITE): 从位编号 0 开始仅取 8 的倍数。
- 读访问 (FETCH): 从任意偏移量开始的任意数量。

#### 示例:

*SR: [/ISO-WRITE-1]DB1,X10.0,16*, 写访问权限, 偏移量 10, 从位编号 0 开始, 数量 16 位

*SR: [/ISO-FETCH-1]DB1,X10.4,9*, 读访问权限, 偏移量 10, 从位编号 4 开始, 数量 9 位

## 定时器和计数器的语法

SR: [<connectionname>]<number>

- $Tnn$   
定时器
- $Cnn$   
计数器

### 2.9.1.5 面向缓冲区的服务

利用面向缓冲区的服务，可通过程序控制较大数据块的传送。这些服务也称为 SEND/RECEIVE 服务。通过 OPC 服务器进行的数据传输通过以下变量实施：

- 发送数据块的变量
- 接收数据块的变量

数据块的默认大小在组态中指定，发送变量时，可以对长度进行限制。  
可在数据块内进行交替访问。

#### 固定的变量名称

为每个连接指定下列固定变量名称：

- *RECEIVE*
- *SEND*

#### 面向缓冲区的服务的过程变量语法

可用选项如下：

```
SR: [<connectionname>] receive{,<type><address>{,<quantity>}}
```

```
SR: [<connectionname>] send{<n>} {,<type><address>{,<quantity>}}
```

从 2006 版 SIMATIC NET PC Software 开始，还可使用第二种选项接收数据。  
激活该项后，数据即使没有发生变化也将发送：

```
SR: [<connectionname>] receivedata{,<type><address>{,<quantity>}}
```

该项不符合 OPC 规范。

#### 说明

##### **SR**

用于访问过程变量的 SEND/RECEIVE 协议。

##### **<connectionname>**

协议特定的连接名称。连接名称在组态中指定。

**receive**

从伙伴接收到的最后一个数据缓冲区。

数据缓冲区的结构不固定。因此，缓冲区始终以字节数组的形式传送。

OLE 数据类型	Visual Basic 类型
VT_ARRAY VT_UI1	字节

**说明**

RECEIVE 变量与接收缓冲区相对应。因此，只能对变量进行读取操作。要对 DEVICE 进行读取访问，请在通信系统中明确留出一个接收缓冲区。

如果在指定时间内此缓冲区没有接收到任何数据字段，则会报告超时。

因此，您应该仅监视这些变量或从缓存中读取它们。

**receivedata**

“receivedata”项不符合 OPC。

从功能角度来说，“receivedata”与上面所述的“receive”项对应。

不过，有下面的一点不同：

激活该类型的项后，即使内容没有发生改变，数据也将会发送。

这样，客户端就可以从伙伴接收未更改的数据缓冲区。OnDataChange

回调也没有协商的更新速率快。对于此功能，您应该始终设置比 SEND/RECEIVE 数据的发送速率更快的更新速率。

**send**

可传送至连接伙伴的发送缓冲区。

在组态过程中指定发送缓冲区大小的默认值。缓冲区始终以字节数组的形式传送。

对这些变量进行写访问会将发送缓冲区传送至伙伴。

OLE 数据类型	Visual Basic 类型
VT_ARRAY VT_UI1	字节

**说明**

禁止读取和激活此变量和源于此变量的变量。

对这些变量进行读访问很可能会造成伙伴终止连接。

## 说明

确认写入发送变量时，结果“S\_OK”仅表示发送缓冲区已成功传送到通信系统，随时可由接收伙伴提取。并不代表伙伴应用程序是否实际接收和处理缓冲区。

### <n>

发送缓冲区的大小（以字节为单位）。

如果连接使用了不同大小的发送缓冲区，则可使用 *n*。如果忽略 *n*，则会使用在组态中输入的缓冲区大小。

如果在本地 TCP/IP 中禁用了微协议，（请参见 SIMATIC STEP 7 或 SIMATIC NCM PC 组态工具），那么将无法指定发送缓冲区的大小。

在这种情况下，将使用在已定义组态中输入的缓冲区大小。

发送缓冲区的大小取决于组态。请注意，与 <type> 大小（以变量的字节数表示）相乘的 <quantity> 不能大于发送缓冲区 <n>。

### <type>

数据类型

数据类型（格式标识符）将转换成 OPC 服务器上对应的 OLE 数据类型。

格式标识符	说明	OLE 数据类型	Visual Basic 类型
X	位（布尔）	VT_BOOL	布尔
B 或 BYTE	字节（无符号 8 位）	VT_UI1	字节
CHAR	字节（有符号 8 位）	VT_I1	整型
W 或 WORD	字（无符号 16 位）	VT_UI2	长整型
INT	字（有符号 16 位）	VT_I2	整型
D 或 DWORD	双字（无符号 32 位）	VT_UI4	双精度
DINT	双字（有符号 32 位）	VT_I4	长整型
REAL	浮点， IEEE 表示	VT_R4	单精度
S5REAL	浮点， S5 表示	VT_R4	单精度

### <address>

区域位编号中变量的字节地址。

对于位数据类型，请求位通过字节编号.位编号进行寻址。

**<quantity>**

要寻址的某一类型变量的数目，起始地址为 *address* 参数中指定的地址。

**说明**

请注意下列各项：

- 通过指定可选信息类型、地址和数量，可以定义对数据块子区域的结构化访问。
- 不同长度或不同连接名称的发送数据或接收数据的变量具有独立的存储区。
- 为独立存储区创建数据项时，将分配发送数据缓冲区并将其初始化为零。  
对发送数据项执行的写入操作将写入内部写入缓冲区，然后进行传送。
- 确认写入发送变量时，结果“S\_OK”仅表示发送缓冲区已成功传送到通信系统，随时可由接收伙伴提取。并不代表伙伴应用程序是否实际接收和处理缓冲区。
- 数据块进行非周期性传送。始终传送完整的发送数据块。

此规则也适用于子元素访问或几个客户端同时写入此数据项的情况。

**面向缓冲区的服务的过程变量示例**

下面几个示例说明了面向缓冲区的服务的变量名称的语法。

**接收变量**

**SR:[MyConnection]receive,w4,6**

receive,w4,6

接收缓冲区中从偏移量 4 开始的 6 个数据字。

**SR:[MyConnection]receive,dword7**

receive,dword7

接收缓冲区中从偏移量 7 开始的一个双字

**SR:[MyConnection]receive,REAL0,2**

receive,REAL0,2

接收缓冲区中从偏移量 0 开始，带有 2 个浮点值的数组。

传送不必更改的数据的示例（请参见“receivedata”）：

**SR:[MyConnection]receivedata,w4,6**

receivedata,w4,6

接收缓冲区中从偏移量 4 开始的 6 个数据字。

## 发送变量

**SR:[MyConnection]send30,dword7**

send30,dword7

30 字节长的发送缓冲区中从字节 7 开始的一个双字。如果发送缓冲区的默认大小不是 30，则变量会访问一个独立的缓冲区。

**SR:[MyConnection]send,B20,6**

send,B20,6

默认大小的发送缓冲区中从偏移量 6 开始的一个 20 字节的数组。

发送缓冲区的默认大小在组态中指定。

**SR:[MyConnection]send8,DINT0**

send8,DINT0

大小为 8 的发送缓冲区中从地址 0 开始的一个有符号双字。

### 2.9.1.6 SEND/RECEIVE 特定的信息变量

使用 SEND/RECEIVE 特定的信息变量，可以查询有关连接状态的信息。

可获取以下信息：

连接状态

## 开放式通信服务 (SEND/RECEIVE) 的信息变量的语法

SR: [<connectionname>] &<informationparameter>()

## 说明

**SR**

用于访问过程变量的 SEND/RECEIVE 协议。

**<connectionname>**

协议特定的连接名称。连接名称在组态中指定。

**<informationparameter>**

可用选项如下：

- statepath
- 到伙伴设备的通信连接的状态。

结果以字符串形式表示。

返回值:

*DOWN*

还未建立连接

*UP*

已建立连接

*RECOVERY*

正在建立连接

*ESTABLISH*

为将来的扩展保留

OLE 数据类型	Visual Basic 类型
VT_BSTR	字符串

#### **statepathval**

到伙伴设备的通信连接的状态。

结果以数字形式表示。

返回值:

*1*

还未建立连接

*2*

已建立连接

*3*

正在建立连接

*4*

为将来的扩展保留

OLE 数据类型	Visual Basic 类型
VT_UI1	字节

#### **SEND/RECEIVE 特定的信息变量示例**

在此，您将找到多个有关开放式通信 (SEND/RECEIVE) 信息变量返回值的示例:

*SR:[SR\_CONNECTION]&statepath()*

**&statepath()**

例如，可返回以下值：

*UP*

已建立连接

*SR:[SR\_CONNECTION]&statepathval()*

**&statepathval()**

例如，可返回以下值：

*2*

已建立连接。

### 2.9.1.7 系统特定信息变量的语法

*SR:[SYSTEM]&version()*

**&version()**

返回 SR OPC 服务器的版本 ID。例如，此处的返回字符串为  
SIMATIC NET Core Server SR V 7.xxxx.yyyy.zzzz Copyright 2012

数据类型： VT\_BSTR

访问权限： 只读

*S7:[SYSTEM]&sapiversion()*

**&winsockversion()**

Winsocket 接口的版本 ID。

数据类型： VT\_BSTR

访问权限： 只读

### 2.9.2 通过 PROFIBUS 的开放式通信服务 (SEND/RECEIVE)

#### ISO/OSI 参考模型

通过 PROFIBUS 的开放式通信服务 (SEND/RECEIVE)

是一种面向框架的简单协议，适用于 SIMATIC 设备。在 PROFIBUS 中使用 ISO/OSI 参考模型第 2 层中现场总线数据链路 (FDL) 的服务。

## 通过 PROFIBUS 的开放式通信服务 (SEND/RECEIVE) 的属性

SIMATIC NET 的 OPC 服务器具有以下特性：

- 使用可编程控制器上的 SIMATIC S5 和 SIMATIC S7 处理块通信
- 可在两个 PC 站之间建立链接
- 数据包传送速度更快
- 在数据包内进行结构化访问
- 显示和监视连接状态

### 2.9.2.1 协议 ID

FDL 协议的协议 ID 是“FDL”。

### 2.9.2.2 连接名称

连接名称是在 STEP 7 中组态的名称，用于标识连接。为 OPC 服务器选择一个唯一的 FDL 连接名称。

#### 连接名称示例

典型的名称为：

*FDL\_Connection*

### 2.9.2.3 面向缓冲区的服务

面向缓冲区的服务允许通过程序控制数据块的传送。

接收到的数据缓冲区和要发送的数据缓冲区将映射到 OPC 变量。

#### 固定的变量名称

为每个连接指定下列固定变量名称：

- *RECEIVE*
- *SEND*
- *SendSDA*
- *SendSDN*

## 面向缓冲区的服务 (FDL) 的过程变量语法

有两个选项：

```
FDL: [<connectionname>] receive{,<type><address>{,<quantity>}}
```

```
FDL: [<connectionname>] send{<n>} {,<type><address>{,<quantity>}}
```

### 说明：

#### **FDL**

用于访问过程变量的 FDL 协议。

#### **<连接名称>**

协议特定的连接名称。连接名称在组态中指定。

#### **receive**

从伙伴接收到的最后一个数据缓冲区。

数据缓冲区的结构不固定。因此，缓冲区始终以字节数组的形式传送。

### 说明

RECEIVE 变量与接收缓冲区相对应。因此，只能对变量进行读取操作。要对 DEVICE 进行读取访问，请在通信系统中明确留出一个接收缓冲区。

如果在指定时间内此缓冲区没有接收到任何数据字段，则会报告超时。

因此，您应该仅监视这些变量或从缓存中读取它们。

OLE 数据类型	Visual Basic 类型
VT_ARRAY VT_UI1	字节类型元素数组

#### **send**

可以传送到连接伙伴的发送缓冲区，始终显示。

在组态过程中指定发送缓冲区大小的默认值。缓冲区始终以字节数组的形式传送。

对这些变量及派生变量进行写访问会将发送缓冲区传送至伙伴。

禁止读取和激活此变量和源于此变量的变量。

根据连接组态中站地址和指定 SAP 的组合，写入发送数据项时需要使用特殊的 FDL 服务：

本地 SAP	远程站	远程 SAP	含义/用于发送的服务
0...62, 255	0...126	0...62, 255	发送和接收 / SDA
0...62, 255	0...126	63	仅发送 / SDA
63	0...126	0...62, 255	仅接收（无发送）
0...62, 255	127	63	广播：仅向所有伙伴发送 / SDN

本地 SAP	远程站	远程 SAP	含义/用于发送的服务
63	127	0...62, 255	广播接收 (无发送)
0..62, 255	127	0...62, 255	组播：仅发送至所有已激活远程 SAP/SDN

OLE 数据类型	Visual Basic 类型
VT_ARRAY VT_UI1	字节类型元素数组

## 说明

确认写入发送变量时，结果“S\_OK”仅表示发送缓冲区已成功传送到通信系统，随时可由接收伙伴提取。并不代表伙伴应用程序是否实际接收和处理缓冲区。

## <n>

发送缓冲区的大小。

借助面向缓冲区的服务，您可以发送和接收数据缓冲区。

接收到的数据缓冲区和要发送的数据缓冲区将映射到 OPC 变量。

例如，适当大的数组可包含整个接收缓冲区。

也可以将缓冲区的子部分分配给各个变量。

数据块进行非周期性传送。发送作业将写入使用“零”初始化的内部写缓冲区。  
始终传送写缓冲区中的完整数据块。  
此规则也适用于交替访问或几个客户端同时写入此数据项的情况。

#### <类型>

数据类型

数据类型（格式标识符）将转换成 OPC 服务器上对应的 OLE 数据类型。

格式标识符	说明	OLE 数据类型	Visual Basic 类型
X	位（布尔）	VT_BOOL	布尔
B 或 BYTE	字节（无符号 8 位）	VT_UI1	字节
CHAR	字节（有符号 8 位）	VT_I1	整型
W 或 WORD	字（无符号 16 位）	VT_UI2	长整型
INT	字（有符号 16 位）	VT_I2	整型
D 或 DWORD	双字（无符号 32 位）	VT_UI4	双精度
DINT	双字（有符号 32 位）	VT_I4	长整型
REAL	浮点， IEEE 表示	VT_R4	单精度
S5REAL	浮点， S5 表示	VT_R4	单精度

#### <地址>

区域位编号中变量的字节地址。值的范围取决于组态。

对于位数据类型，请求位通过字节编号.位编号进行寻址。

<bytetenumber>{.<bitnumber>}

如果 <bitnumber> 中没有输入内容，位将通过 0 进行寻址。

#### <数量>

要寻址的某一类型变量的数目，起始地址为 *address* 参数中指定的地址。

值范围取决于组态。

## 使用 SDA 和 SDN (FDL) 的面向缓冲区的服务的过程变量语法

如果使用的服务不取决于伙伴站地址和 SAP 的组合，这将成为一项优势。

通过使用名称“SendSDA”和“SendSDN”，您可以指定仅将相关服务“SDA”或“SDN”用于发送。这些特殊名称不会显示在 OPC 浏览器中。

## 语法:

有两个选项:

```
FDL: [<connectionname>] SendSDA{<n>} {,<type><address>{,<quantity>}}
```

```
FDL: [<connectionname>] SendSDN{<n>} {,<type><address>{,<quantity>}}
```

## 说明

### FDL

用于访问过程变量的 FDL 协议。

#### <连接名称>

协议特定的连接名称。连接名称在组态中指定。

SendSDA

SendSDN

仅有服务 SDA 或 SDN 用于发送。

#### <n>

发送缓冲区的大小。

借助面向缓冲区的服务，您可以发送和接收数据缓冲区。

接收到的数据缓冲区和要发送的数据缓冲区将映射到 OPC 变量。

例如，适当大的数组可包含整个接收缓冲区。

也可以将缓冲区的子部分分配给各个变量。

#### <类型>

数据类型

数据类型（格式标识符）将转换成 OPC 服务器上对应的 OLE 数据类型。

格式标识符	说明	OLE 数据类型	Visual Basic 类型
X	位（布尔）	VT_BOOL	布尔
B 或 BYTE	字节（无符号 8 位）	VT_UI1	字节
CHAR	字节（有符号 8 位）	VT_I1	整型
W 或 WORD	字（无符号 16 位）	VT_UI2	长整型
INT	字（有符号 16 位）	VT_I2	整型
D 或 DWORD	双字（无符号 32 位）	VT_UI4	双精度
DINT	双字（有符号 32 位）	VT_I4	长整型

格式标识符	说明	OLE 数据类型	Visual Basic 类型
REAL	浮点, IEEE 表示	VT_R4	单精度
S5REAL	浮点, S5 表示	VT_R4	单精度

### <地址>

区域位编号中变量的字节地址。值的范围取决于组态。

对于位数据类型, 请求位通过字节编号.位编号进行寻址。

### <数量>

要寻址的某一类型变量的数目, 起始地址为 *address* 参数中指定的地址。

值范围取决于组态。

### 说明

请注意下列各项:

- 通过指定可选信息类型、地址和数量, 可以定义对数据块子区域的结构化访问。
- 不同长度或不同连接名称的发送数据或接收数据的变量具有独立的存储区。
- 为独立存储区创建数据项时, 将分配发送数据缓冲区并将其初始化为零。  
对发送数据项执行的写入操作将写入内部写入缓冲区, 然后进行传送。
- 数据块进行非周期性传送。可以对相同的数据同时执行多个网络作业。  
始终传送完整的发送数据块。

此规则也适用于子元素访问或几个客户端同时写入此数据项的情况。

几个客户端同时对同一发送数据块或发送数据块的子区域执行写操作会导致不一致。

因此, 我们建议:

- 始终读取或写入完整的数据块。
- 将“并行网络作业的最大数目”(Maximum number of parallel network jobs)  
设置为“一”(one), 尽管这样会降低传输性能。

## 面向缓冲区的服务 (FDL) 的过程变量示例

在此, 您将找到说明 FDL 变量的变量名称的语法示例。

### 读变量

*FDL:[MyConnection]receive,w0,6*

接收缓冲区中从字节 0 开始的 6 个数据字。

*FDL:[MyConnection]receive,dword7*

从字节 7 开始的一个双字。

*FDL:[MyConnection]Receive,REAL0,2*

接收缓冲区中从字节 0 开始的两个实数值

## 写变量

*FDL:[MyConnection]send30,dword7*

30 字节长的发送缓冲区中从字节 7 开始的一个双字。

*FDL:[MyConnection]SendSDN,B5,20*

默认大小的发送缓冲区中从偏移量 5 开始的一个 20 字节数组。

默认大小在组态过程中设置。不管在组态中指定哪个 SAP，都将使用 PROFIBUS FDL SDN 服务。

*FDL:[MyConnection]Send8,DINT0*

大小为 8 字节的发送缓冲区中从地址 0 开始的一个有符号双字。

### 2.9.2.4 FDL 特定的信息变量

用于通过 PROFIBUS (FDL) 的开放式通信服务 (SEND/RECEIVE) 的 OPC 服务器可提供变量，其中的信息能够用于查询通信网络的状态和范围。

## FDL 特定信息变量的语法

*FDL: [ || <CPname> ] &<informationparameter>()*

## 说明

### FDL

用于访问过程变量的 FDL 协议。

### <CPname>

CP 名称在组态过程中指定。仅支持组态了 FDL 连接的 CP 名称。

### <informationparameter>

五个选项如下所示：

- busparameter
- defaultsap

- identify
- ts
- lifelist

下文将介绍这五个选项。

### **busparameter**

获取在指定连接上运行的 PROFIBUS 网络的总线参数。值以字节数组形式返回，与 FDL 服务 FDL\_READ\_VALUE 返回的值对应。有关详细信息，请参见有关 FDL 编程接口的手册。

内容	说明	数据类型	长度（字节）
HSA	总线上最高的 PROFIBUS 地址, 2...126	BYTE	1
TS	本地站的 PROFIBUS 地址, 0...hsa 或 126。	BYTE	1
Station_Type	本地站的类型	整型	2
传输速率	传输速率	整型	2
Medium_red	冗余	整型	2
Retry_Ctr	对无响应站（远程）的重复调用次数, 0...7。	UWORD	2
Default_SAP	站（本地）的默认 SAP 数, 0...63	BYTE	1
network _con_SAP	保留	BYTE	1
TSL	SLOT 时间	UWORD	2
TQUI	调制器淡出时间/中继器切换时间	UWORD	2
TSET	设置时间	UWORD	2
MIN_TSDR	最小站延迟响应器	UWORD	2
MAX TSDR	最大站延迟响应器	UWORD	2
TTR	目标循环时间	DWORD	4
GAP	GAP 更新因数	BYTE	1
in_Ring_ desired	请求输入环	BOOLEAN	1

内容	说明	数据类型	长度 (字节)
physical_layer	可选择物理总线特性	整型	2
ident	供应商名称、控制器类型、硬件和软件版本	字符串	211

OLE 数据类型	Visual Basic 类型
VT_ARRAY   VT_UI1	Byte()

**defaultsap**

返回默认 SAP 的值 (SAP = Service Access Point, 服务访问点)。未明确指定 SAP 时，将使用默认 SAP 寻址。

OLE 数据类型	Visual Basic 类型
VT_UI1	字节

**identify**

以包含 4 个字符串的数组形式获取指定连接的站标识符：

返回值的元素：

供应商

控制器

硬件版本

软件版本

OLE 数据类型	Visual Basic 类型
VT_ARRAY   VT_BSTR	String()

**ts**

返回指定模块的本地站地址。

OLE 数据类型	Visual Basic 类型
VT_UI1	字节

**lifelist**

总线上可访问节点的相关信息。

该数组拥有 127 个元素，包含了每一个可能站地址的信息。

每个站地址都会分配一个数组索引。

数组条目的返回值：

*FDL\_STATION\_NON\_EXISTENT*

不存在节点（值 0x10）

*FDL\_STATION\_PASSIVE*

被动节点（值 0x00）

*FDL\_STATION\_READY\_FOR\_RING*

准备好在 PROFIBUS 令牌环中输入的节点（值 0x30）

*FDL\_STATION\_ACTIVE*

主动节点（值 0x20）

OLE 数据类型	Visual Basic 类型
VT_ARRAY of VT_UI1	含 127 个字节类型元素的数组()

**说明**

此信息使用 FDL 服务 *FDL\_LIFE\_LIST\_CREATE\_REMOTE* 获得。

该服务会使总线的负载明显增加，因此，应尽可能不主动监视该变量。

**2.9.2.5 系统特定信息变量的语法**

FDL:[SYSTEM]&version()

**&version()**

返回 FDL OPC 服务器的版本 ID。例如，此处返回字符串

*SIMATIC NET Core Server FDL V 7.xxxx.yyyy.zzzz Copyright 2012*

数据类型：VT\_BSTR

访问权限：只读

## 2.10 通过工业以太网实现的 OPC UA 的开放式通信服务 (SEND/RECEIVE)

### 2.10.1 与 OPC UA 进行的 SR 通信的属性

SIMATIC NET OPC 服务器允许使用通过 OPC UA 进行的 SR 通信。

通过工业以太网提供开放式通信服务 (SEND/RECEIVE) 的 SIMATIC NET OPC UA 服务器已发布用于与 S7 设备进行通信。它也允许用户与第三方设备通信。

SIMATIC NET 的 SR OPC UA 服务器具有以下特征：

- 使用 SIMATIC S5 处理块和 S7 函数块进行通信
- 允许使用发送和接收功能链接两个 PC 站。
- 支持用于访问伙伴设备对象的 WRITE 和 FETCH 函数。
- 通过 SEND 和 RECEIVE 模式快速访问大型数据包。
- 在数据包内进行交替访问。
- 显示和监视连接状态

### 2.10.2 SR 协议的 SIMATIC NET OPC UA 服务器

#### 简介

以下部分介绍了同样支持 OPC UA 的 SR 协议的组态变型。为此，将 SR COM OPC 数据访问服务器设置为进程外 OPC 服务器。

## 组态

在“通信设置”(Communication Settings) 组态程序的“OPC 协议选择”(OPC protocol selection) 目录中选择“SR”和“OPC UA”可激活 SR OPC UA 服务器：



图 2-56 “通信设置”组态程序中用于为 SR 协议选择 OPC UA 的窗口

## 优点/缺点

在使用 SR OPC UA 服务器时，只有 SR 的进程外模式可用。必须运行 SR OPC UA 服务器进程以维持准备接收的状态。即使所有 OPC UA 客户端都已注销，也不会退出 SR OPC UA 服务器。不能同时运行高速进程内的 OPC DA SR 服务器。

但另一方面，则具有以下优点：

- 不再需要 COM/DCOM 组态。
- 高速、安全的通信
- 事件和数据访问仅需要一个服务器。

### 2.10.3 如何寻址 SR OPC UA 服务器?

#### 服务器 URL

对于 TCP 协议, OPC 客户端有两种对服务器进行寻址的方法:

- 直接寻址:
  - opc.tcp://<hostname>:55102
  - 或
  - opc.tcp://<IP-Adresse>:55102
  - 或
  - opc.tcp://localhost:55102

SR OPC UA 服务器使用端口 55102。

- 也可以使用 OPC UA 发现服务查找 SR OPC UA 服务器的 URL。

要查找发现服务器, 请输入以下内容:

- opc.tcp://<hostname>:4840
- 或
- opc.tcp://<IP-Adresse>:4840
- 或
- http://<hostname>:52601/UADiscovery/
- 或
- http://<IP-Adresse>:52601/UADiscovery/

发现服务器使用端口 4840 (用于 TCP 连接) 和端口 52601 (用于 HTTP 连接)。

#### IPv6 地址

IPv6 地址也可以用作 IP 地址。地址必须位于括号中, 例如  
[fe80:e499:b710:5975:73d8:14]

#### 端点和安全模式

SIMATIC NET SR OPC UA 服务器支持采用本地二进制 TCP 协议的端点, 并允许使用证书进行验证以及进行加密传送。

已寻址主机上的“发现”服务将用信号通知服务器的端点，换句话说就是服务器的安全要求和协议支持。

SR OPC UA 服务器上的服务器 URL "opc.tcp://<hostname>:55102" 提供了以下端点：

- 在“签名和加密”安全模式下的端点：

必须进行签名和加密才能与服务器进行通信。通过交换证书和输入密码来保护通信。

除了安全模式，还显示了安全策略 Basic128Rsa15。

- 在“无”安全模式下的端点：

在此模式下，服务器不需要使用安全功能（安全策略“无”）。

有关安全功能的详细信息，请参见“对 OPC UA 接口进行编程 (页 586)”部分。

安全策略“Basic128Rsa18”和“无”在 OPC 基金会的 UA 规范中，其 Internet 地址如下所示：

[> "Specifications" > "Part 7"](http://opcfoundation.org/UA)

有关更多的详细信息，请参见以下 Internet 页面：

OPC 基金会 ([> "Security Category" > "Facets" > "Security Policy"](http://www.opcfoundation.org/profilereporting/index.htm))

## OPC Scout V10 的 OPC UA 发现

在 OPC Scout V10 中，您可以打开“OPC UA 发现”(OPC UA Discovery) 对话框并在 OPC Scout V10 的导航区域中输入 UA 端点。

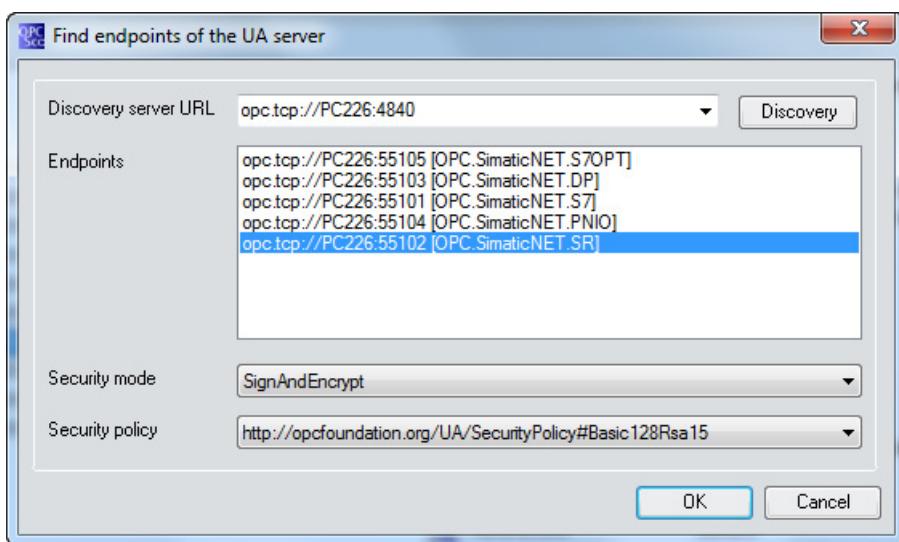


图 2-57 OPC Scout V10 的“查找 UA 服务器端点”对话框

## 2.10 通过工业以太网实现的 OPC UA 的开放式通信服务 (SEND/RECEIVE)

可使用 OPC UA 发现服务查找 SR OPC UA 服务器。相关条目，请参见上述“服务器 URL”。

OPC Scout V10 包含了一系列 OPC UA 端点。

然后，被寻址主机上的“发现”服务会用信号通知已注册的 OPC UA 服务器、其端口和安全模式。

更多详细信息，请参见 OPC Scout V10 的在线帮助。

### 2.10.4 协议 ID

OPC UA 中的 OPC UA SEND/RECEIVE 协议的协议 ID 为“SR”。

### 2.10.5 SR OPC UA 服务器提供了哪些命名空间？

**SR OPC UA 服务器提供了以下命名空间：**

命名空间索引	“标识符”（命名空间 URI）/注释
0	“ <a href="http://opcfoundation.org/UA/">http://opcfoundation.org/UA/</a> ” (由 OPC 基金会指定)
1	“urn:Siemens.Automation.SimaticNET.SR:(GUID)”本地高速 SR OPC UA 服务器的唯一标识符。
2	“SRTYPES:” SR 特定的对象类型的定义。
3	“SR:” 本地高速 SR OPC UA 服务器的标识符，使用新的简化语法（可浏览并可与 UA 一起使用）
4	“SRCOM:” 服务器的标识符，使用旧的语法，与 SR OPC DA 兼容（可与 UA 一起使用，但不能浏览）

命名空间索引 0 和 1 保留，其意义由 OPC 基金会指定。

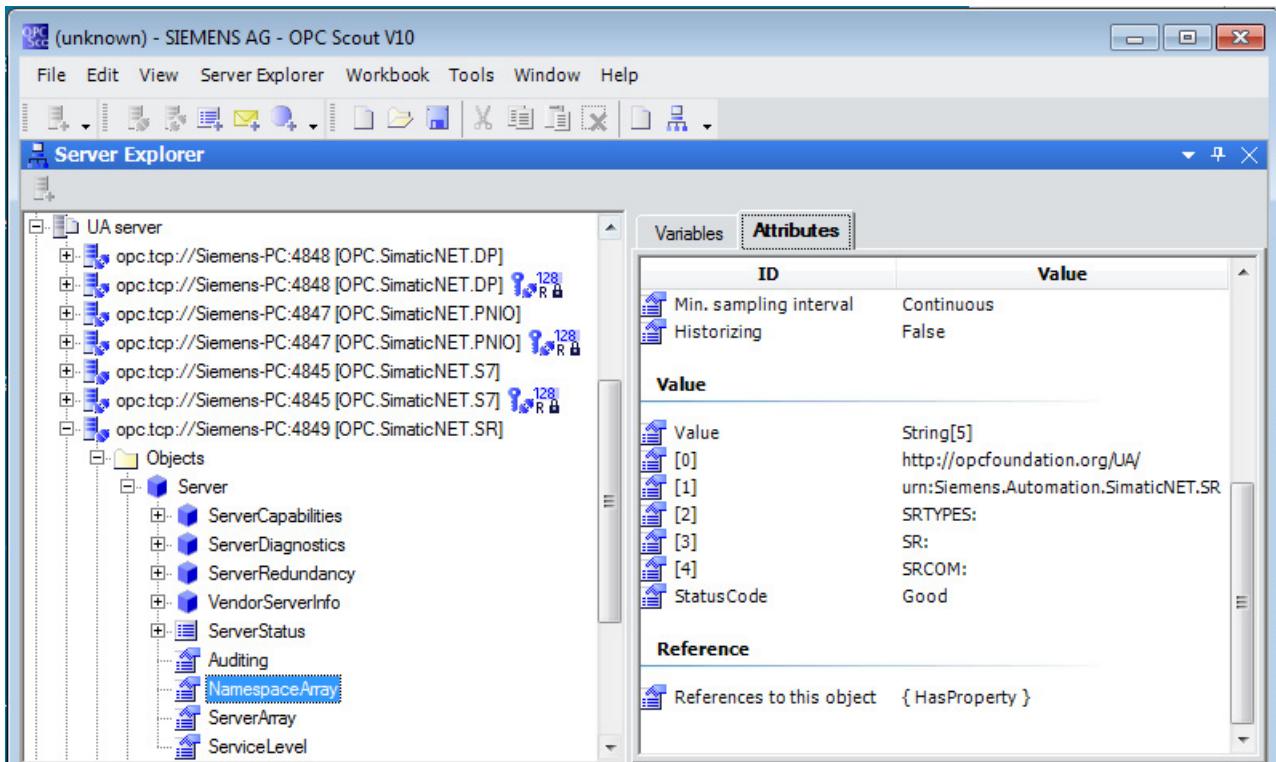


图 2-58 使用 OPC Scout V10 的浏览功能显示 SR OPC UA 命名空间

## 2.10.6 连接名称

连接名称是在 STEP 7 中组态的名称，用于标识连接。

OPC UA 服务器支持以下连接类型：

- ISO 传输连接
- ISOOnTCP 连接
- TCP 连接

### 连接类型

可通过 SR 连接的 SR 访问选项类型将在 STEP 7 中设置。连接可以是：

- 仅 Fetch,
- 仅 Write 或
- 仅 Send/Receive

## 连接名称示例

典型示例包括：

- ISOonTCP conn-1
- ISO-on-TCP connection1

## 2.10.7 Nodeld

### SR 过程变量的标识

Nodeld 借助下列元组标识 SR 过程变量：

- 命名空间索引
- 标识符（字符串、数字值）

### 示例

- Nodeld:

- 命名空间 URI:

*SR:*

(= 命名空间索引 3) 针对 Siemens.Automation.SimaticNET.SR

- 标识符:

*SRconnectionname.send,100.0,b,100*

- Nodeld:

- 命名空间 URI:

*SRCOM:*

(= 命名空间索引 4) 针对 OPC.SimaticNET; 语法兼容 SR OPC DA

- 标识符:

*SR:[SRconnectionname]send,b0,100*

## 新的命名空间是如何适应 OPC UA 运作的？

OPC UA 视图与自动化对象相关，也与对象的各种属性相关。OPC UA 不再单独访问各项，而是还会访问对象及其子对象。

- 例如，数据变量是 SR 连接对象的子对象。特性和属性可更详细地定义对象。
- 访问数据块的“OPC 数据访问”数据项最接近 OPC UA 数据变量。

合格的 NodId 标识符在 OPC UA 中比在“OPC 数据访问”中更有意义。每次单独访问对象、子对象、特性及属性均使用其 NodId。

OPC UA 提供显示名称等内容以支持本地语言。

也就是说，可通过不同方式浏览相同对象（例如在 OPC UA 客户端指定的各种语言环境下），每次都显示同一个 NodId。显示名称的选择与相关 NodId 类似。整个命名空间的文本均采用英语。

## SR OPC UA 数据对象的语法

通常情况下，SR UA 服务器上的 OPC UA 对象的 NodId 具有以下结构：

*<connectionobject>*."*<subobject>*".*<property>*

子对象可包含其它子对象。

无法解析的 NodId 会被拒绝，并会显示错误消息。任何项中的字母“A-Z”都不区分大小写。

## 符号对象表示法

OPC UA 规范建议对地址空间的层级描述使用统一的符号表示法。

本文档中使用以下符号：

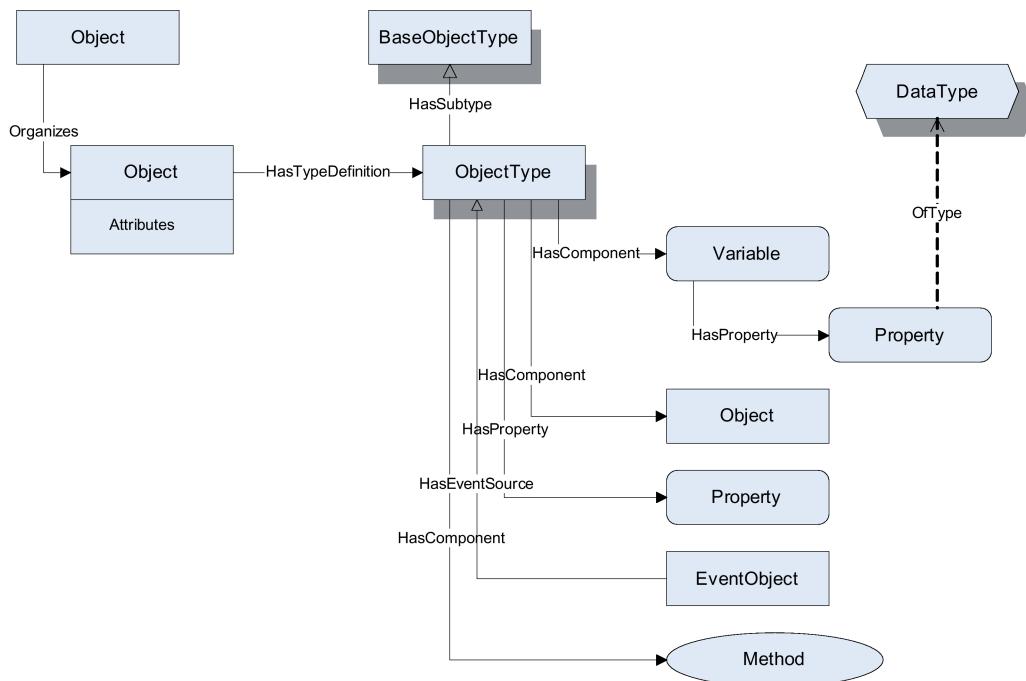


图 2-59 OPC UA 地址空间的符号

## 2.10.8 S5 数据块和区域的数据变量（与 S5 兼容的通信）

S5 数据块和区域的数据变量的读写（与 S5 兼容的通信）需要待组态的 Fetch 或 Write 连接。

Fetch 连接上的数据变量为只读变量。在 Write 连接上，只能对数据变量执行写入操作。如果需要读取和写入通信伙伴的数据块，则需要对两个连接进行组态，但这将完全由 OPC UA 服务器独立进行管理。

### S5 数据块和区域的数据变量的语法

用于读取和写入数据变量的 SR OPC UA 节点 ID 的过程变量的简化语法：

命名空间 URI: SR: (命名空间索引: 3)

## 语法

有两个选项：

```
<FETCHconnectionname>.<datavariable>.{<offset>{,<SRtype>{,<quantity>}}}
```

```
<WRITEconnectionname>.<datavariable>.{<offset>{,<SRtype>{,<quantity>}}}
```

## 说明

### **<FETCHconnectionname> 和 <WRITEconnectionname>**

协议特定的连接名称。连接名称在组态中指定。

### **<datavariable>**

与 S5 兼容的通信的数据变量。

数据变量		注释
DB<db>	无符号 8	数据块
DX<dx>	无符号 8	已扩展的数据块
DE<de>	无符号 8	外部存储器中的数据块
I		输入
Q		输出
P		外围设备 I/O 的输入和输出
QB		已扩展的 I/O
M		位存储器
C		计数器
T		定时器

- 数据变量“db”、“dx”和“de”的区域的相关信息

在“DB”、“DX”和“DE”标识符之后，也需要指定块编号，例如

DB10；协议特定的寻址却仅支持 0-255（8 位）的块编号。

在这些区域中，协议仅允许对整个字（16 位）进行寻址。

因此，这些区域的默认数据类型为“w”。

- 在 Fetch 连接上，所寻址的区域也可不与整个字相对应。

这意味着，例如，您可以对字节或位（包括数组）以及带有奇数字节偏移量的 16 位和 32 位数据类型进行寻址。然而，OPC UA

服务器却始终要求通信伙伴提供整个字并从这些字中提取相关数据。

- 在 Write 连接上，出于一致性的考虑，仅能对与整个字相对应的区域进行寻址。

这意味着，需要指定偶数和整数块偏移量以及仅与 16

位的倍数（整数）相对应的区域。使用 S7 通信伙伴，还可以指定奇数块偏移量。  
示例：

语法示例	FETCH	WRITE
SRconnectionname.db10.1,b	只读	不可用
SRconnectionname.db10.4, b,8	只读	只写
SRconnectionname.db10.3,x1	只读	不可用
SRconnectionname.db10.5,x4, 6	只读	不可用
SRconnectionname.db10.6,x0, 32	只读	只写
SRconnectionname.db10.7,w	只读	只写
SRconnectionname.db10.8,w	只读	只写

- 数据变量“m”、“i”、“q”、“p”和“qb”的区域的相关信息

在这些区域中，协议仅允许对整个字节（8位）进行寻址。

因此，对于大多数数据类型没有限制，例外情况是：位。

- 在 Fetch 连接上，所寻址的区域也可不与整个字节相对应。

这尤其意味着，可以对位进行寻址（包括数组）。然而，OPC UA 服务器却始终要求通信伙伴提供整个字节并从这些字节中提取相关数据。

- 在 Write

连接上，出于一致性的考虑，仅能对与整个字节相对应的区域进行寻址。这意味着，需要针对位指定偶数和整数块偏移量以及仅与 8 位的倍数（整数）对应的区域。

示例：

语法示例	FETCH	WRITE
SRconnectionname.m.1,b	只读	只写
SRconnectionname.m.4,b,7	只读	只写
SRconnectionname.m.3,x1	只读	不可用
SRconnectionname.m.5,x4,6	只读	不可用
SRconnectionname.m.6,x0,3 2	只读	只写
SRconnectionname.m.7,w	只读	只写
SRconnectionname.m.8,w	只读	只写

- 数据变量“c”的区域的相关信息

地址信息包括计数器编号。

- 数据变量“t”的区域的相关信息

地址信息包括定时器编号。

**<offset>**

要寻址元素在数据记录中的字节地址。在使用 OPC UA 的情况下，此偏移量始终为一个字节。

**<SRtype>**

数据类型。

数据类型将转换成 OPC UA 服务器上对应的 OPC UA 数据类型。

表格 2-7 数据类型说明

数据类型	OPC UA 数据类型	注释 S7 数据类型
x<bitaddress>	布尔	位 (布尔) 除区域内的字节偏移量外, 还必须指定相关字节中的 <bitaddress>。 值范围是 0 到 7
b	字节 字节字符串	字节 (无符号) 如果未指定 <SRtype>, 则用作默认值。 OPC UA 不识别“Byte[]”, 但对此使用标量数据类型“ByteString”。
w	UInt16	字 (无符号)
dw	UInt32	双字 (无符号)
c	SByte	字节 (有符号)
i	Int16	字 (有符号)
di	Int32	双字 (有符号)
r	Float	浮点 (4 字节)
s5r	Float	S5 编码实数
c	UInt16	只在计数器区域
t	UInt16	只在定时器区域

**<quantity>**

元素的数量。 变量的数据类型是具有指定格式元素的数组。

指定多个数组元素将导致形成相应类型的数组, 即使只对一个数组元素进行寻址也是如此。

## S5 数据块和区域的数据变量示例

- **FETCHconnectionname.db10.10,w**  
标识从字节地址 10 开始的数据块 10 的数据字。
- **WRITEconnectionname.q.3**  
标识一个从字节地址 3 开始的输出字节（默认为“字节”）。
- **FETCHconnectionname.i.0,x0**  
标识一个从字节地址 0 位 0 开始的输入位。
- **WRITEconnectionname.m.3,x4,16**  
标识一个从字节地址 3 位 4（只读）开始的储存器位数组
- **FETCH/WRITEREconnectionname.t.22**  
标识定时器 22（默认）

### 2.10.9 面向缓冲区的服务

利用面向缓冲区的服务，可通过程序控制较大数据块的传送。这些服务也称为 SEND/RECEIVE 服务。可通过以下变量执行使用 OPC UA 服务器进行的数据传送：

- 发送数据块的变量
- 接收数据块的变量

数据块的默认大小在组态中指定，发送变量时，可以对长度进行限制。  
可在数据块内进行交替访问。

#### 固定的变量名称

为每个连接指定下列固定变量名称：

- *receive*
- *send*

#### 面向缓冲区的服务的过程变量语法

用于读取和写入数据变量的 SR OPC UA 节点 ID 的过程变量的简化语法：

命名空间 URI: SR: (命名空间索引: 3)

## 语法

可用选项如下：

```
<SRconnectionname>.send{,<block>}{.<offset>{,<SRtype>{,<quantity>}}}
```

```
<SRconnectionname>.receive{.<offset>{,<SRtype>{,<quantity>}}}
```

## 说明

### **<SRconnectionname>**

协议特定的连接名称。连接名称在组态中指定。

#### **send**

可传送至连接伙伴的发送缓冲区。

在组态过程中指定发送缓冲区大小的默认值。

缓冲区将始终作为字节数组进行传送（OPC UA 数据类型“Bytestring”）。

对这些变量进行写访问会将发送缓冲区传送至伙伴。

---

## 说明

禁止读取和激活此变量和源于此变量的变量。

对这些变量进行读访问很可能会造成伙伴终止连接。

---

#### **receive**

从伙伴接收到的最后一个数据缓冲区。

数据缓冲区的结构不固定。因此，缓冲区将始终作为字节数组进行传送（OPC UA 数据类型“Bytestring”）。

---

## 说明

如果使用 OPC UA 监视服务（receive 作为 MonitoredItem）读取数据块的接收信息，则可在监视服务的组态中设置对接收数据块的响应。假设将监控服务中的 DataChangeTrigger 设置为 OpcUa\_DataChangeTrigger\_StatusValueTimestamp，则新接收到的具有相同数据的数据块时还可以向 OPC UA 客户端发送信号。这样，OPC UA 客户端就可以从伙伴接收未更改的数据缓冲区。DataChange 通知不及协商的更新速率快。对于此功能，您应该始终设置比 send/receive 数据的发送速率更快的更新速率。

---

## 说明

**RECEIVE** 变量与接收缓冲区相对应。因此，只能对变量进行读取操作。若对此 **nodeId** 进行读访问，将在通信系统中明确留出一个接收缓冲区。

如果在指定时间内此缓冲区没有接收到任何数据字段，则会报告超时。

因此，您应该仅监视这些变量（通过 UA 订阅进行的访问）。

### <block>

发送缓冲区的大小（以字节为单位）。

如果连接中使用了不同大小的发送缓冲区，则可使用 **<bl>**。如果忽略 **<bl>**，则使用在组态中输入的缓冲区大小。

如果在本地 TCP/IP 中禁用了微协议，（请参见 SIMATIC STEP 7 或 SIMATIC NCM PC 组态工具），那么将无法指定发送缓冲区的大小。

在这种情况下，将使用在已定义组态中输入的缓冲区大小。

发送缓冲区的大小取决于组态。请注意，与 **<SRtype>**

大小（以变量的字节数表示）相乘的 **<quantity>** 不能大于发送缓冲区 **<bl>**。

### <SRtype>

DP 数据类型转换为 OPC UA 服务器上相应的 OPC UA 数据类型。

下表列出了类型标识符以及可用来表示变量值的对应 OPC 数据类型。

数据类型	OPC UA 数据类型	注释
x<bitaddress>	布尔	位（布尔） 除区域内的字节偏移量外，还必须指定相关字节中的 <b>&lt;bitaddress&gt;</b> 。 值范围是 0 到 7
b	字节	字节（无符号） 如果未指定
	字节字符串	<DPtype>，则用作默认值。 OPC UA 不识别“Byte[]”，但对此使用标量数据类型“ByteString”。
char	字节	整型
w	UInt16	字（无符号）
int	整型	
dw	UInt32	双字（无符号）

数据类型	OPC UA 数据类型	注释
di	Int32	双字 (有符号)
r	浮点	浮点 (4 字节)
s5r	浮点, S5 表示	单精度

**<offset>**

要寻址元素在数据记录中的字节地址。

**<quantity>**

元素的数量。 变量的数据类型是具有指定格式元素的数组。

指定多个数组元素将导致形成相应类型的数组，即使只对一个数组元素进行寻址也是如此。  
。

**说明**

请注意下列各项：

- 通过指定可选信息类型、地址和数量，可以定义对数据块子区域的结构化访问。
- 不同长度或不同连接名称的发送数据或接收数据的变量具有独立的存储区。
- 为独立存储区创建数据项时，将分配发送数据缓冲区并将其初始化为零。  
对发送数据项执行的写入操作将写入内部写入缓冲区，然后进行传送。
- 数据块进行非周期性传送。始终传送完整的发送数据块。  
此规则也适用于子元素访问或几个客户端同时写入此数据项的情况。

**面向缓冲区的服务的过程变量示例**

下面几个示例说明了面向缓冲区的服务的变量名称的语法。

**接收变量**

- **SRconnectionname.receive.4,w,6**  
标识接收缓冲区中从偏移量 4 开始的 6 个数据字。
- **SRconnectionname.receive.7,dw**  
标识接收缓冲区中从偏移量 7 开始的双字
- **SRconnectionname.receive.0,r,2**  
标识接收缓冲区中从偏移量 0 开始的带 2 个浮点值的数组。

## 发送变量

- **SRconnectionname.send,30.7,dw**  
标识 30 字节长度的发送缓冲区中从字节 7 开始的一个双字。  
如果发送缓冲区的默认大小不是 30，则变量会访问一个独立的缓冲区。
- **SRconnectionname.send,256.6,b,20**  
标识标准大小的发送缓冲区中从偏移量 6 开始的由 20 个字节组成的数组。  
发送缓冲区的默认大小在组态中指定。
- **SRconnectionname.send,8.0,di**  
标识大小为 8 的发送缓冲区中从地址 0 开始的带符号双字。

### 2.10.10 SR 特定的信息变量

使用 SEND/RECEIVE 特定的信息变量，可查询有关连接状态的信息。

#### 开放式通信服务 (SEND/RECEIVE) 的信息变量的语法

用于读取和写入 SR 特定的信息变量的 SR OPC UA Node ID 的过程变量的简化语法：

命名空间 URI: SR: (命名空间索引: 3)

#### 典型语法

```
<SRconnectionname>.statepath.statepath
```

## 说明

### <SRconnectionname>

协议特定的连接名称。连接名称在组态中指定。

### statepath.statepath

<b>statepath</b>	与伙伴设备通信连接的状态 此变量的值以数字形式读出，并能通过额外读取相关 Enumstring {UNKNOWN, DOWN, UP, RECOVERY, ESTABLISH} 将其分配到文本。 UA 类型“MultistateDiscreteType”的变量，只读		
	1	DOWN	未建立连接
	2	UP	已建立连接
	3	RECOVERY	未建立连接。正在尝试建立连接。
	4	ESTABLISH	为将来的扩展保留
	0	UNKNOWN	为将来的扩展保留

## 2.10.11 SR OPC UA 模板数据变量

### 通过 OPC UA SR

协议的过程变量，您可拥有灵活的设置选项以所需的数据格式获取设备的过程数据。

不过，在完全可浏览的命名空间中，不能将各种寻址选项放在一起。

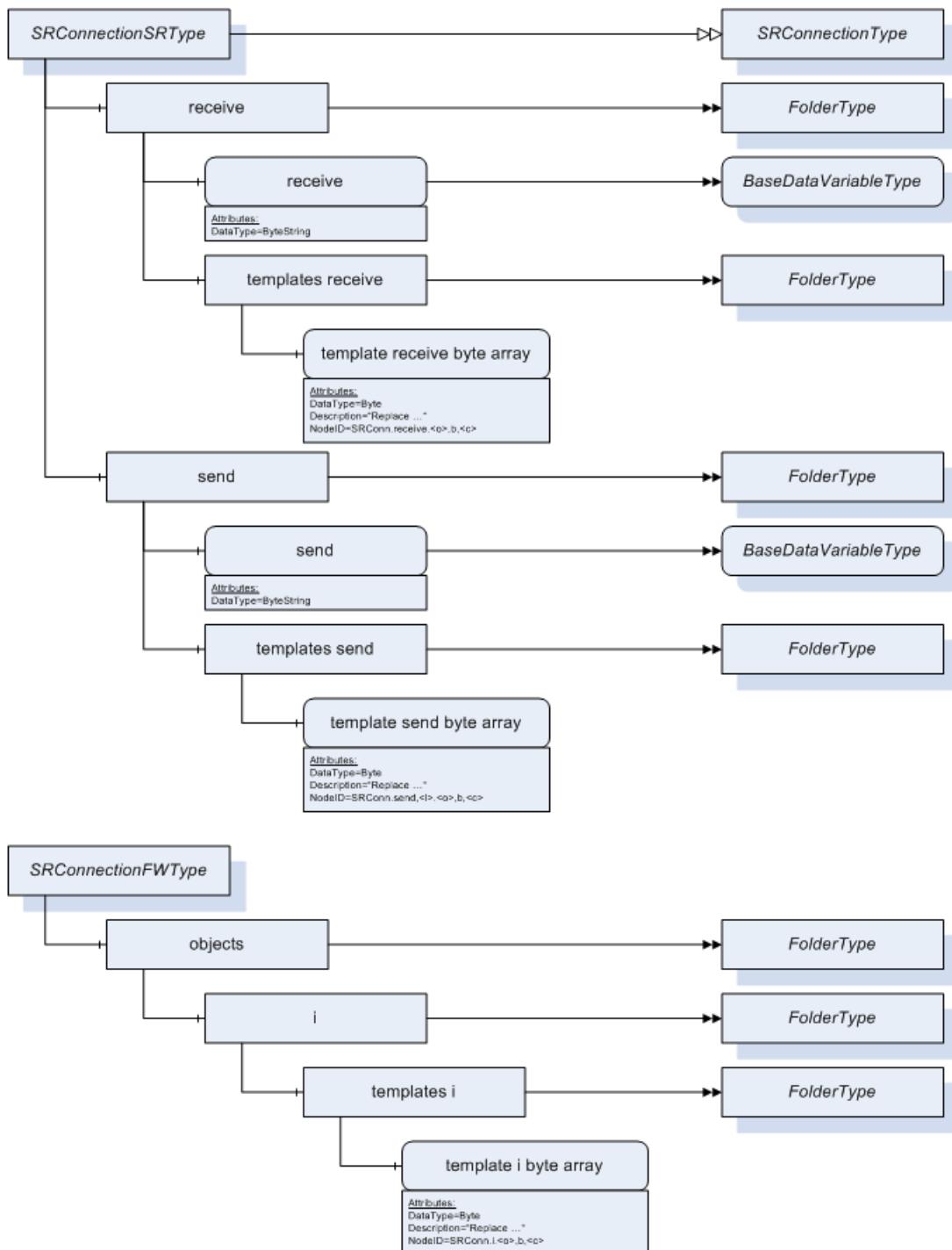
即使单字节长度的数据块也有大约 40 个不同的数据格式选项 - 从 Byte 和 SByte

开始，由这些数据类型的一个元素组成的数据块、单个位、最多 8

个数组元素的位数组，每个元素均起始于不同的位偏移。

因此，OPC UA 服务器支持 SR 命名空间中具有模板数据变量的用户。在典型的 OPC UA 客户端文本输入框中，只需更改几个字符，就可将这些模板转换为有效的 ItemID。

## 2.10 通过工业以太网实现的 OPC UA 的开放式通信服务 (SEND/RECEIVE)



### 说明

可以在“通信设置”组态程序中启用和禁用 OPC UA SR 模块数据变量的可用性：“OPC 协议选择”(OPC protocol selection) > 单击“SR”旁的箭头符号。

### 浏览层级结构中的模板数据变量

在命名空间表示中相应的文件夹旁将模板数据变量进行排序以便需要时便于使用。

### 模板数据变量的语法

可用选项如下：

- <SRconnectionname>send,<bl>.<o>,<SRtypetemplate>,<c>
- <SRconnectionname>receive.<o>,<SRtypetemplate>,<c>
- <FETCHconnectionname>.<datavariable>.<o>,<SRtypetemplate>,<c>
- <WRITEconnectionname>.<datavariable>.<o>,<SRtypetemplate>,<c>

### 说明

<SRconnectionname>、<FETCHconnectionname> 和 <WRITEconnectionname> 协议特定的连接名称。连接名称在组态中指定。

#### <bl>

发送缓冲区长度的占位符。

#### <o>

字节偏移量的占位符。

#### <SRtypetemplate>

SR 模板数据类型将转换为 OPC UA 服务器上相应的 OPC UA 数据类型。

下表列出了数据类型以及可用于表示变量值的相应 OPC UA 数据类型。

数据类型	OPC UA 数据类型	注释
x<bitaddress>	位 (布尔)	VT_BOOL
b	字节 (无符号 8 位)	VT_UI1
char	字节 (有符号 8 位)	VT_I1

数据类型	OPC UA 数据类型	注释
w	字 (无符号 16 位)	VT_UI2
int	字 (有符号 16 位)	VT_I2
dw	双字 (无符号 32 位)	VT_UI4
di	双字 (有符号 32 位)	VT_I4
r	浮点, IEEE 表示	VT_R4
s5r	浮点, S5 表示	VT_R4

**<datavariable>**

与 S5 兼容的通信的数据变量。

数据变量		注释
db<db>	无符号 8	数据块
dx<dx>	无符号 8	已扩展的数据块
de<de>	无符号 8	已扩展的数据块
i		输入
q		输出
p		外围设备 I/O 的输入和输出
qb		已扩展的 I/O
m		位存储器
c		计数器
t		定时器

**<c>**

元素数量的占位符。

示例：

Nodeld	BrowseName	说明
SRconnectionname.send,<bl>.<o>,b	模板发送字节	<bl> 发送缓冲区的长度 <o> 偏移量
SRconnectionname.receive.<o>,b	模板接收字节	<o> 偏移量
FETCHconnectionname.i.<o>,x<bit>,<c>	模板 i 位数组	<o> 偏移量 <bit> 位偏移量 (0 到 7) <c> 数组大小
WRITEDBconnectionname.db<db>.<o>,x<bit>,<c>	模板 db 位数组	<db> 数据块的编号 <o> 偏移量 <bit> 位偏移量 (0 到 7) <c> 数组大小

## 2.11 通过工业以太网实现的 SNMP 通信

### SNMP 的过程变量

SNMP 数据主要为诊断变量（写访问也是设备参数）。

SIMATIC NET 的 SNMP OPC 服务器提供了以下变量：

- 过程变量
- 信息变量
- 陷阱变量

关于 SNMP OPC 服务器的详细信息：

(<http://www.automation.siemens.com/mcms/industrial-communication/en/ie/software/network-management/snmp-opc-server/Pages/snmp-opc-server.aspx>)

## 2.11.1 协议 ID

### 协议 ID

SNMP 协议的协议 ID 是“SNMP”。

## 2.11.2 SNMP 协议的数据类型

### 可用数据类型上的映射

在 MIB 对象的声明中，SNMP 协议使用其自身的数据类型。SNMP OPC 服务器将按如下方式将 SNMP 协议的数据类型映射到 OPC 接口的可用数据类型上：

SNMP 数据类型 代码	标准的 OLE 数据类型	Visual Basic 类型	含义
ASN_INTEGER 02h	VT_I4	长整型	有符号长整型 (4 字节)
ASN_INTEGER32< 02h	VT_I4	长整型	有符号长整型 (4 字节)
ASN_UNSIGNED32 47h	VT_UI4	双精度	无符号长整型 (4 字节)
ASN_OCTETSTRING 04H	VT_BSTR	字符串	如果不是可打印的字符 , 则表示为: <i>ostring:xx.xx.xx.xx</i>
ASN_OBJECTIDENTIFI ER 06h	VT_BSTR	字符串	用 MIB 表示法表示： .a.b.c.d.e.
ASN_IPADDRESS 40h	VT_BSTR	字符串	以十进制表示法表示： 129.168.0.4
ASN_COUNTER32 41h	VT_UI4	双精度	无符号长整型 (4 字节)
ASN_GAUGE32 42h	VT_UI4	双精度	无符号长整型 (4 字节)

SNMP 数据类型 代码	标准的 OLE 数据类型	Visual Basic 类型	含义
ASN_TIMETICKS 43h	VT_UI4	双精度	无符号长整型 (4 字节)
ASN_OPAQUE 44h	VT_ARRAY of VT_UI1	VT_ARRAY of VT_UI1	无符号字节

### 2.11.3 SNMP 变量服务的过程变量

#### 变量和设备配置文件

SNMP 变量是伙伴设备上的变量。 OPC 服务器可基于设备配置文件获知这些变量。

这些设备配置文件可用 MIB 编译器以及设备的 MIB 文件创建。

标准设备配置文件随组态工具提供。 然后，将所定义的组态下载至带有 OPC 服务器的 PC 中以使这些变量可用于 OPC 服务器。

#### 语法

---

##### 说明

与其他协议的语法不同，下列语法用于 SNMP。

---

SNMP: [<devicename>]<objectname>

---

#### 说明

##### SNMP

访问过程变量（MIB 对象）和陷阱变量的 SNMP 协议。

##### <devicename>

系统组态期间指定的唯一的设备名称。

##### <objectname>

伙伴设备的 MIB 对象的符号名称。

#### 示例

*SNMP:[OSM]sysName*

查询 MIB 对象 sysName。在系统组态中对设备名称进行了组态，在该示例中为 OSM。

## 2.11.4 SNMP 特定的信息变量

### 简介

以下部分说明了 SNMP OPC 服务器所提供的变量：

- 通信系统和连接状态的变量
- SNMP OPC 服务器的变量

### 有关 SNMP 设备的信息

OPC 服务器所提供的变量可用于获取有关 SNMP 设备（例如：端口、开关）的信息。

可查询以下信息：

- 在组态过程中输入的注释
- 伙伴设备的已组态 IP 地址
- 与伙伴设备的连接状态
- 不与 SNMP 相兼容设备的附加信息

### 语法

SNMP: [<devicename>]<informationparameter>

### 说明

#### SNMP

用于访问过程变量（MIB 对象）和陷阱变量的 SNMP 协议。

#### <devicename>

系统组态期间指定的设备名称。

## &lt;informationparameter&gt;

下列信息参数的含义：

&description()	组态期间在注释字段中输入的文本。 VT_BSTR, 只读	
&ipaddress()	伙伴设备的 IP 地址。 VT_BSTR, 只读。	
&statepath()	与伙伴设备的连接状态。 文本形式的值 (VT_BSTR, 只读)：  <i>DOWN</i> 无法用 ping 访问设备或没有活动的读取或写入作业。 <i>UP</i> 可用此设备成功执行作业。  <i>RECOVER</i> 不能用 ping <i>Y</i> 访问设备, 或最后的作业终止, 并显示一条错误信息, , 表示通信中断但设备仍有活动的作业。	
&statepathval()	与伙伴设备的连接状态。整型值。 VT_UI1, 只读。	
	1	无法用 ping 访问设备或没有活动的读取或写入作业。
	2	可用此设备成功执行作业。
	3	不能用 ping 访问设备, 或最后的作业终止, 并显示一条错误信息, 表示通 信中断但设备仍有活动的作业。
&ping()	与伙伴设备的连接状态。 VT_UI1, 只读。	
	0	无法用 ping 访问设备。
	1	可以用 ping 访问设备。
&alarmagentmib2() ( )	与 SNMP 代理伙伴设备的连接状态 (可访问 MIB2 配置文件的项目 <i>sysUpTime</i> )。 VT_UI1, 只读。	
	0	无法读取“sysUpTime”。
	1	可读取“sysUpTime”。

只有在组态期间选择“无 SNMP”(No SNMP)

后，以下信息参数才处于可见状态，因为设备仅支持 ping 而不支持 SNMP。

<b>&amp;syscontact()</b>	组态期间在 sysContact 字段中输入的文本。 VT_BSTR, 只读。
<b>&amp;syslocation()</b>	组态期间在 sysLocation 字段中输入的文本。 VT_BSTR, 只读。
<b>&amp;sysname()</b>	组态期间在 sysName 字段中输入的文本。 VT_BSTR, 只读。

## 示例

*SNMP:[OSM]&ipaddress()*

返回带有名称 OSM 的节点名称的已组态 IP 地址。

## 有关 SNMP OPC 服务器的信息

OPC 服务器所提供的变量可用于查询有关 SIMATIC NET SNMP OPC 服务器的信息。

可查询以下信息：

- SIMATIC NET SNMP OPC 服务器的版本
- Winsocket 的版本
- 有关陷阱接收的信息

## 语法

*SNMP: [SYSTEM] &<informationparameter>()*

## 说明

### SNMP

用于访问本地系统信息变量的 SNMP 协议。

### SYSTEM

本地系统的标识符；固定名称。

## &lt;informationparameter&gt;

下列信息参数的含义：

&version()	SIMATIC NET SNMP OPC 服务器的版本标识符。 VT_BSTR, 只读。
&winsockversion()	Winsocket 的版本标识符。 VT_BSTR, 只读。
&traplisten()	指示 SIMATIC NET SNMP OPC 服务器是否能够登陆以接收陷阱。 VT_BOOL; 只读。  <i>FALSE</i> SNMP OPC 服务器无法登陆以接收陷阱。  <i>TRUE</i> SNMP OPC 服务器能够成功登陆来接收陷阱。

## 示例

*SNMP:[SYSTEM]&version()*

返回 SIMATIC NET SNMP OPC 服务器的版本，例如 *SIMATIC NET Core Server SNMP V6.1.1000.2815 Copyright ©SIEMENS AG*

## 2.11.5 SNMP 特定的陷阱

## OPC 服务器处理陷阱的方法

陷阱是设备自动发送至 OPC SNMP 服务器的事件。 OPC 服务器将按下列方式处理这些事件：

- 陷阱将作为简单事件映射到报警和事件接口上。
- 对于每个已组态的陷阱，将在数据访问接口上创建两个变量：  
一个变量将随相关陷阱的出现而递增，另一个变量将存储对陷阱的描述。

## 语法

事件发生次数的第一个变量：

*SNMP:[<devicename>]<trapname>*

用于描述陷阱的第二个变量：

---

*SNMP:[<devicename>]<trapname>\_description*

## 说明

### SNMP

用于访问过程变量（MIB 对象）和陷阱变量的 SNMP 协议。

#### <devicename>

系统组态期间指定的设备名称。

#### <trapname>

陷阱的名称。

## 示例

第一个变量将返回由伙伴设备触发的冷启动陷阱的数量：

*SNMP:[OSM]coldStart*

第二个变量将返回对陷阱的描述：

*SNMP:[OSM]coldStart\_description*

## 2.12 通过工业以太网实现的 PROFINET IO 通信

分配给 PROFINET IO 控制器的 PROFINET IO 设备的系统模型为面向模块的模型。

一台 PROFINET IO 设备可包含若干个模块，每个模块可包含若干个子模块。

一般来说，一个子模块包含待连接设备硬件的物理终端或驱动程序块（通道），例如：传送带或传感器，将采用 IO 数据对它们的输入数据或调节变量进行寻址。读取和写入 IO 数据是 PROFINET IO 最重要的应用。

设备、模块/子模块均可提供数据记录并生成中断。数据记录和中断的使用与设备相关。

### 2.12.1 适用于 PROFINET IO 协议的功能强大的 SIMATIC NET OPC 服务器

## 简介

PROFINET IO 协议的组态变量满足更高的性能要求。低级别的 PROFINET IO COM 服务器将作为进程内服务器在进程外 OPC 服务器上进行加载。如果在 OPC 服务器的进程中处理协议，则可避免在进程和多协议模式之间切换的附加执行时间。OPC 客户端和 OPC 服务器之间的过程切换仍然必不可少。

## 2.12 通过工业以太网实现的 PROFINET IO 通信

### 组态

通过在“通信设置”组态程序中选择“PROFINET IO”协议作为唯一协议，隐式启用此高速变型（如果选择了其它协议或 OPC UA 接口，则会失去下面介绍的性能优点）：



图 2-60 “通信设置”组态程序中用于选择 PROFINET IO 协议的窗口

也可选择“符号”(Symbols)。

### 说明

使用 STEP 7 或符号编辑器创建符号时，允许使用以下字符：A-Z、a-z、0-9、\_、-、^、!、#、\$、%、&、'、/、(、)、<、>、=、?、~、+、\*、'、:、|、@、[、]、{、}、"。使用 STEP 7 创建符号时，还应记住，如果符号文件中的符号同时具有 `<symbolname>` 和 `<symbolname>[<索引>]` 形式，则解析数组时会出现问题。

## 优点/缺点

不过，使用功能强大的 SIMATIC NET OPC 服务器具有以下缺点：只能采用 PROFINET IO 单协议模式。

但另一方面，则具有以下优点：

- 比使用多协议模式的性能更高
- 简单组态
- 通过 ProgID“OPC.SimaticNET”进行访问
- 多个客户端可同时使用服务器
- OPC 服务器的稳定性不取决于客户端。

### 说明

服务器的组态循环时间并不是自动采用调用下级 PROFINET IO 协议接口的时间。根据 OPC 客户端更新时间的不同，调用率也可以变慢。

### 说明

如果已经启用了“通信设置”程序的“安全性”(Security) 对话框中的“锁定...”(Lock...) 按钮，若要通过 PROFINET IO OPC 服务器恢复远程通信，需要单击“允许”(Allow) 按钮（远程基本通信和 OPC 通信）。

## 2.12.2 IO 数据的寻址方式是什么？

通常，不支持对涉及多个子模块、模块或设备的 IO 数据进行寻址。

无法进行内部组合调用；也就是说，必须单独对每个 PROFINET IO 地址进行读取或写入操作。

### 寻址

使用逻辑寻址。

逻辑地址参考了由控制器管理的过程映像中所有设备的 IO 区域的一个映像。

对于 I 和 Q 区域 ( $2^{32}-1$ )，每个过程映像的长度为 4 千兆字节。可使用 PROFINET IO 组态工具分配过程映像。逻辑地址指定了过程映像中设备 IO 区域的字节偏移量。

逻辑寻址是读取和写入 IO 数据的理想寻址方式。如果用户小心组态，则设备的 IO 映像将紧密地排列在一起。可同时读取和写入几个设备的连续区域。

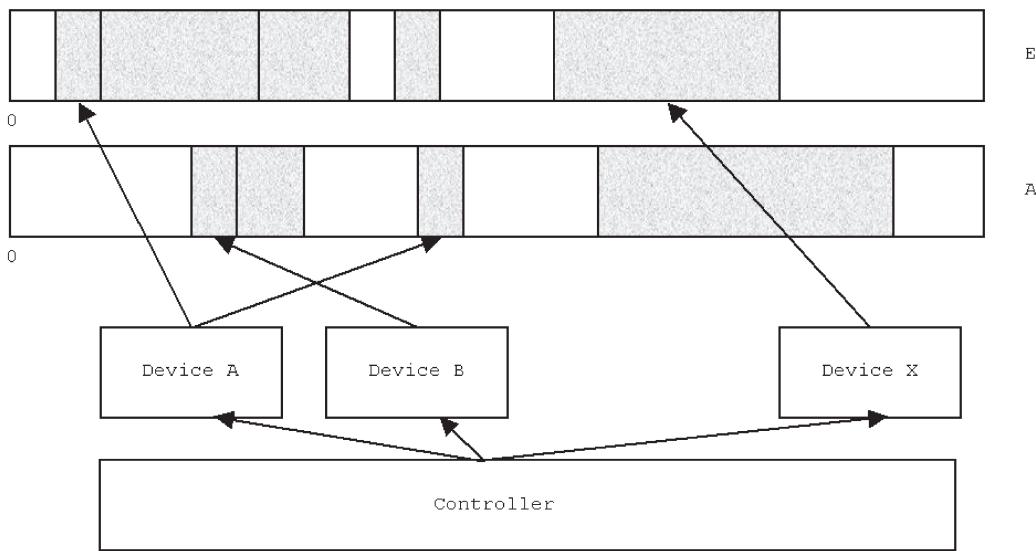


图 2-61 通过控制器在单独或连续区域内读取和写入设备的 IO 数据

### 示例

对带有 OPC 服务器应用程序和 IM 151 的控制器 CP 1612 PROFINET IO 进行寻址的示例。

地址有：

- 输入 0，长度 4 位
- 输出 0 和 1，长度 2 位

IM151 的诊断地址是 16382。

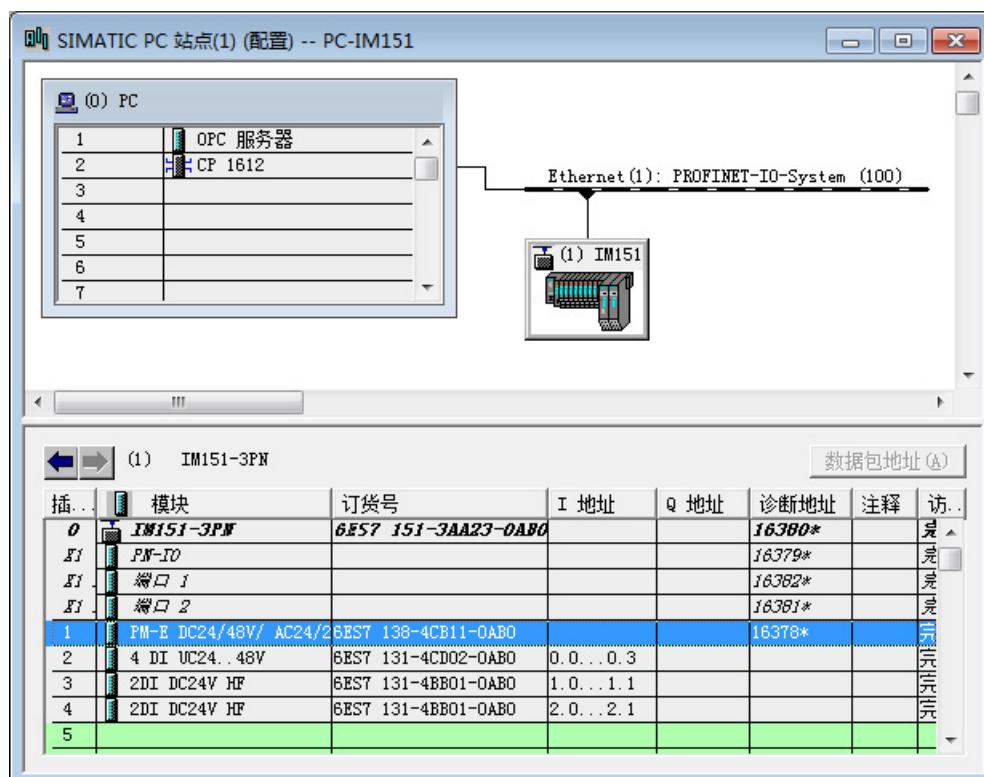


图 2-62 PC 站模块地址在“硬件组态”站窗口中的显示

### 在实际情况下寻址的扩展方法是什么？

例如，如果仅想读取一个宽度为 64 位的子模块的单个输入位，那么必须读取全部的 8 个字节。这对于写入和输出位也适用；必须写入全部的 8 个字节。

设备结构；也就是说，组态有关实际现有设备及其模块和子模块的全部信息，以便用户获知此信息。然而，它必须能够在运行过程中移除或插入模块和子模块。

转换 IO 数据时，为向 OPC 用户提供一定程度的便利，引入了扩展寻址。

用户（假定其对该设备非常了解）可在本地通过数据项语法为 PROFINET IO OPC 服务器提供必要的组态信息。

#### 示例：

假设 PROFINET IO 设备通过 IO 子模块控制 64 个电子开关。

通过组态软件将子模块映射至逻辑基址 400，此模块有 64 位的取值范围，即 8 个字节。此信息是预先定义的。通过使用下列数据项，用户可以读取或写入从偏移量 4 开始的子模块的位 5。

*PNIO:[ctrl1]QB400,8,X4.5*

### 说明

请注意，使用简单寻址时，读写 PROFINET IO

数据需要精确的组态长度信息（使用高级寻址时，可以使用部分寻址）。

示例：

无法使用 OPC 项 PNIO:[CTRL1]IWORD0 (2 字节) 对 4 个字节长度的 IO 模块进行部分读取。

这种情况下，可使用：“PNIO:[CTRL1]IB0,4”或“PNIO:[CTRL1]IDWORD0”。

---

### 2.12.3 如何浏览已组态的 PROFINET IO 命名空间？

OPC 服务器的已组态 PROFINET IO 设备可通过 OPC DA 的浏览功能进行显示。

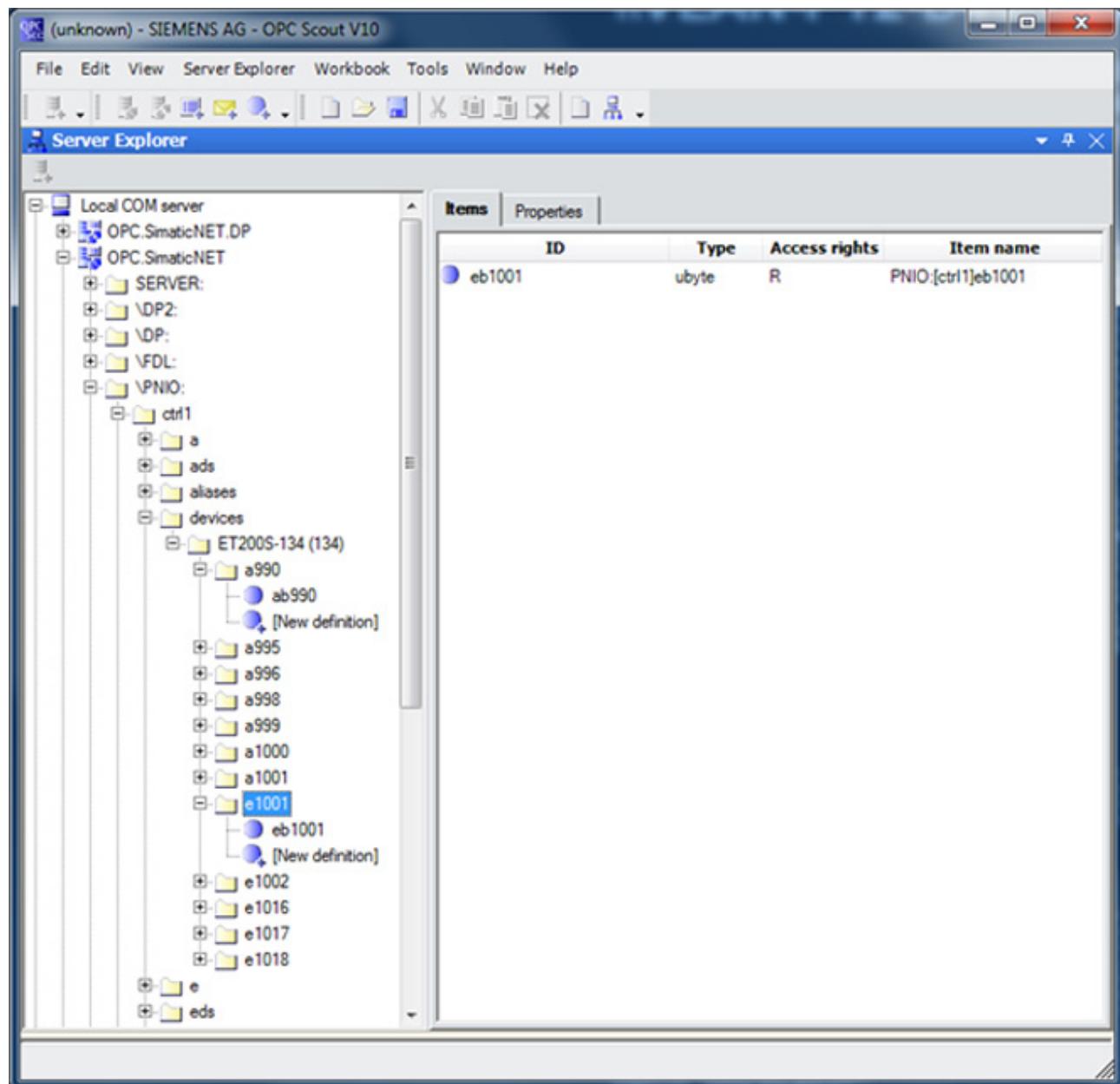


图 2-63 通过 OPC Scout 浏览已组态的命名空间

显示现有输入和输出以及它们的地址。

由于组态数据长度已知，因此预定义数据项有效。

## 2.12 通过工业以太网实现的 PROFINET IO 通信

在工厂视图中，所有组态设备及其名称（子模块在其下方）均在“设备”分支中列出。显示每个子模块现有的所有输入和输出以及它们的地址。预定义数据项已存在。

### 2.12.4 哪些 OPC 变量可用于 PROFINET IO？

SIMATIC NET 的 PROFINET IO OPC 服务器提供了以下变量类型：

- 过程变量 - 读取/写入/监视数据
- 信息变量 - 状态查询
- 控制变量 - 启用/禁用设备

### 2.12.5 协议 ID

PROFINET IO 协议的协议 ID 是“PNIO”。

### 2.12.6 控制器名称

连接名称通过指定连接到 PROFINET 以太网的通信处理器来指定通信访问。

在 PROFINET IO 协议下，控制器的名称是通信模块的名称，该模块包括 PC 站（站组态编辑器）中的插槽（索引）。

控制器名称 = *CTRL<Index>*

控制器名称示例：

- CTRL3
- CTRL5

### 2.12.7 PROFINET IO 过程变量

读取和写入 PROFINET IO 过程变量的数据项是最常见的应用。可进行简单访问。

用户必须指定项目工程中的控制器名称及地址。

可在 OPC 命名空间中浏览组态的地址空间。

添加数据项时，将检查这些地址。

## 简单寻址

### 语法

输入:

```
PNIO:[<controllername>]I<format><address>
{.<bit|stringlength>}[,<quantity>]
```

输出:

```
PNIO:[<controllername>]Q<format><address>
{.<bit|stringlength>}[,<quantity>]
```

### 说明

#### PNIO

用于访问过程变量的 PROFINET IO 协议。

#### <devicename> = CTRL<索引>

协议特定的设备名称。PC 站组态期间指定设备名称的索引。值范围为 1 到 32

示例: CTRL1

|

输入的标识符。输入为只读。

#### Q

输出的标识符。可读取和写入输出。

#### <format>

所传送数据的格式。

指定格式即指定数据类型。

原则上，所有列出的 OLE 数据类型都可通过 OPC

的自动化和自定义接口读取。然而，某些开发工具，如 Visual Basic  
仅能提供有限数量的数据类型。

因此，下表列出了可用于表示变量值的相应 Visual Basic 类型。

长度超过一个字节的数据类型将以 Motorola 格式解释。

格式标识符	OLE 数据类型	Visual Basic V6.0 型自动化 数据类型	说明
X	VT_BOOL	布尔	<p>此外，也必须指定相关字节中的位地址。 出于一致性方面的考虑，只有使用扩展寻址才能写入各个位；但，OPC 客户端应始终同时写入逻辑地址的所有位。 数组长度也可以用位来指定。 任何时候都可以读取位数组，但出于一致性方面的考虑，只有当位地址有值 0 且位长度为 8 的倍数时才能进行写入操作。 但为了保持统一性，也只有当位地址有值 0 且位长度为 8 的倍数时才能读取位数组。</p>
B 或 BYTE	VT_UI1	字节	字节（无符号）
W 或 WORD	VT_UI2	长整型	字（无符号）
D 或 DWORD	VT_UI4	货币型	<p>双字（无符号） 由于不存在对应的数据类型，需要映射至更多的 Visual Basic 型 6.0 货币 8 字节。</p>
LWORD	VT_UI8	VT_UI8	自动化型整型 8 字节（无符号），Visual Basic 6.0 不支持该数据类型。
CHAR	VT_I1	整型	字节（有符号）映射至更广泛的 Visual Basic 型整型 2 字节中。
INT	VT_I2	整型 (%)	字（有符号）
DINT	VT_I4	长整型 (&)	双字（有符号）
LINT	VT_I8	VT_I8	自动化型整型 8 字节（有符号），Visual Basic 6.0 不支持该数据类型。
REAL	VT_R4	单精度	浮点 4 字节

格式标识符	OLE 数据类型	Visual Basic V6.0 型自动化 数据类型	说明
LREAL	VT_R8	双精度	浮点 8 字节
String	VT_BSTR	字符串	字符串 还必须指定为字符串保留的长度。 写入时还可写入较短的字符串，使传送的数据长度始终为保留的字符串长度（字节）。 不需要的字节将以值 0x00 填充。

**<bit|stringlength>**

指定数据类型 X: 已寻址字节中的位编号。

范围为 0 到 7。

指定数据类型 STRING: 保留的字符串长度。

对于格式标识符 X 和 STRING，也必须指定 .<bit|stringlength>。

**<quantity>**

元素数量

只有在满足以下条件时才能读取和写入位数组

- 位地址有值 0
- 其长度是 8 的整数倍
- 达到确切的模块长度。

**<address>=<逻辑地址>**

简单逻辑寻址

设备的逻辑基址跟项目工程中设置的相同

值范围为 0 到 4 294 967 295。

**说明**

请注意，只有在使用精准组态的长度时才可以读取和写入 PROFINET IO 数据。

例如，在一定程度上来说，使用 OPC 数据项 PNIO:[CTRL1]QWORD0

不能将一个长度为 4 字节的 IO 模块写入一个长度仅为 2 字节的 IO 模块。

以下是所允许的格式，例如 PNIO:[CTRL1]QB0,4 或 PNIO:[CTRL1]QDWORD0。

## 扩展的寻址

扩展的地址包括整个子模块的数据区域。此信息可在组态中找到。

### 语法

输入:

```
PNIO:[<controllername>] I<expanded address>,
      <format><offset>{.<bit|stringlength>}[,<quantity>]
```

输出:

```
PNIO:[<controllername>] Q<expanded address>,
      <format><offset>{.<bit|stringlength>}[,<quantity>]
```

### 附加的解释

**<expandedaddress>** =

B 或 BYTE<logicaladdress>、<numberofsubareas>

扩展的地址描述了子模块的整个地址区域，在该区域内可对下至单个位的任何子区域进行寻址。

扩展地址通过从逻辑基址开始的字节编号来描述。

**<offset>**

要寻址元素所在的已寻址子模块内的字节偏移量。

---

### 说明

使用扩展的寻址，可以对各个位进行读取和写入操作。

也可以在扩展的地址内读取和写入各个位组成的数组，而无需考虑起始地址和数组长度。

---

## PROFINET IO 过程变量的示例

下面几个示例说明了 PROFINET IO 变量的变量名称的语法。

- 输入

*PNIO:[CTRL3]IB10*

控制器索引 3, 逻辑地址 10, 首个输入字节。

*PNIO:[CTRL3]IB4,3*

控制器索引 3, 逻辑地址 4, 前 3 个字节的数组。

*PNIO:[CTRL 1]ID2*

控制器索引 1, 逻辑地址 2, 首个双字。

*PNIO:[CTRL 1]IX10.0,64*

控制器索引 1, 逻辑地址 10.0, 前 64 个输出位的数组。只取从第一个字节开始的 8 的倍数。

*PNIO:[CTRL 1]IReal4*

控制器索引 1, 逻辑地址 4, 输入区域中的首个浮点数。

**扩展的寻址:**

*PNIO:[CTRL 1]IB2,1,X1.0,2*

控制器索引 1, 扩展地址 2, 区域数量 1 个字节, 偏移量 1 个字节, 从输入位 0 开始的前 2 位的数组。

*PNIO:[CTRL 1]IB10,4,X1.3,7*

控制器索引 1, 扩展地址 10, 区域数量 4 个字节, 偏移量 1 个字节, 从输入位 3 开始的前 7 位的数组。

- 输出

*PNIO:[CTRL3]QB7*

控制器索引 3, 逻辑地址 7, 第一个输出字节。

*PNIO:[CTRL 1]QW0,8*

控制器索引 1, 逻辑地址 0, 前 8 个字的数组。

*PNIO:[CTRL3]QX2.0,64*

控制器索引 3, 逻辑地址 2.0, 前 64 个输出位的数组, 可读写。从字节开始仅取 8 的倍数。

*PNIO:[CTRL 1]QSTRING100.32,3*

控制器索引 1, 逻辑地址 100, 前 3 个长度为 32 个字符的字符串的数组。

扩展的寻址:

*PNIO:[CTRL 1]QB1,1,X0.0*

控制器索引 1, 扩展地址 1, 区域数量 1 个字节, 偏移量 0 个字节, 首个输出位 0。

*PNIO:[CTRL 1] QB2,4,X1.0,2*

控制器索引 1, 扩展地址 2, 区域数量 4 个字节, 偏移量 1 个字节, 从输出位 0 开始的前 2 位的数组。

## 2.12.8 PROFINET IO 数据状态

使用状态 (IOPS 和 IOCS) 的项目, OPC 客户端可以查询或设置逻辑地址的状态字节。

下图显示了过程映像中的数据状态和值流。

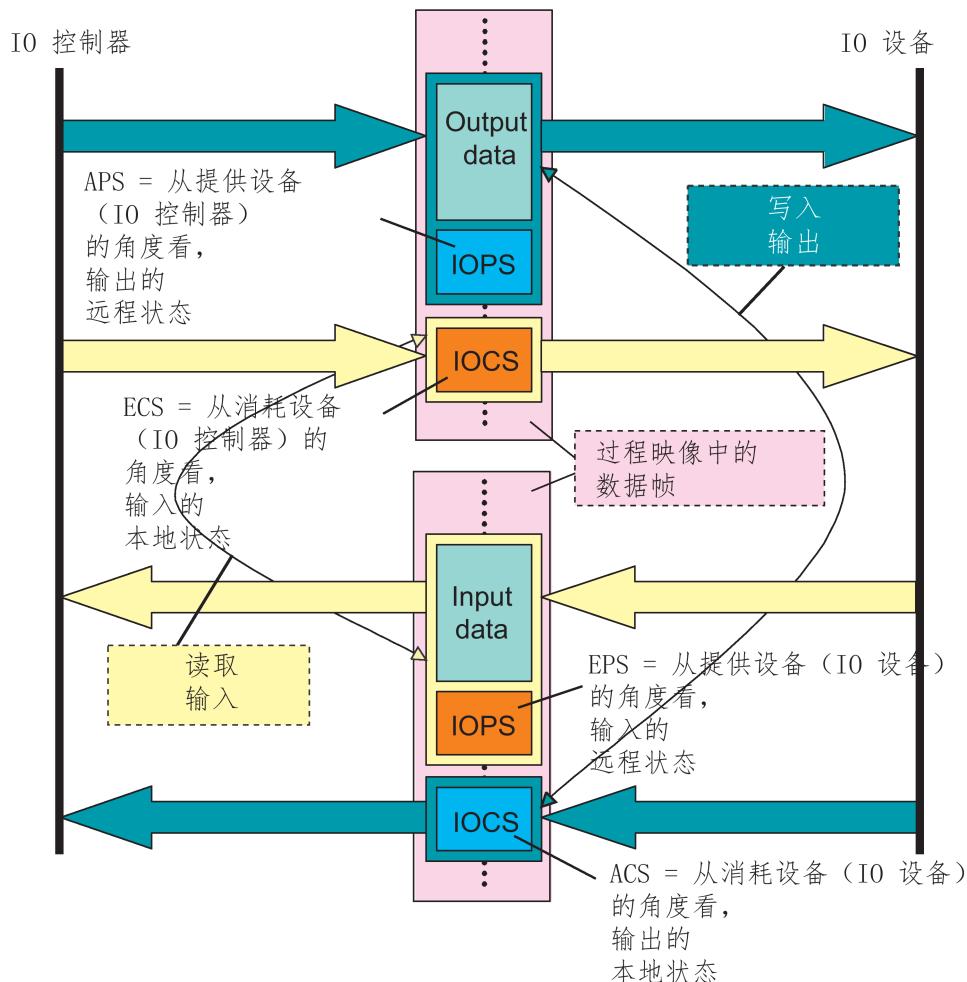


图 2-64 在 IO 控制器和 IO 设备之间进行 PROFINET IO 通信时过程映像中的数据状态概述

从“QPS”的供应商角度看, OPC

客户端向过程映像写入输出数据时, 该客户端包括远程输出状态。

从“QPS”的用户角度看, 设备包括其自身的本地输出状态。

从具有“**ICS**”的用户角度来看，当 OPC 客户端从过程映像读取输入数据时，其中包括本地输入状态。从具有“**IPS**”的提供者角度来看，设备包括远程输入状态。

### 说明

#### 对循环输出数据的读访问

读取、监控或激活用于映射循环输出数据的 OPC 数据项/节点将引起底层 PROFINET IO 协议接口的一次更新或不断更新，其中，数据包含在包括提供者状态 (PNIO\_data\_write()) 的 OPC 服务器缓存中。请记住：在 PROFINET IO 设备或模块返回后，该输出数据将在更新的时间点之后再次传送到设备中。

### 语法

PNIO: [<controllername>]<I | Q><PS | CS><address>

### 说明

#### **PNIO**

用于访问数据记录的 PROFINET IO 协议。

#### <controllername> = CTRL<index>

协议特定的控制器名称。在 PC 站组态期间指定控制器名称索引。取值范围是 1 到 32

#### I

输入标识符。输入为只读输入。

#### Q

输出标识符。可读取和写入输出。

#### PS

远程状态标识符（供应商状态）。

#### CS

本地状态标识符（用户状态）。

或

**IPS** 远程状态输入

**QPS** 远程状态输出

**ICS** 本地状态输入

**QCS** 本地状态输出

**<address>=<logical address>**

简单逻辑寻址

设备的逻辑基址跟组态中设置的相同。

取值范围为 0 到 4 294 967 295。

名称	说明
IPS	<p>输入地址的远程状态</p> <p>数据类型: VT_I4, 只读。</p> <p>从 OPC 服务器的角度来看, 输入的数据供应商是设备。设备还提供输入数据的状态。</p> <p>IPS 由轮询进行监控。</p> <p>OPC 应用程序并不绝对需要使用项, 因为项内容是在具有相同地址的链接输入项的 OPC 质量中进行反映。</p>
ICS	<p>输入地址的本地状态</p> <p>数据类型: VT_I4, 读写。</p> <p>默认值: <i>GOOD</i> = 0</p> <p>从 OPC 服务器的角度来看, 输入的数据用户是 OPC 客户端。从自身角度来看, OPC 客户端也会将输入数据的状态告知设备。通常, 该状态始终为 <i>GOOD</i>。</p> <p>读取 IO 数据时, 始终包括由 OPC 应用程序写入的输入地址的最后本地状态 (已写入到伙伴设备)。如果 OPC 应用程序未创建项或未将一个值写入到其中, 则在读取相应输入数据时, 会传送值 <i>GOOD</i>。</p> <p>状态由 OPC 客户端通过缓存进行读取。</p>

名称	说明
QCS	<p>输出地址的本地状态</p> <p>数据类型: <i>VT_I4</i>, 只读。</p> <p>从 OPC 服务器的角度来看, 输出的数据用户是设备。</p> <p>从自身角度来看, 设备也包括输出数据的状态。</p> <p>如果 OPC 客户端首先 (或同时) 写入具有相同地址的链接输出项, 则可读取状态。</p> <p>QCS 由轮询监视, 同时始终将 OPC 客户端写入的最后一个输出数据传送到函数中。如果 OPC 客户端无法成功写入一个值, 则从设备进行读取将导致错误或使 OPC 质量变为不良。</p>
QPS	<p>输出地址的远程状态</p> <p>数据类型: <i>VT_I4</i>, 读写。</p> <p>默认值: <i>GOOD = 0</i></p> <p>从 OPC 服务器的角度来看, 输出的数据供应商是 OPC 客户端。从自身角度来看, OPC 客户端也会将输出数据的状态告知设备。通常, 该状态始终为 <i>GOOD</i>。</p> <p>写入 IO 数据时, 也包括由 OPC</p> <p>应用程序写入的输出数据的最后本地状态和输出地址的输出数据 (可写)。如果 OPC 应用程序尚未创建 IO 项或尚未成功写入一个输出值, 则写入相应输出数据时, 将传送值 <i>GOOD</i>。</p> <p>当 OPC</p> <p>客户端写入一个新状态时, 使用先前 (或同时) 成功写入的数据值, 而该数据值具有链接输出项的地址。</p> <p>如果 OPC 客户端尚未成功写入一个数据值, 则写入项将产生特殊返回值 (<i>S_OPN_COREPNIOSTATUS</i>)。</p> <p>然后, 写入状态便可由具有下一成功写入的相应输出项进行写入。</p>

### 状态值的结构

状态值, 类型 *VT\_UI4* 始终具有以下结构:

0 = GOOD

1 = BAD

## 2.12.9 PROFINET IO 数据记录

通过数据记录功能, 可以读出设备特定的信息或写入参数。

数据记录的含义由供应商指定。例如, 数据记录可用于驱动器的组态数据。

注册数据记录变量后, OPC

服务器只能检查语法是否正确，而不能根据设备组态检查变量在伙伴设备上是否有效以及数据记录大小是否足够。

## 语法

```
PNIO: [<controllername>]<I|Q>DS<address>, DATA<datarecordindex>,
{<datarecordlength>} {,<subelement>}
```

## 说明

### PNIO

用于访问数据记录的 PROFINET IO 协议。

#### <controllername> = CTRL<索引>

协议特定的控制器名称。在 PC 站组态期间指定控制器名称索引。

值范围：1 ... 32

|

输入的标识符。

### Q

输出的标识符。

### DS

数据记录的标识符。

#### <address>=<逻辑地址>

简单逻辑寻址

设备的逻辑基址跟组态中设置的相同。

值范围为 0 到 4 294 967 295。

### DATA

数据的标识符。

- <datarecordindex>

数据记录索引，设备特定的参数。

值范围：0 ... 65535。

在 PROFINET IO 规范中标准化的一部分数据记录索引如下所示：

0x8030 = 32816 读取输入数据对象

0x8031 = 32817 读取输出数据对象

0x8020 = 32800 写入输出数据对象的替换值

60k+10 = 0xF00A = 61450 读取 IO 设备级的诊断信息

56k+10 = 0xE00A = 57354 读取 AR 级的诊断信息

$48k+10 = 0xC00A = 49162$  读取插槽级的诊断信息

$32k+10 = 0x800A = 32788$  读取子插槽级的诊断信息

- <datarecordlength>

数据记录长度。

读取数据记录时，将向设备告知要以字节读取的数据记录的准确长度。然后，OPC 客户端必须指定长度。如果指定了长度，则也可写入数据项。

如果未指定长度，则不能写入数据项。

在进行读取操作时，数据记录长度是可选的，而且也可以被忽略掉。

在这种情况下，（返回）值数组的长度是变量。

用户无需知道数据记录的长度（其取值范围为：0 ... 65535）。0 ... 65535).

- **<subelement>**

要在构建数据记录时支持用户，可选择一个或多个子区域。再次使用上述 IO 过程变量的有效数据类型。

读取：

由于从伙伴读取的数据记录的结构和长度无需固定，因此允许定义和请求区域外的数据。若在接收数据时不再填充此区域，则变量的质量将会相应地发生变化。

写入：在写入子区域时，将始终发送整个数据记录。若在 OPC

写入作业中写入数据记录的若干个子区域，则数据记录只能在其子区域全部更新后才能发送。

在数据记录中，数据类型以 Motorola 格式进行解释，然后转换为相应的 Intel 格式。

*<subelement>=<format><offset>{<bit/stringlength>}<quantity>}*

- **<offset>**

子元素相对于被寻址数据记录的字节偏移量。

值 *<format>*、*<bit/stringlength>* 和 *<quantity>* 的含义请参见“PROFINET IO 过程变量 (页 380)”部分。

#### 说明

也可以用数据记录读取和写入各个位。

也可以在数据记录中读取和写入各个位组成的数组，而无需考虑起始地址和数组长度。

#### 说明

通过指定可选信息类型、地址和数量，可以定义对数据记录子区域的结构化访问。

带有不同控制器名称、不同地址、不同数据记录索引或不同长度的输入或输出数据记录的变量拥有独立的存储区。

为独立存储区创建数据项时，将分配输出数据缓冲区并将其初始化为零。对 ADS 数据项执行的写入操作将写入内部写入缓冲区，然后进行传送。

OPC 读取缓存是一个独立的缓冲区，此缓冲区只能通过设备读取作业进行更新。

数据记录非周期性传送。可以对相同的数据同时执行多个网络作业。

始终传送完整的发送数据记录。

此规则也适用于子元素访问或几个客户端同时写入此数据项的情况。

几个客户端同时写入相同的发送数据记录或发送数据记录的子区域会造成不一致。因此，我们建议始终读取或写入完整的数据记录。

## 数据记录的值

数据类型: *VT\_ARRAY / VT\_UI1*。

只读; 如果指定了数据记录长度, 则也可以将其写入。

包含从设备读取的最后一个数据记录或写入设备的最后一个数据记录。

数据记录的结构和长度是灵活的。可用数据记录的长度暗含在值的数组长度中。

请牢记, 根据被寻址设备和数据记录索引的不同, 可能无法将修改的数据记录读回, 一次也不可以。

因此, 只有当数据记录支持多重读取时才能对数据记录进行轮询 (活动的数据项)。

如果对一个带有读取限制或写入限制的数据记录进行寻址, 则会在 OPC 接口上返回错误 (通信错误或访问错误)。

## PROFINET IO 数据记录的示例

*PNIO:[CTRL 1]IDS16382,DATA61450,22*

控制器索引 1, 诊断地址 16382, 数据记录索引 61450, 长度 22 字节, 读取 IO 设备级的诊断信息。

*PNIO:[CTRL 1]IDS32,DATA32788,100,W8*

控制器索引 1, 逻辑地址 32, 数据记录索引 32788, 长度 100 字节。

读取子插槽的诊断信息。字 8 (如果存在) 包含第一个通道错误的错误类型。

*PNIO:[CTRL 1]IDS10,DATA32768*

控制器索引 1, 模块地址 10, 数据记录索引

32768, 未指定长度, 只能对数据项进行读取操作。读取插槽的诊断信息。

*PNIO:[CTRL 1]IDS16382,DATA61452*

控制器索引 1, 诊断 (基本) 地址 16832, 数据记录索引

61452, 未指定长度, 只能对数据项进行读取操作。

读取供应商特定的、块的、插槽的、通道的和数据的诊断信息。

*PNIO:[CTRL 1]QDS10,DATA32768,22,B0,10*

控制器索引 1, 逻辑地址 10, 数据记录索引 32768, 指定长度 22

个字节, 数据项既可读取也可写入, 前 10 个字节的子区域。插槽的诊断信息。

*PNIO:[CTRL 1]QDS10,DATA32768,22,X0.0,13*

控制器索引 1, 逻辑地址 10, 数据记录索引 32768, 指定长度 22

个字节, 数据项既可读取也可写入, 从位 0 开始的前 13 位的子区域。插槽的诊断信息。

*PNIO:[Ctrl2]IDS11000,DATA2147483648,80,X0.0,1*

由于此索引无法组态且位于取值范围之外, 所以即使达不到良好的质量, 也可将此数据项插入到 OPC 组中。

## 2.12.10 系统特定信息变量的语法

### 语法

```
PNIO: [SYSTEM] &version()
```

### 说明

返回 PROFINET IO OPC 服务器的版本 ID，例如，此处返回字符串  
SIMATIC NET Core Server PNIO V 7.xxxx.yyyy.zzzz Copyright 2012

数据类型： VT\_BSTR

访问权限： 只读

## 2.12.11 PROFINET IO 特定的信息变量

### 语法

```
PNIO: [<controllername>] &<informationparameter>
```

### 说明

#### PNIO

用于访问过程变量的 PROFINET IO 协议。

#### <controllername>

协议特定的控制器名称。在 PC 站组态期间指定控制器名称索引。

#### &<informationparameter>

可能的值如下：

- deviceactivate()
- deviceactivateval()
- devicedeactivate()
- devicedeactivateval()
- mode()
- modeval()

### **&deviceactivate()**

激活设备

该模式只能写入。只有在 PROFINET IO

应用环境的上下文中，才能通过写入如下所示的一个值来设置模式。

必须遵循允许的模式更改。这些模式更改包括：

未激活 → 激活

作为字符串的输入值：

符合该组态的设备的逻辑地址以及输入 I 或输出 Q 的标识符：

**<Q|I><logicaldiagnosticsaddress>**

示例： I16373。

### **&deviceactivateval()**

使用数值激活设备。

数据类型 VT\_ARRAY | VT\_I4，只写。数组长度 2 个元素（IO 标记和逻辑地址）。

作为由 2 个数值组成的数组的输入值：

数组的第一个元素始终是一个用于指定该地址是输入地址还是输出地址的标记。

**0 = 输入地址**

**1 = 输出地址**

第二个元素描述设备的任意逻辑地址。

示例： 值 {0|16375} 对应于 I16375。

---

### 说明

由于激活了设备，协议可能触发“站恢复”中断。

---

### **&devicedeactivate()**

取消激活设备。

该模式只能写入。只有在 PROFINET IO

应用环境的上下文中，才能通过写入如下所示的一个值来设置模式。

必须遵循允许的模式更改。

这些模式更改包括：

激活 → 未激活

作为字符串的输入值：

符合该组态的设备的逻辑地址以及输入 I 或输出 Q 的标识符：

**<Q|I><logicaldiagnosticsaddress>**

示例： I16373。

也可选择设备的诊断地址。

**&devicedeactivateval()**

使用数值取消激活设备。

数据类型： VT\_ARRAY | VT\_I4，只写。

数组长度： 2 个元素（IO 标记和逻辑地址）。

数组的第一个元素始终是用于指定该地址是输入地址还是输出地址的标记。

**0 = 输入地址**

**1 = 输出地址**

第二个元素描述设备的任意逻辑地址。

示例： 值 {0|16373} 对应于 I16373。

**说明**

由于取消激活设备，协议可能触发“站故障”中断。

**&mode()**

查询或设置控制器模式。

写入新模式时，允许所有模式更改。

OFFLINE ↔ CLEAR ↔ OPERATE

或

OFFLINE ↔ OPERATE

例如，这意味着可以直接从 OFFLINE 模式更改为 OPERATE 模式，反之亦然。

例如，OFFLINE 模式可以设置为停止控制器。不会对 CLEAR

模式进行自动更改，但是如果使用 NCM 组态了“将 PNIO 控制器自动设置为 OPERATE 模式”，则会在第一个客户端连接至 OPC 服务器时于启动期间对 OPERATE 模式进行自动更改。

在退出 OPC 服务器后，OFFLINE 状态被自动设置。

数据类型 VT\_BSTR，可写，可读。

- OFFLINE
- CLEAR
- OPERATE

**&modeval()**

查询控制器模式或设置为值。

数据类型: VT\_UI4, 可写, 可读

可能值	含义
0	OFFLINE
1	CLEAR
2	OPERATE

## 2.13 PROFINET IO 通过工业以太网与 OPC UA 进行通信

分配给 PROFINET IO 控制器的 PROFINET IO 设备的系统模型为面向模块的模型。

一台 PROFINET IO 设备可包含若干个模块，每个模块可包含若干个子模块。

一个子模块通常包含待连接设备硬件的物理终端或驱动程序块（通道），例如传送带或传感器。将采用 IO 数据对它们的输入数据或设定值进行寻址。读取和写入 IO 数据是 PROFINET IO 最重要的使用案例。

设备、模块和子模块均可提供数据记录并生成报警。

数据记录和报警的使用与设备相关。

SIMATIC NET 的 PROFINET IO OPC UA 服务器有以下特性：

- 过程变量 - 读取/写入/监视数据
- 信息变量 - 状态查询
- 控制变量 - 启用/禁用设备

### 2.13.1 PROFINET IO 协议的 SIMATIC NET OPC UA 服务器

#### 简介

以下部分介绍了 OPC UA 同样支持的 PROFINET IO 协议的组态变型。为此，将 PROFINET IO COM OPC 数据访问服务器设置为进程外 OPC 服务器。

## 组态

在“通信设置”(Communication Settings) 组态程序的“OPC 协议选择”(OPC protocol selection) 目录中选择“PROFINET IO”和“OPC UA”可激活 PROFINET IO UA 服务器：



图 2-65 “通信设置”组态程序中用于为 PROFINET IO 协议选择 OPC UA 的窗口

## 优点/缺点

使用 PROFINET IO OPC UA 服务器时，只有 PROFINET IO 的进程外模式可用。

必须运行 PROFINET IO OPC UA 服务器进程以维持准备接收的状态。即使所有 OPC UA 客户端都已注销，PROFINET IO OPC UA 服务器也不会更不能退出。

但另一方面，则具有以下优点：

- 不再需要 COM/DCOM 组态。
- 高速、安全的通信
- 事件和数据访问仅需要一个服务器。
- 多个客户端可同时使用服务器。

## 2.13 PROFINET IO 通过工业以太网与 OPC UA 进行通信

如果选中“OPC UA”和“PROFINET IO”复选框，则会始终启动 PROFINET IO UA 服务器。如果使用适合的 PROFINET IO 组态，则表示 PROFINET IO 控制器处于 OFFLINE 模式。仅当至少一个 OPC UA 客户端已连接到服务器时，PROFINET IO 控制器才会更改为 OPERATE 模式。

---

### 说明

服务器的组态循环时间并不是自动采用调用下级 PROFINET IO 协议接口的时间。根据 OPC 客户端更新时间的不同，调用率也可以变慢。

---

### 说明

如果 PROFINET IO 协议的 OPC UA 处于激活状态，则当首个 OPC UA 客户端登录到 OPC UA PROFINET IO 服务器时，OPC 服务器的已组态 PROFINET IO 控制器将立即切换为 OPERATE 模式。如果未连接任何 OPC UA 客户端（例如，计算机启动后），则 OPC 服务器的已组态 PROFINET IO 控制器仍将保持 OFFLINE 模式。

---

## 2.13.2 如何对 PROFINET IO OPC UA 服务器进行寻址？

### 服务器 URL

对于本地二进制 TCP 协议，OPC 客户端有两种对服务器进行寻址的方法：

- 直接寻址：

- opc.tcp://<hostname>:55104

或

- opc.tcp://<IP-Adresse>:55104

或

- opc.tcp://localhost:55104

PNIO OPC UA 服务器使用端口 55104。

- 也可以使用 OPC UA 发现服务查找 PROFINET IO OPC UA 服务器的 URL。

用于定位发现服务器的条目如下所示：

- opc.tcp://<hostname>:4840

或

- opc.tcp://<IP-Adresse>:4840

或

- http://<hostname>:52601/UADiscovery/

或

- http://<IP-Adresse>:52601/UADiscovery/

发现服务器使用端口 4840（用于 TCP 连接）和端口 52601（用于 HTTP 连接）。

### IPv6 地址

IPv6 地址也可以用作 IP 地址。地址必须位于括号中，例如 [fe80:e499:b710:5975:73d8:14]。

### 端点和安全模式

SIMATIC NET PROFINET IO OPC UA 服务器支持本地二进制 TCP 协议的端点，并允许使用证书进行验证以及进行加密传送。

## 2.13 PROFINET IO 通过工业以太网与 OPC UA 进行通信

已寻址主机上的“发现”服务将用信号通知服务器的端点，换句话说就是服务器的安全要求和协议支持。

PROFINET IO OPC UA 服务器的服务器 URL "opc.tcp://<hostname>:55104"

可提供以下端点：

- 在“签名和加密”安全模式下的端点：

必须进行签名和加密才能与服务器进行通信。通过交换证书和输入密码来保护通信。

除了安全模式，还显示了安全策略 **Basic128Rsa15**。

- 在“无”安全模式下的端点：

在此模式下，服务器不需要使用安全功能（安全策略“无”）。

有关安全功能的详细信息，请参见“对 OPC UA 接口进行编程 (页 586)”部分。

安全策略“**Basic128Rsa18**”和“无”在 OPC 基金会的 UA 规范中，其 Internet 地址如下所示：

"<http://opcfoundation.org/UA>" > "Specifications" > "Part 7"

有关更多的详细信息，请参见以下 Internet 页面：

OPC 基金会 (<http://www.opcfoundation.org/profilereporting/index.htm>)

> "Security Category" > "Facets" > "Security Policy"

### OPC Scout V10 的 OPC UA 发现

在 OPC Scout V10 中，您可以打开“OPC UA 发现”(OPC UA Discovery) 对话框并在 OPC Scout V10 的导航区域中输入 UA 端点。

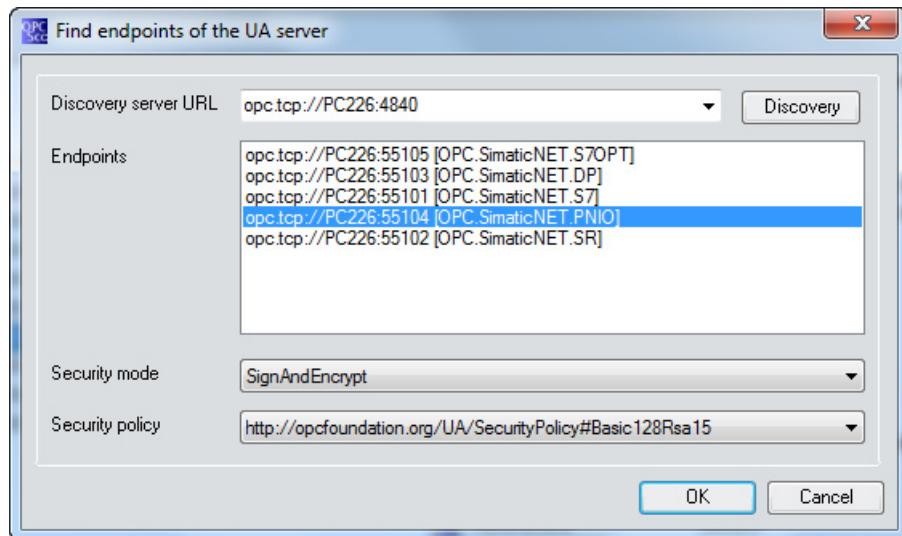


图 2-66 OPC Scout V10 的“查找 UA 服务器端点”对话框

可使用 OPC UA 发现服务查找 PROFINET IO OPC UA 服务器。

相关条目，请参见上述“服务器 URL”。

OPC Scout V10 包含了一系列 OPC UA 端点。这些端点为永久性端点。

然后，被寻址主机上的“发现”服务会用信号通知已注册的 OPC UA 服务器、其端口和安全模式。

更多详细信息，请参见 OPC Scout V10 的在线帮助。

### 2.13.3 PROFINET IO 与 OPC UA 之间的通信提供哪些命名空间？

PROFINET IO OPC UA 服务器提供了以下命名空间：

表格 2- 8 OPC UA 的命名空间

命名空间索引	“标识符”（命名空间 URI）/注释
0	“ <a href="http://opcfoundation.org/UA/">http://opcfoundation.org/UA/</a> ” (由 OPC 基金会指定)
1	“urn:Siemens.Automation.SimaticNET.PNIO:(GUID)” 本地高速 PROFINET IO OPC UA 服务器的唯一标识符。
2	“PNIOTYPES:” PROFINET IO 类型的命名空间的标识符。
3	“PNIO:” 本地高速 PROFINET IO OPC UA 服务器的标识符，使用新的简化语法（可浏览并可与 UA 一起使用）
4	“PNIOCOM:” 使用旧语法的服务器的标识符。 (可与 UA 一起使用但不能浏览) 在此命名空间中，与 PROFINET IO OPC DA 兼容的语法可用于 OPC UA 服务器，但此处不可使用与 OPC DA 服务器上的别名项目或符号相对应的任何标识符。 示例：在 PNIOCOM 命名空间中：存在标识符 PNIO:[cltr1]ab0，而该标识符同样也是 OPC DA 服务器上的项目。

根据组态，命名空间索引与命名空间标识符之间的分配可以在 OPC UA  
客户端上针对索引 2 至 4 进行更改。因此，标识符必须始终唯一。

命名空间索引 0 和 1 保留，其意义由 OPC 基金会指定。

将剩余命名空间索引分配至标识符（命名空间  
URI），这一操作必须由指定该标识符的客户端通过 OPC UA  
会话开始处的数据变量“NamespaceArray”来完成。 PNIOTYPES 标识符： PNIO:  
PNIOCOM: 始终与 PROFINET IO OPC UA 服务器一起存在。

## 2.13 PROFINET IO 通过工业以太网与 OPC UA 进行通信

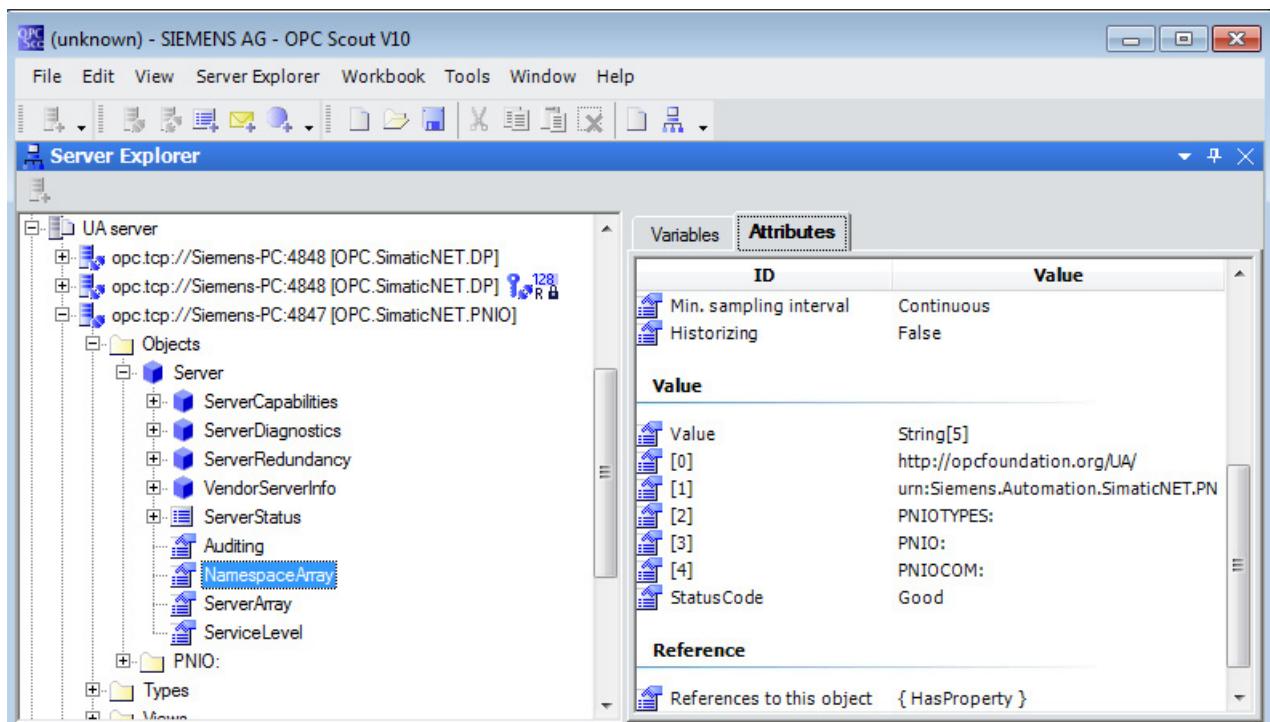


图 2-67 使用 OPC Scout V10 的浏览功能显示 PROFINET IO 的命名空间

## 2.13 PROFINET IO 通过工业以太网与 OPC UA 进行通信

### 如何浏览已组态的 PROFINET IO 命名空间？

OPC 服务器的已组态 PROFINET IO 设备可通过“OPC 数据访问”的浏览功能进行显示。

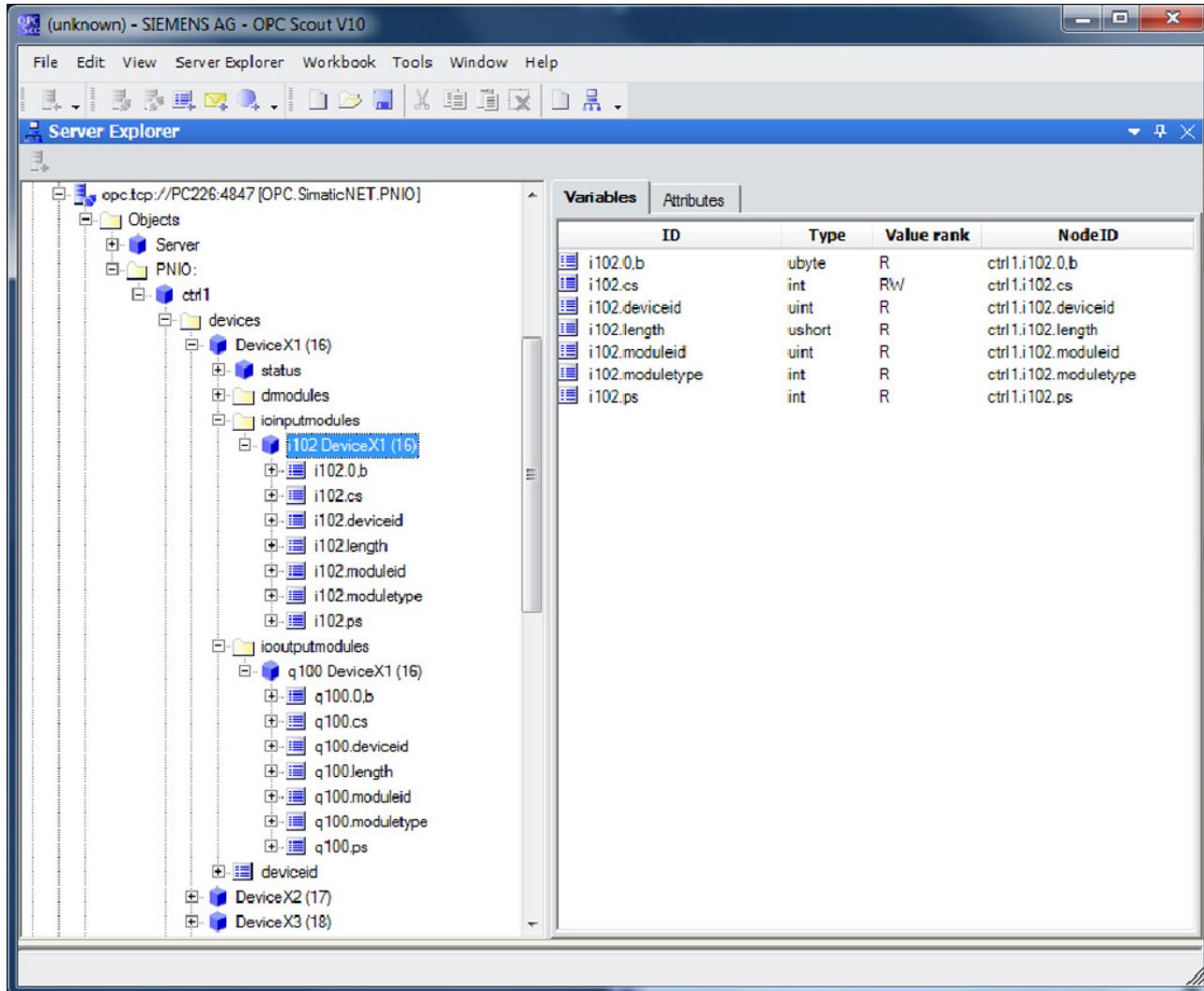


图 2-68 通过 OPC Scout 浏览已组态的命名空间

显示现有输入和输出以及它们的地址。由于组态数据长度已知，因此预定义变量有效。

在工厂视图中，所有组态设备及其名称（子模块在其下方）均在“设备”分支中列出。

显示每个模块现有的所有输入和输出以及它们的地址。预定义数据项已存在。

## 2.13.4 NodId

### PROFINET IO 过程变量的标识

NodId 借助下列元组标识 PROFINET IO 过程变量：

- 命名空间索引
- 标识符（字符串、数字值）

### 示例

- NodId:

- 命名空间 URI:

*PNIO:*

(= 命名空间索引 3)，适用于 Siemens.Automation.SimaticNET.PNIO

- 标识符

*ctrl1.q0.0,b*

- NodId:

- 命名空间 URI:

*PNIOCOM:*

(= 命名空间索引 4)

- 标识符

*PNIO:[ctrl1]AB0*

### 新的命名空间是如何适应 OPC UA 运作的？

尽管“OPC

数据访问”项往往被限制为读取和写入过程变量，而且非常独立并与警报完全隔离，但是 OPC UA 视图也可侧重于具有不同属性的自动化对象。

因此，OPC UA 不会再单独访问各项，而且也不会再单独访问对象及其子对象。

例如，数据变量和方法是 PROFINET IO 设备对象的子对象。

属性更详细地定义对象，所有信息均由对象的属性加以提供。

用于 PROFINET IO 设备的 IO 区域访问的“OPC 数据访问”项与 OPC UA 数据变量相对应。用于激活/取消激活设备的“OPC 数据访问”项与 OPC UA 方法相对应。

合格的标识符 (NodId) 在 OPC UA 中比在“OPC 数据访问”中更有意义。每次单独访问对象、子对象、特性及属性均使用其 NodId。

## 2.13 PROFINET IO 通过工业以太网与 OPC UA 进行通信

系统会将 PROFINET IO 的 OPC 命名空间表示为树状结构，并可浏览其 NodId。命名空间的所有文本均采用英语。

### 语法

所有 OPC UA 对象的 NodId 均具有以下结构：

```
<controllerobject>"."<object>"."<subobject>"."<property>
```

一个对象可包含其它子对象。

无法解析的 NodId 会被拒绝，并会显示错误消息。任何对象中的字母“A-Z”都不区分大小写。

### 符号对象表示法

OPC UA 规范建议对地址空间的层级描述使用统一的符号表示法。

本文档中所使用的符号如下所示：

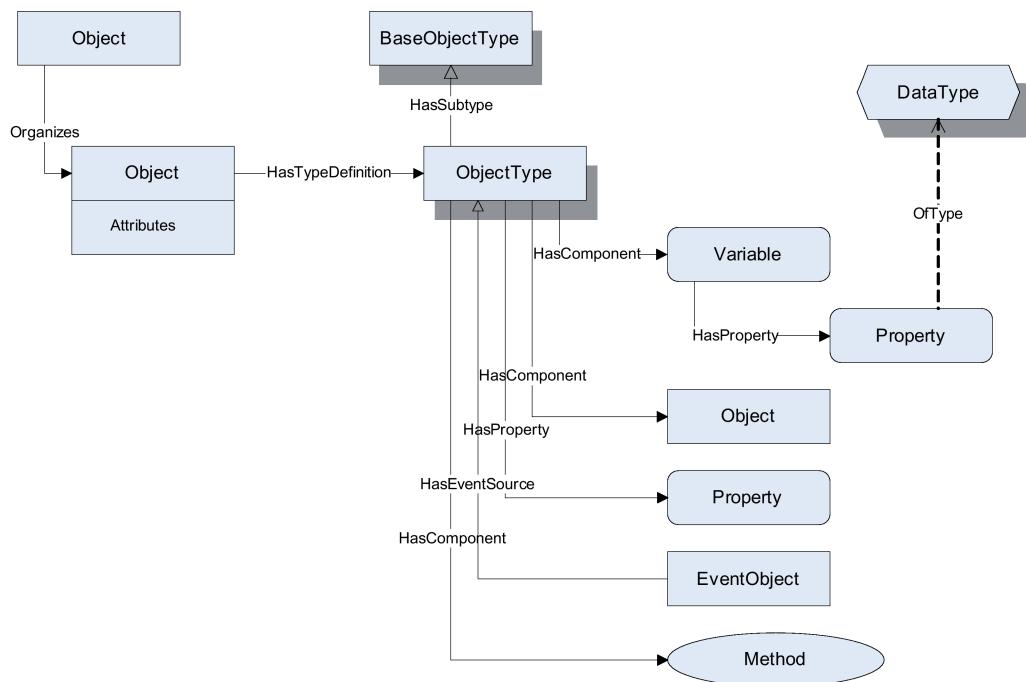


图 2-69 OPC UA 地址空间的符号

## 2.13.5 板对象和 PROFINET IO 控制器

### PROFINET IO 控制器的板名称

始终将所有协议特定的对象分配给板，在 PROFINET IO 中，其是 PROFINET IO 控制器。板名称是在 STEP 7 或 TIA 门户中组态的用于标识模块的 PNIO 控制器。

#### 语法

```
ctrl<slot>
```

#### 说明

**ctrl**

PNIO 控制器的标识符

**<slot>**

“站组态编辑器”中的插槽

**示例：**

```
ctrl112
```

标识“站组态编辑器”的插槽 12 中的 PNIO 控制器

## 2.13 PROFINET IO 通过工业以太网与 OPC UA 进行通信

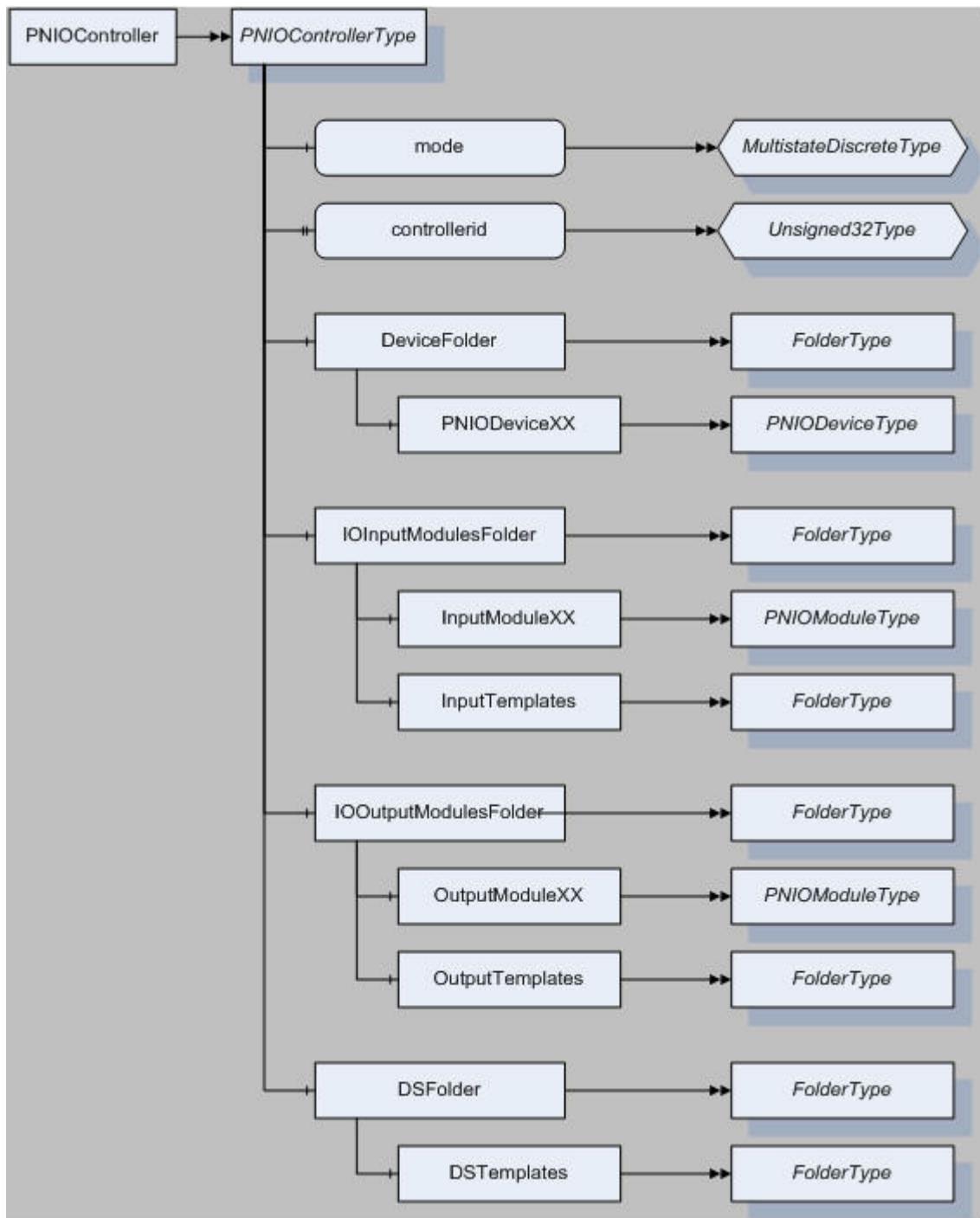


图 2-70 PROFINET IO 控制器类别树

## 2.13.6 PROFINET IO 控制器的数据变量

### PROFINET IO 控制器数据变量的语法

<controller>.<datavariable>

#### 说明

##### **<controller>**

协议特定的连接名称。连接名称在组态中指定。在 PNIO 协议中，连接名称是通信模块的已组态名称。

##### **<datavariable>**

数据变量	说明						
mode	<p>查询或设置控制器的模式。</p> <p>UA 类型“MultistateDiscreteType”的数据变量，读写。</p> <table border="1" style="margin-left: 20px;"> <tr> <td>0</td><td>OFFLINE</td></tr> <tr> <td>1</td><td>CLEAR</td></tr> <tr> <td>2</td><td>OPERATE</td></tr> </table> <p>写入一个新的操作状态时，必须记住允许的操作状态更改。这些模式更改包括：  <b>OFFLINE ↔ CLEAR ↔ OPERATE</b></p> <p>如果出错，则返回通信错误。</p> <p>CLEAR 状态通常仅由 DP 从站进行实施，而该从站被通过 IE-PB 链接（以太网和 PROFIBUS 之间的链接）仿真为 PROFINET IO 设备。</p> <p>根据组态值 AutoOnline，在客户端第一次登录时，控制器由 OPC 服务器自动更改为 OPERATE 模式。当最后一个客户端注销时，控制器自动更改为 OFFLINE 模式，而不管组态值 AutoOnline 为何。如果一个客户端崩溃，则其将会被 UA 协议栈根据客户端与服务器之间协商的保持连接机制检测到；在所选的等待时间过后，该客户端将自动从协议栈注销。</p> <p>在 OPC 服务器关闭后，控制器将会关闭。</p>	0	OFFLINE	1	CLEAR	2	OPERATE
0	OFFLINE						
1	CLEAR						
2	OPERATE						
controllerid	<p>“站组态编辑器”中的插槽</p> <p>“UInt32”类型的属性，只读</p>						

**示例：**

```
ctrl14.mode  
ctrl14.controllerid
```

## 2.13.7 设备对象

PROFINET IO 控制器的通信伙伴为具有输入和输出子模块的 PROFINET IO 设备。

### 语法

```
ctrl<slot>.dev<devicenumber>
```

### 说明

**ctrl**

PNIO 控制器的标识符

**<slot>**

“站组态编辑器”中的插槽

**dev**

PNIO 设备的标识符

**<devicenumber>**

根据 STEP 7 或 TIA Portal 组态的设备编号

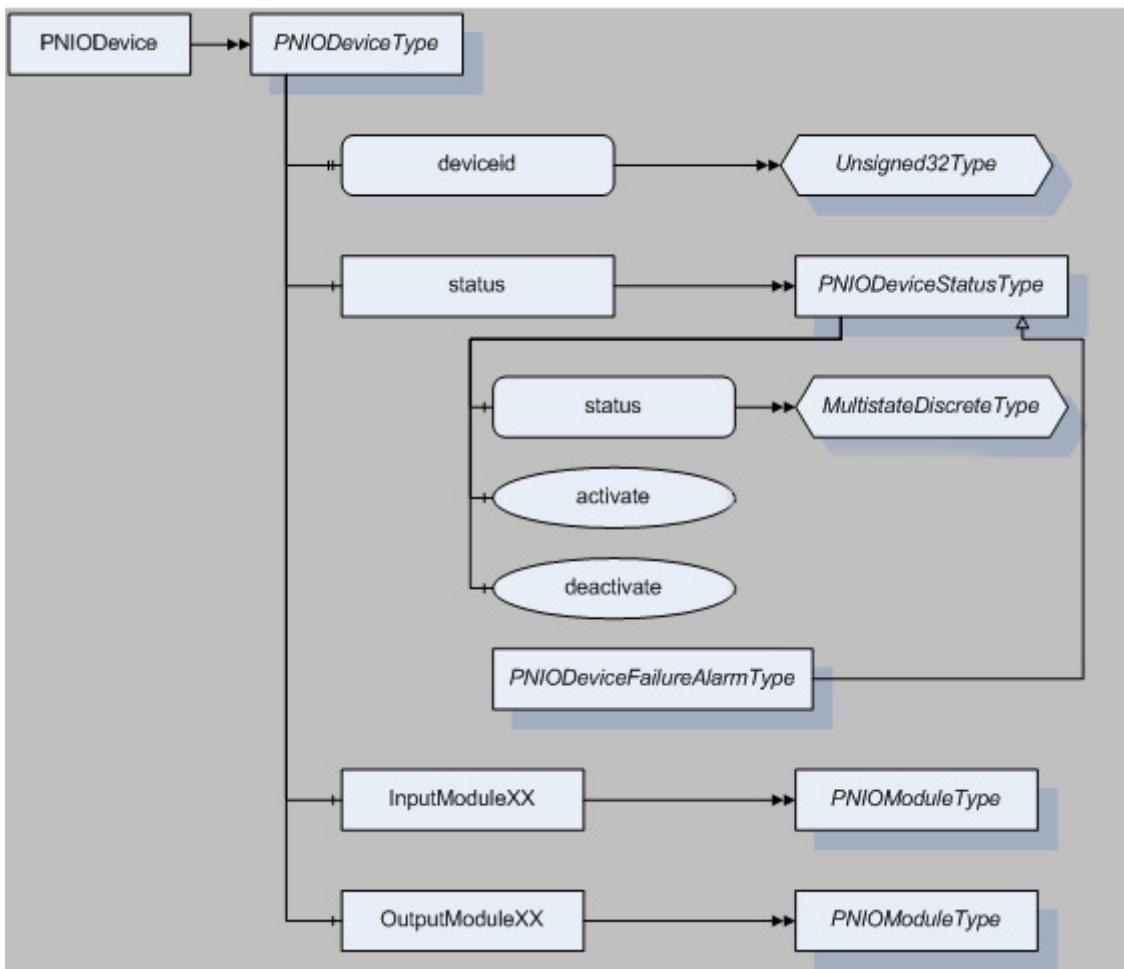


图 2-71 设备对象

PROFINET IO 控制器类别树的设备对象显示在图 2-48

中。在寻址的框架中，PROFINET IO

设备的意义较小且仅在连接建立期间由协议栈进行使用。在 OPC 服务器-客户端接口上，PROFINET IO 设备仅通过其模块进行寻址。

即使一条命令（例如，设备激活/取消激活）影响 PNIO

设备的所有模块，该命令仅需要对 PNIO 设备的任意一个模块进行寻址。

从组态中，OPC 服务器获得已分配给该 PNIO 设备的 I/O 和诊断地址。它们唯一且用于 OPC UA Nodeld 和通信。

然而，设备对象的所有数据变量均接收已在 STEP 7 内名称设备部分的附加 UA 属性“DisplayName”中组态的设备名称。

PROFINET IO 设备对象负责提供与 PNIO 设备有关的全局状态信息。PNIO

设备状态将在下级对象中进行管理，而且还具有两种方法来激活和取消激活带全部模块的 PNIO 设备。

## 2.13.8 设备对象的数据变量和方法

### 设备对象的数据变量和方法的语法

存在四种可能性：

- `ctrl<slot>.dev<devicenumber>.<datavariable>`
- `ctrl<slot>.dev<devicenumber>.<method>`
- `ctrl<slot>.dev<devicenumber>.status.<datavariable>`
- `ctrl<slot>.dev<devicenumber>.status.<method>`

### 说明

#### **ctrl**

PNIO 控制器的标识符

#### **<slot>**

“站组态编辑器”中的插槽

#### **dev**

PNIO 设备的标识符

#### **<devicenumber>**

根据 STEP 7 或 TIA Portal 组态的 PNIO 设备编号

#### **status**

PNIO 设备的状态对象

#### **<datavariable> / <method>**

数据变量/方法	说明	
deviceid	STEP 7 或 TIA 门户上的 PNIO 设备编号 “UInt32”类型的属性，只读	
status	PNIO 设备的状态 UA 类型“MultistateDiscreteType”的数据变量，只读	
	0	FAILURE
	1	OPERATE

数据变量/方法	说明
activate()	<p>激活 PNIO 设备</p> <p>“activate()”OPC UA 方法只能写入。它没有任何参数。</p> <p>如果成功完成方法调用，则表示激活已成功传输到 PNIO 设备。从 PNIO 控制器的角度来看，无论 PNIO 设备是否真的处于激活状态，都必须使用“status”数据变量进行检查。</p>
deactivate()	<p>取消激活 PNIO 设备</p> <p>“deactivate()”OPC UA 方法只能写入。它没有任何参数。</p> <p>如果成功完成方法调用，则表示取消激活已成功传输到 PNIO 设备。从 PNIO 控制器的角度来看，无论 PNIO 设备是否真的处于取消激活状态，都必须使用“status”数据变量进行检查。</p>

### 示例：

```

ctrl4.dev32.deviceid

ctrl4.dev32.status.status

ctrl4.dev32.status.activate()

ctrl4.dev32.status.deactivate()

```

## 2.13.9 PROFINET IO 模块对象

### 常规

实际的 PNIO 通信通过 IO 子模块进行。一个 IO 子模块始终被分配给一台 PNIO 设备。OPC 服务器接收来自 STEP 7 或 TIA 门户组态的 I/O 子模块的类型和长度。

### PNIO 子模块对象的语法

有两个选项：

- `ctrl<slot>.i<address>`
- `ctrl<slot>.q<address>`

## 说明

### ctrl

PNIO 控制器的标识符。

### <slot>

“站组态编辑器”中的插槽。

### q

输出的标识符。 可读取和写入输出。

### i

输入的标识符。 输入为只读。

### <address>

IO 子模块的逻辑地址。

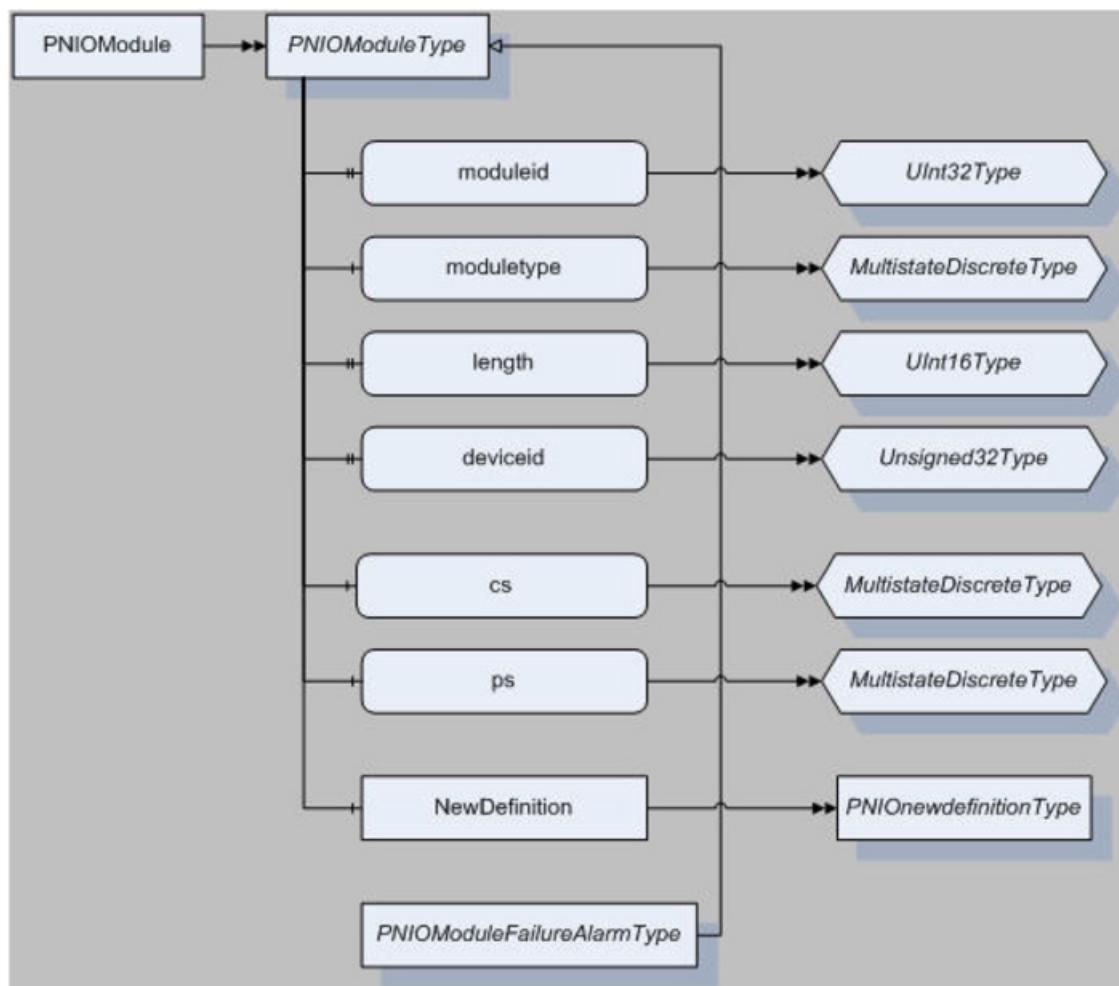


图 2-72 PROFINET IO 模块对象

### 2.13.9.1 PROFINET IO 模块的数据变量

#### PNIO 模块对象数据变量的语法

有两个选项：

- `ctrl<slot>.i<address>.<IOmodulevariable>`
- `ctrl<slot>.q<address>.<IOmodulevariable>`

#### 说明

##### **ctrl**

PNIO 控制器的标识符。

##### **<slot>**

“站组态编辑器”中的插槽。

##### **q**

IO 输出子模块的标识符。可读取和写入输出。

##### **i**

IO 输入子模块的标识符。输入为只读。

##### **<address>**

IO 子模块的逻辑地址。

##### **<IOmodulevariable>**

moduleId	STEP 7 或 TIA 门户的子模块地址 “UInt32”类型的属性，只读
模块类型	STEP 7 或 TIA 门户的 IO 子模块类型 “MultistateDiscreteType”类型的数据变量，只读
0	未知
1	I (Input) 当前由 PROFINET IO UA 服务器传送的值为 I。
2	Q (Output) 当前由 PROFINET IO UA 服务器传送的值为 O。
3	I (Diag) 当前由 PROFINET IO UA 服务器传送的值为 I。 没有循环数据，只有数据记录。

## 2.13 PROFINET IO 通过工业以太网与 OPC UA 进行通信

	用于信息目的的变量
length	STEP 7 或 TIA 门户中以字节表示的子模块长度 “UInt16”类型的属性，只读 特殊字节长度：
	0 I/O 地址范围内无模块 (modulename = 3)
deviceid	属于 STEP 7 或 TIA 门户 IO 子模块的 PNIO 设备编号 “UInt32”类型的属性，只读

### 示例：

```
ctrl14.q27.moduleid  
ctrl14.q28.modulename  
ctrl14.q28.length  
ctrl14.q28.deviceid
```

## 2.13.9.2 PROFINET IO 模块的 IO 数据变量

### 常规

读取和写入循环 IO 数据的 **NodeID** 是最常见的应用。这将允许对 PNIO 地址空间进行简单访问。用户必须指定 PNIO 控制器名称和项目工程组态中 IO 子模块的逻辑地址。

### PNIO 模块对象数据变量的语法

有两个选项：

- `ctrl<slot>.i<address>.<offset>{,<PNIOtype>{,<quantity>}}`
- `ctrl<slot>.q<address>.<offset>{,<PNIOtype>{,<quantity>}}`

### 说明

#### ctrl

PNIO 控制器的标识符。

**<slot>**

“站组态编辑器”中的插槽。

**q**

IO 输出子模块的标识符。可读取和写入输出。

**i**

IO 输入子模块的标识符。输入为只读。

**<address>**

IO 子模块的逻辑地址。

**<offset>**

要寻址的 IO 子模块在数据区域中的字节地址。

**<PNIObjectType>**

数据类型	OPC UA 数据类型	注释
x<bitaddress>	布尔	位（布尔） 除区域内的字节偏移量外，还必须在字节中指定 <bitaddress>。 值范围是 0 到 7
b	字节 字节字符串	字节（无符号） 如果未指定 <PNIObjectType>，则用作默认值。
w	UInt16	字（无符号）
dw	UInt32	双字（无符号）
lw	UInt64	长字（无符号）
c	SByte	字节（有符号）
i	Int16	字（有符号）
di	Int32	双字（有符号）
li	Int64	长字（有符号）
r	浮点	浮点（4 字节）
lr	双精度	浮点（8 字节）
s<stringlength>	字符串	还必须指定为字符串保留的 <stringlength>。 写入时还可写入较短的字符串，使传送的数据长度始终为保留的字符串长度（字节）。不必要的字节用值 0 填充。

### <quantity>

元素的数量。 变量的数据类型是具有指定格式元素的数组。

指定多个数组元素将导致形成相应类型的数组，即使只对一个数组元素进行寻址也是如此。

#### 示例：

ctrl4.i10.0,w

始于字节地址 10 的输入字

ctrl4.q17.3

始于字节地址 3 的输出字节（默认为“字节”）

ctrl4.i10.0,x0

始于字节地址 0 的输入位，位 0

ctrl4.q17.1,x4,16

始于字节地址 1 的由 16 个输出位组成的数组，位 4

ctrl4.i27355.100,s32,3 具有 3 个字符串的数组（每个字符串有 32 个字符）

## 2.13.9.3 IOPS 和 IOCS 的数据变量

### 常规

使用状态（IOPS 和 IOCS）的项目，OPC 客户端可以查询或设置逻辑地址的状态字节。

下图显示了过程映像中的数据状态和值流。

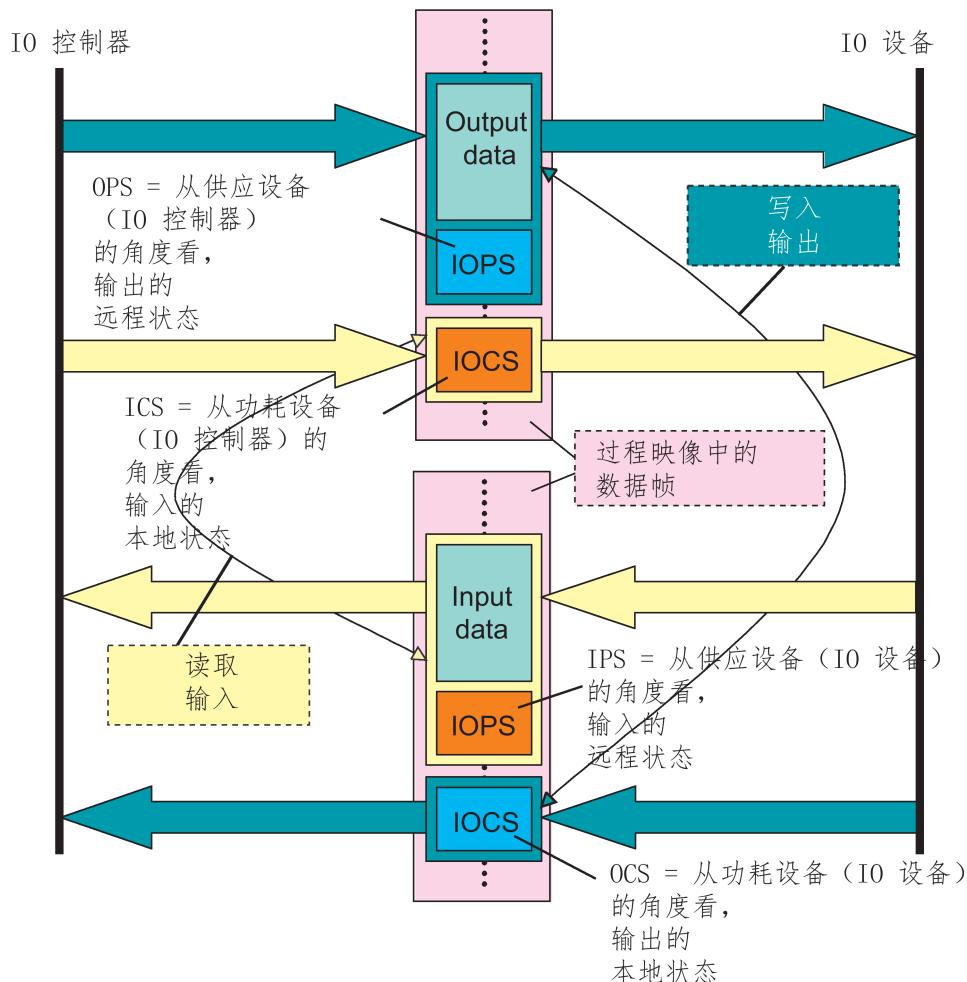


图 2-73 IO 控制器与 IO 设备之间的 PROFINET IO 通信中过程映像的数据状态概述。

从具有“OPS”的提供者角度来看，当 OPC

客户端将输出数据写出到过程映像时，其中包括远程输出状态。

从具有“OCS”的用户角度来看，设备包括其自己的本地输出状态。

从具有“ICS”的用户角度来看，当 OPC 客户端从过程映像读取输入数据时，其中包括本地输入状态。从具有“IPS”的提供者角度来看，设备包括远程输入状态。

## 说明

### 对循环输出数据的读访问

读取、监控或激活用于映射循环输出数据的 OPC 数据项/节点将引起底层 PROFINET IO 协议接口的一次更新或不断更新，其中，数据包含在包括提供者状态 (PNIO\_data\_write()) 的 OPC 服务器缓存中。请记住：在 PROFINET IO 设备或模块返回后，该输出数据将在更新的时间点之后再次传送到设备中。

## IOPS 和 IOCS 的数据变量的语法

有四个选项可用：

IPS 变量：

- `ctrl<slot>.i<address>.ps`

OPS 变量

- `ctrl<slot>.q<address>.ps`

ICS 变量：

- `ctrl<slot>.i<address>.cs`

OCS 变量：

- `ctrl<slot>.q<address>.cs`

## 说明

**ctrl**

PNIO 控制器的标识符。

**<slot>**

“站组态编辑器”中的插槽。

**q**

IO 输出子模块的标识符。可读取和写入输出。

**i**

IO 输入子模块的标识符。输入为只读。

**<address>**

IO 子模块的逻辑地址。

**ps**

IO 子模块的提供者状态。

**Cs**

IO 子模块的用户状态。

名称	说明
IPSVariable	<p>输入地址的远程状态</p> <p>UA 类型“MultistateDiscreteType”的数据变量，只读</p> <p>输入的提供者为设备。设备还提供输入数据的状态。</p> <p>IPS 由轮询进行监控。</p> <p>OPC 应用程序并不绝对需要使用项，因为项内容是在具有相同地址的链接输入项的 OPC 质量中进行反映。</p>
ICSVariable	<p>输入地址的本地状态</p> <p>UA 类型“MultistateDiscreteType”的数据变量，可写且可读</p> <p>默认值： <i>GOOD</i> = 0</p> <p>输入的用户为 OPC 客户端。从自身角度来看，OPC 客户端也会将输入数据的状态告知设备。通常，该状态始终为 <i>GOOD</i>。</p> <p>读取 IO 数据时，始终包括由 OPC 应用程序写入的输入地址的最后本地状态（已写入到伙伴设备）。如果 OPC 应用程序未创建项或未将一个值写入到其中，则在读取相应输入数据时，会传送值 <i>GOOD</i>。</p> <p>状态由 OPC 客户端通过缓存进行读取。</p>

名称	说明
OCSVariable	<p>输出地址的本地状态</p> <p>UA 类型“MultistateDiscreteType”的数据变量，只读</p> <p>输出的用户为设备。从自身角度来看，设备也包括输出数据的状态。</p> <p>如果 OPC 客户端首先（或同时）写入具有相同地址的链接输出项，则可读取状态。</p> <p>OCS 由轮询进行监控，同时由 OPC 客户端写入的最后输出数据始终被传送到函数中。</p> <p>如果 OPC 客户端无法成功写入一个值，则从设备进行读取将导致错误或使 OPC 质量变为不良。</p>
OPSVariable	<p>输出地址的远程状态</p> <p>UA 类型“MultistateDiscreteType”的数据变量，可写且可读</p> <p>默认值： <i>GOOD</i> = 0</p> <p>输出的提供者为 OPC 客户端。从自身角度来看，OPC 客户端也会将输出数据的状态告知设备。通常，该状态始终为 <i>GOOD</i>。</p> <p>写入 IO 数据时，也包括由 OPC 应用程序写入的输出数据的最后本地状态和输出地址的输出数据（可写）。如果 OPC 应用程序尚未创建 IO 项或尚未成功写入一个输出值，则写入相应输出数据时，将传送值 <i>GOOD</i>。</p> <p>当 OPC 客户端写入一个新状态时，使用先前（或同时）成功写入的数据值，而该数据值具有链接输出项的地址。</p> <p>如果 OPC 客户端尚未成功写入一个数据值，则写入项将产生特殊返回值 <i>OPCUA_GoodClamped</i>。</p> <p>然后，写入状态便可由具有下一成功写入的相应输出项进行写入。</p>

### 状态值的结构

UA 类型“MultistateDiscreteType”的数据变量始终具有以下结构：

0 = GOOD

1 = BAD

正常情况下，ctrl.xxx.cs 和 ctrl.xxx.ps 不需要被写入 UA

接口，因为这些值由写入/读取变量进行设置。

在设备需要被告知缺乏可用性和处理可能性时，状态才会被明确设置。

### 示例：

ctrl1.i10.ps	逻辑输入地址 10 的远程状态
ctrl1.q20.cs	逻辑输出地址 20 的本地状态

## 2.13.9.4 PROFINET IO 模块的数据记录数据变量

### 常规

使用数据记录数据变量（“数据记录”=“dr”），可以读出或写入设备特定的信息。可能的地址被链接到组态模块或诊断地址。

### 数据记录和长度信息

语法可区分具有长度信息和没有长度信息的数据记录。

写入数据记录时，始终需要长度信息，此时，变量不以字节字符串的形式对整个数据记录进行寻址。然后，要写入的数据长度已知。使用子元素寻址时，也会如此。

如果在没有长度信息以及未使用子元素寻址时进行写入，则 OPC UA 服务器会基于要写入的字节数计算数据记录的长度。

没有长度信息时，读取和写入数据记录被限制为 480 个字节。

### 子元素寻址

用户数据记录的构造由子元素寻址在客户端进行提供。

在此，数据记录的子区域将被寻址。

仅可读数据变量：可在没有数据记录的长度信息的情况下，对子元素进行寻址。

由于数据记录的设备特定长度事先未知或可以更改，因此甚至可以使用实际可接收长度之外的子元素。然而，OPC UA 状态代码会在随后显示错误。

可读和可写数据变量：

在整个数据记录上写入单个子元素或同时写入多个子元素在通过长度信息指定的范围内是一致的。

如果之前从未写入该数据记录，则任何间隙都可用最后的写入值（未读取值）或“0”值进行填充。

## PROFINET IO 子模块的数据记录数据变量的语法

存在四种可能性：

- `ctrl<slot>.i<address>.dr<number>{.<DRlength>,<offset>, <PNIObjectType>{,<quantity>}}`
- `ctrl<slot>.q<address>.dr<number>{.<DRlength>,<offset>, <PNIObjectType>{,<quantity>}}`
- `ctrl<slot>.i<address>.dr<number>.<offset>,<PNIObjectType>{,<quantity>}`
- `ctrl<slot>.q<address>.dr<number>.<Offset>,<PNIObjectType>{,<quantity>}`

## 说明

### **ctrl**

PNIO 控制器的标识符。

### **<slot>**

“站组态编辑器”中的插槽。

### **q**

IO 输出子模块的标识符。可读取和写入输出。

### **i**

IO 输入子模块的标识符。输入为只读。

### **<address>**

IO 子模块的逻辑地址。

### **dr**

数据记录数据变量的标识符。

### **<number>**

IO 子模块的数据记录数。

### **<DRlength>**

IO 子模块的数据记录长度。

### **<offset>**

要寻址的 IO 子模块在数据记录中的字节地址。

### <PNIObjectType>

数据类型	OPC UA 数据类型	注释
x<bitaddress>	布尔	位（布尔） 除区域内的字节偏移量外，还必须在字节中指定 <bitaddress>。 值范围是 0 到 7
b	字节 字节字符串	字节（无符号） 如果未指定 <PNIObjectType>，则用作默认值。
w	UInt16	字（无符号）
dw	UInt32	双字（无符号）
lw	UInt64	长字（无符号）
c	SByte	字节（有符号）
i	Int16	字（有符号）
di	Int32	双字（有符号）
li	Int64	长字（有符号）
r	浮点	浮点（4 字节）
lr	双精度	浮点（8 字节）
s<stringlength>	字符串	还必须指定为字符串保留的 <stringlength>。 写入时还可写入较短的字符串，使传送的数据长度始终为保留的字符串长度（字节）。不必要的字节用值 0 填充。

### <quantity>

元素的数量。 变量的数据类型是具有指定格式元素的数组。

指定多个数组元素将导致形成相应类型的数组，即使只对一个数组元素进行寻址也是如此。

**示例：**

ctrl11.i10.dr61450

具有逻辑输入地址 10

的设备的数据记录诊断信息。整个数据记录，Byte[]  
，可读和可写

ctrl11.q99.dr32788,100.8,w,3

具有数据记录长度的数据记录，子元素寻址起始于字  
节 8，UInt16[3]，可读和可写

ctrl11.q99.dr32788.8,dw

不具有数据记录长度的数据记录，子元素寻址起始于  
字节 8，UInt32，只读**2.13.10 PROFINET IO OPC UA 模板****2.13.10.1 模板数据变量****通过 OPC UA PNIO 协议的 IO**

数据变量和数据记录数据变量，您可拥有灵活的设置选项来以所需的数据格式获取设备的过程数据。

不过，在完全可浏览的命名空间中，不能将各种寻址选项放在一起。即使单字节长度的 IO 子模块也有大约 40 个不同的数据格式选项 - 从 Byte 和 SByte 开始，由这些数据类型的一个元素组成的数组、单个位、最多 8 个数组元素的位数组，每个元素均起始于不同的位偏移。

因此，OPC UA 服务器支持 PNIO 命名空间中具有模板数据变量的用户。在典型的 OPC UA 客户端文本输入框中，只需更改几个字符，就可将这些模板转换为有效的 ItemID。

**示例：**

ctrl4.i&lt;x&gt;.&lt;o&gt;,dw

用于输入模块的 UInt32 的模板

通过将 <x> 替换为模块地址以及将 <o> 替换为模块的偏移量，可获得有效的 Nodeld。→ ctrl4.i8.15,dw

**另一个示例：**

ctrl4.i&lt;x&gt;.dr&lt;dr&gt;,&lt;l&gt;.&lt;o&gt;,c,&lt;c&gt;

数据记录变量的模板，Char[]

→ ctrl4.i125.dr61450,100.0,c,100

此概念的一大优点是：几乎所有 OPC UA 客户端均可使用它，而无需对客户端进行任何调整。

另一个优点是：仅需做出很小的努力即可实现，并且该概念也可用于包括 SIMATIC NET OPC 环境的 COM 的其它协议服务器。

### 说明

可在“通信设置”组态程序的“OPC 协议选择”(OPC protocol selection) 中启用和禁用 OPC UA PNIO 模板数据变量的可用性。

## 树状结构中的模板数据变量

在命名空间表示中相应的文件夹旁将模板数据变量进行排序以便需要时便于使用。

### 2.13.10.2 模板数据变量的目录

#### 模板数据变量部分属性的特殊用法

在模板变量中，“`NodeId`”、“`BrowseName`”和“`Description`”属性的使用极其智能。

#### 模板数据变量的语法

有两个选项：

- `ctrl<slot>.i<x>,<o>,<PNIOtypetemplate>,<c>`
- `ctrl<slot>.q<x>,<o>,<PNIOtypetemplate>,<c>`

### 说明

#### **ctrl**

PNIO 控制器的标识符。

#### **<slot>**

“站组态编辑器”中的插槽。

#### **q**

IO 输出子模块的标识符。可读取和写入输出。

#### **i**

IO 输入子模块的标识符。输入为只读。

#### **<x>**

包含输入或输出区域的子模块的编号占位符。

**<o>**

IO 子模块的地址空间内字节偏移量占位符。

**<PNIOtypeTemplate>**

PNIO 模板数据类型转换为 OPC UA 服务器上相应的 OPC UA 数据类型。

下表列出了类型标识符以及可用来表示变量值的对应 OPC 数据类型。

数据类型	OPC UA 数据类型	注释
x<bit>	布尔	位偏移量（0 到 7）的占位符
b	字节	字节（无符号）的占位符
w	UInt16	字（无符号）的占位符
dw	UInt32	双字（无符号）的占位符
lw	UInt64	长字（无符号）的占位符
c	SByte	字节（有符号）的占位符
i	Int16	字（有符号）的占位符
di	Int32	双字（有符号）的占位符
li	Int64	长字（有符号）的占位符
r	浮点	浮点（4 字节）的占位符
lr	双精度	浮点（8 字节）的占位符
s<s >	字符串	字符串长度的占位符

**<c>**

元素数量的占位符。

**示例：**

Nodeld	BrowseName	说明
ctrl.i<x>.<o>,x<bit>,<c>	模板输入位	<x> 子模块的地址 <o> 子模块内的偏移量 <bit> 位偏移量 (0...7) <c> 数组大小
ctrl.i<x>.<o>,b<c>	模板输入字节	<x> 子模块的地址 <o> 子模块内的偏移量 <c> 数组大小

Nodeld	BrowseName	说明
ctrl.i<x>.<o>,w,<c>	模板输入 UInt16	<x> 子模块的地址 <o> 子模块内的偏移量 <c> 数组大小
ctrl.i<x>.<o>,s<sl>,<c>	模板输入字符串	<x> 子模块的地址 <o> 子模块内的偏移量 <sl> 字符串长度 <c> 数组大小
ctrl.q<x>.<o>,x<bit>,<c>	模板输出位	<x> 子模块的地址 <o> 子模块内的偏移量 <bit> 位偏移量 (0...7) <c> 数组大小
ctrl.i<x>.dr<dr>,<l>.<o>,b,<c>	模板输入数据集字节	<x> 子模块的地址 <dr> 数据记录号 <l> 文件长度 <o> 子模块内的偏移量 <c> 数组大小

## 2.14 服务器诊断

### 什么是服务器诊断？

OPC 客户端和 OPC 服务器诊断信息可用于 OPC 数据访问（COM、XML、UA）的“SIMATIC NET PC Software V8.0”及以上版本。例如，可以读出已连接客户端的数目和活动 OPC 组的数目。也可读出 OPC 服务器的版本数据。

---

#### 说明

OPC 服务器诊断（位于 SERVER 下的命名空间中：）仅适用于具有多个选定协议（位于“通信设置”…“协议选择”(Protocol selection) 中）的 OPC.SimaticNET 服务器。对于具有单个协议的数据访问 OPC 服务器（如 OPC.SimaticNET.DP 和 OPC.SimaticNET.PD）并不适用。对 OPC 报警和事件服务器亦不适用。  
此项限制同样适用于 OPC XML 数据访问。

## 2.14.1 协议 ID

### 什么是协议 ID?

用于服务器诊断的协议 ID 称为 SERVER:

该 ID 用于 OPC DA 和 OPC XML DA。

对于 OPC UA DA, 服务器诊断由标准指定; 您将在“服务器”下命名空间 0 中发现“诊断节点”。

## 2.14.2 OPC DA 服务器诊断项

### 存在哪些 OPC DA 服务器诊断项?

OPC 服务器提供了一个定位多种诊断项的分层命名空间。基于 OPC UA 标准, 诊断信息的结构如下:

服务器:

能力

MinSupportedUpdateRate

DiagnosticsSummary

CurrentSessionCount

CumulatedSessionCount

CurrentSubscriptionCount

CumulatedSubscriptionCount

VendorInfo

ComponentVersion

VendorInfoString

诊断 ItemID	数据类型	说明
Server\Capabilities\MinSupportedUpdateRate	VT_UI4	OPC 组的最小更新率
Server\DiagnosticsSummary\CurrentSessionCount	VT_UI4	OPC 客户端会话的当前数目（或 OPC 服务器实例的数目）
Server\DiagnosticsSummary\CumulatedSessionCount	VT_UI4	由于 OPC 服务器已启动，因此所有 OPC 客户端会话已开始。 然而，与此同时，一些会话可能已再次关闭。
Server\DiagnosticsSummary\CurrentSubscriptionCount	VT_UI4	OPC DA 订阅的当前数目，换言之，被激活的 OPC 组的当前数目可以将 OnDataChange 回调发送到 OPC 客户端。
Server\DiagnosticsSummary\CumulatedSubscriptionCount	VT_UI4	由于 OPC 服务器已启动，因此所有 OPC DA 订阅已开始。 然而，与此同时，一些订阅可能已再次关闭。
Server\VendorInfo\ComponentVersion	VT_BSTR	OPC 服务器版本字符串，例如“V03.08.00.00_01.1 4.00.01”
Server\VendorInfo\VendorInfoString	VT_BSTR	OPC 服务器名称，例如 “SIMATIC NET OPC 服务器 DataAccess-V2.05/3.0 (C) SIEMENS AG 2010”

## 2.14 服务器诊断

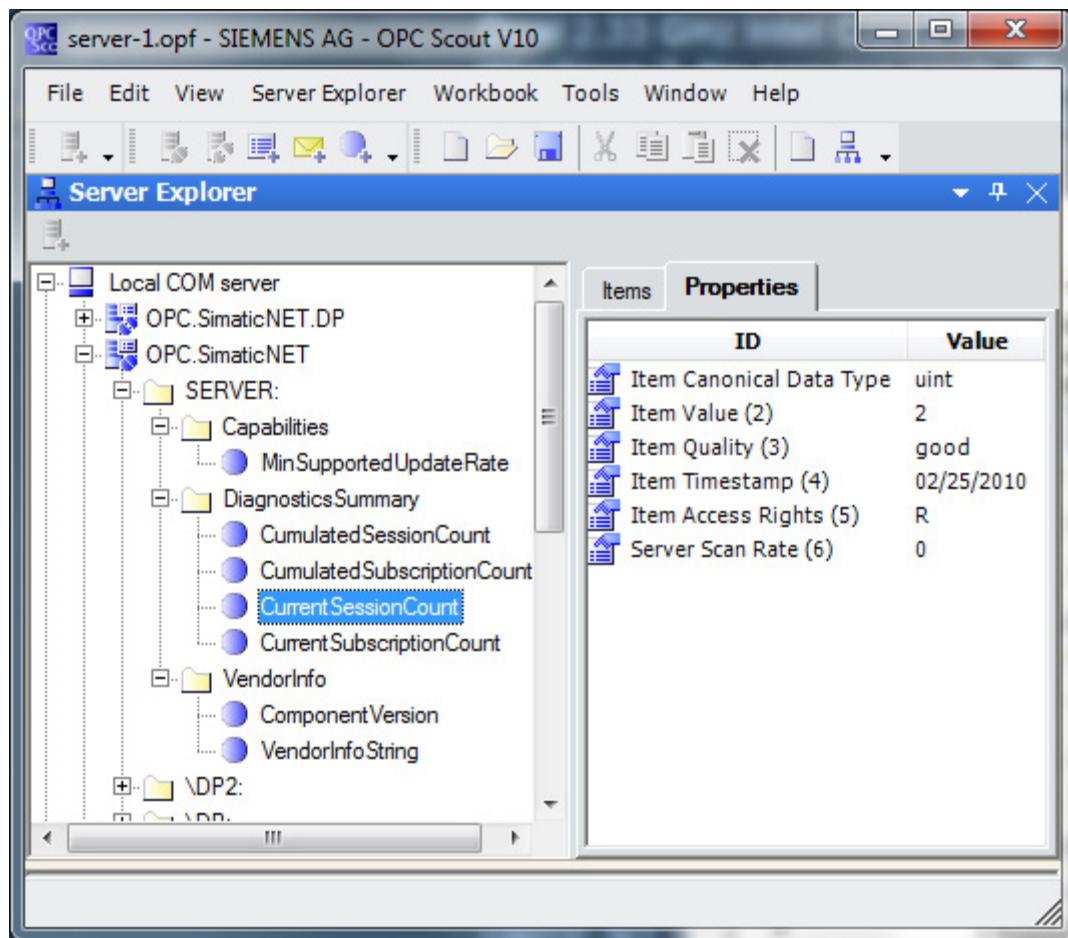


图 2-74 COM/DCOM OPC.SimaticNET 的 OPC 服务器诊断项

### OPC XML DA 诊断项的外观如何？

除了 MinSupportedUpdateRate，OPC DA 服务器中的诊断项都相同，因为在 OPC XML DA 中没有回调。由于没有使用组注册，会话 CurrentSessionCount 的数目将按如下方式实现。

在这种情况下，会话表示持续了较长时间的客户端和服务器之间的通信关系。在 XML DA 方法中，只有订阅是兼容的。

不能分配到会话中的所有其它方法（GetStatus、Browse、Read、Write）单次访问。

XML DA 订阅通过调用“Subscribe”进行创建。客户端使用 PolledRefresh 获取这些订阅的数据。

同时，保证一个列表的所有订阅均视为相同客户端，并因而将这些订阅连接到同一会话中。SessionCount 将根据一个客户端的现有订阅数进行计算。

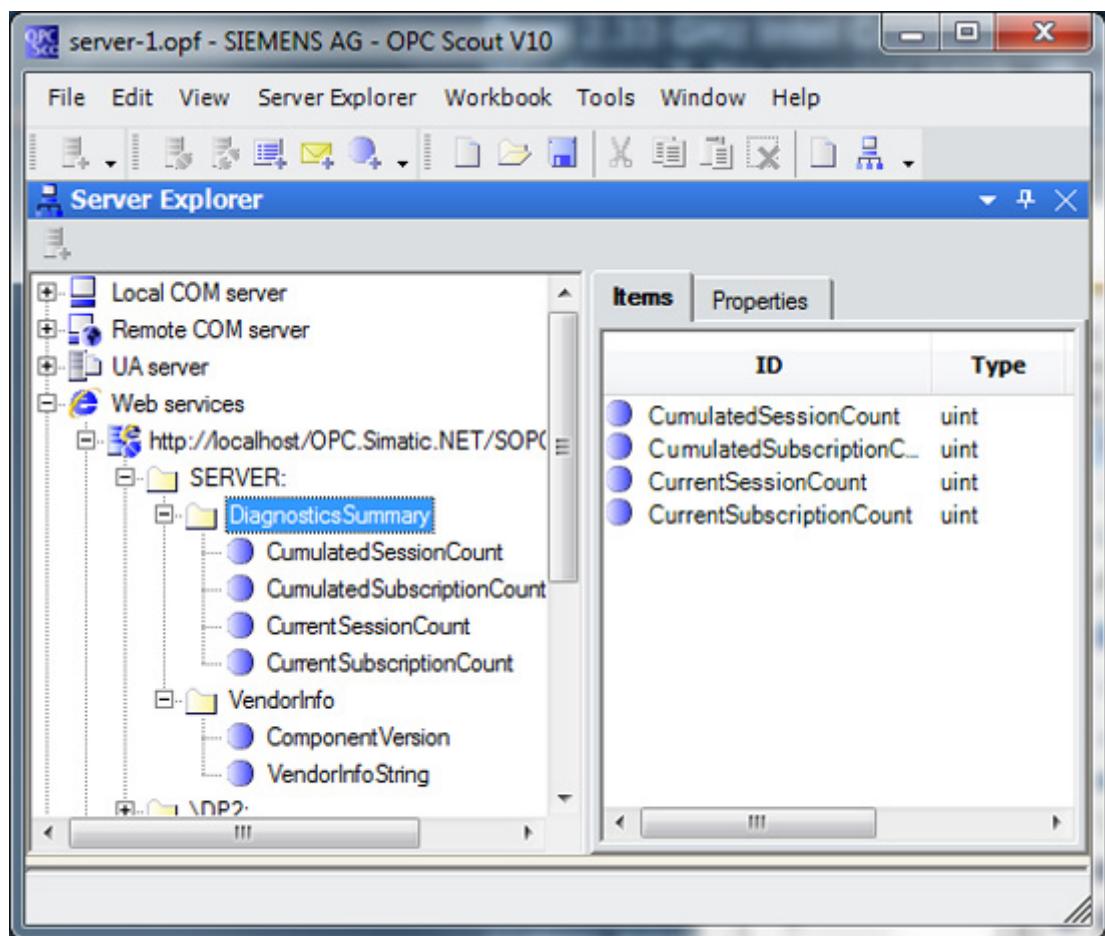


图 2-75 OPC XML DA 服务器诊断项

<http://localhost/OPC.Simatic.NET/SOPCWeb.asmx>

## 2.14 服务器诊断

### OPC UA 服务器诊断提供哪些功能？

OPC UA 服务器诊断用作模型，并提供与其它诊断和信息节点相同的功能。

您将从下图中获得印象。此外，还应参见 OPC UA 规范。

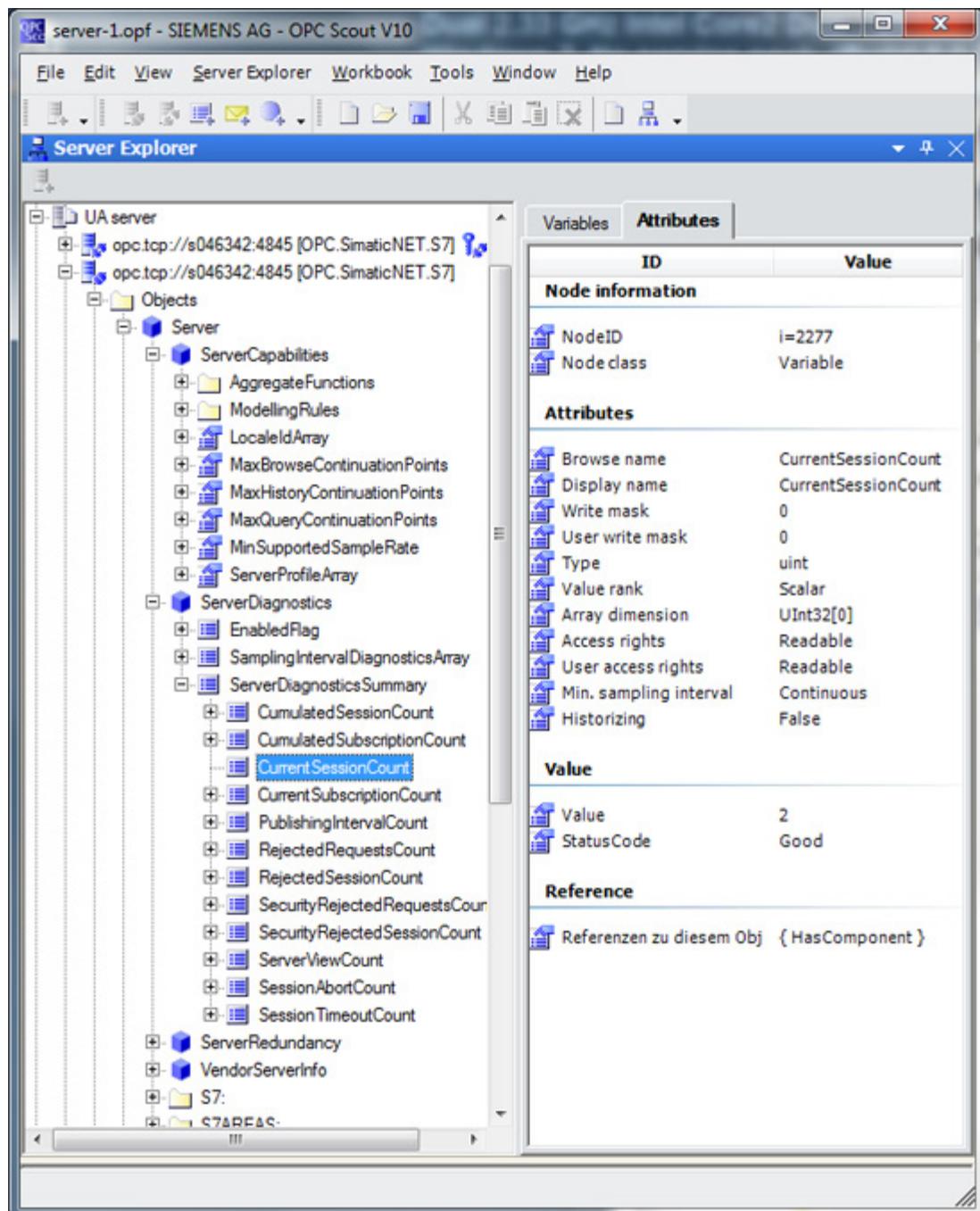


图 2-76 OPC UA 服务器诊断节点

## 2.15 具有 OPC 接口的面向缓冲区的服务

### 简介

本部分介绍了具有 OPC 的面向缓冲区的服务的使用。您将了解将数据缓冲区映射到 OPC 变量的方法。此部分也介绍了将 TCP/IP 本地与面向缓冲区的服务一起使用时要注意的特殊事项。

### 2.15.1 使用缓冲区发送/接收服务

对于大数据包的传送，工业以太网上的 S7 通信和 S5 兼容的通信提供了缓冲区发送/接收服务。数据包将在通信伙伴间进行发送。仅当伙伴明确启动发送工作时，数据传送才会对网络施加负载。

使用 SIMATIC NET 的 OPC 服务器，可以构造数据块。这样一来，数据包的各部分便可被分配给 OPC 数据项。

#### 说明

S7 缓冲区发送/接收服务当前仅对 S7400 和 M7 系列以及 PC 站的设备可用。S5 缓冲区发送/接收服务对 S5 和 S7 系列以及 PC 站的几乎所有设备可用。请注意与 SIMATIC NET CD 主文件夹的自述文件中与面向缓冲区服务的未来支持相关的通信伙伴的信息。

### 2.15.2 面向缓冲区的通信的属性

#### 所支持的服务

在面向缓冲区的服务中，数据缓冲区由发送器使用通信系统发送到接收器。

以下协议支持面向缓冲区的服务：

- S7 通信 (BSEND/BRECEIVE)
- 以太网上的开放式通信服务 (SEND/RECEIVE)
- PROFIBUS (SDA, SDN/指示) 上的开放式通信服务 (SEND/RECEIVE)

面向缓冲区的服务的显著特点是仅在发送器启动传送时传送数据。接收器不能启动数据传输。

## 交换数据缓冲区时发送器和接收器的活动

### 发送器

- 将发送缓冲区与其内容放在一起
- 发送用于与通信伙伴的连接的缓冲区
- 接收数据传输结果的确认

### 接收器

- 准备一个用于连接的接收缓冲区
- 在伙伴向其发送数据缓冲区时收到通知
- 向发送器发送确认
- 评估接收的数据

## 2.15.3 在 OPC 变量上映射数据缓冲区

### 发送项与接收项的区别

OPC 数据访问接口仅识别过程变量。要使面向缓冲区的服务的优点能够与 OPC 配合使用，缓冲区必须映射到 OPC 数据项上：

#### 发送项 (S7 通信：BSEND，开放式通信：发送)

- 一个 OPC 数据项代表一个发送缓冲区或部分发送缓冲区。
- 写入 OPC 数据项（同步/异步）时，一个写入作业将在网络上启动。
- 如果几个代表部分缓冲区的项立即被写入组操作中，则由所有部分组成的整个发送缓冲区将先被放在一起，然后再进行发送。
- 读取将返回从发送缓冲区发送的最后一个数据。  
如果未发送任何数据，则项可能是可读的，然而，其质量为 BAD。

#### 接收项 (S7 通信：BRCV，开放式通信：接收)

- 一个 OPC 数据项代表一个接收缓冲区。
- 如果从设备上读取 OPC 数据项（同步/异步），则通信模块准备接收。  
该状态会一直存在到数据包被接收或者特定连接的超时时间已过。  
如果在超时时间到达之前没有接收任何数据包，则 OPC 数据项的质量为 BAD。
- 如果 OPC  
数据项被监控（激活组中的激活接收项），则一个接收缓冲区将被永久建立在通信模

块上。接收数据包时，如果数据与之前接收的数据不同，则 OnDataChange 回调会将此信号发送到应用程序中。

- 无法写入接收项。

## 2.15.4 使用面向缓冲区的服务

### 处理发送项和接收项

发送项只应被写入；读取或监控发送项将仅返回之前写入的数据。

接收项应被监控；换言之，接收项应以激活组中激活项的形式存在。

无论回调功能是否实现，当每个数据包到达时，OPC 服务器的缓存都会被更新。

### 从缓存读取和直接从设备读取

同步或异步读取作业均应读取缓存。

如果接收项被监控，缓存将包含所接收的最后一个数据块。

使用缓存还可确保读取该项的几个客户端能够同时获取相同的值。

读取至设备（直接读取至设备）没有用，这是因为在连接的超时时间间隔期间只有一个接收缓冲区可用（通常几秒）。

发送器必须在这段时间内发送，否则接收器将不会接受数据缓冲区。

### 更新率

确保 OPC 服务器获取接收缓冲区的速度比发送器发送接收缓冲区的速度更快。

接收缓冲区的获取速率由组特定的更新率 OPC 参数进行指定。

根据协议，所接收的数据块将以其它方式被覆盖而不会告知 OPC 客户端（S7，带有 ISO 的 SEND/RECEIVE 和 RFC

1006），或者数据积压或数据包将形成，并且接收器将仅接收废弃数据（带有 TCP/IP 本地的 SEND/RECEIVE）。

## 2.16 限制 OPC 变量的访问权限

### 2.15.5 使用带有 TCP/IP 本地的面向缓冲区的服务时要注意的几个事项

#### 套接接口的含义

用于带有 TCP/IP 本地的开放式通信的 SEND/RECEIVE 协议基于 Windows Socket 服务。不传送任何显式缓冲区，而是在“套接”接口上传送一个连续的数据流。协议在发送和接收端均使用中间缓冲区，入站和出站数据将在 FIFO 缓冲区内进行管理（先进/先出）。

#### TCP/IP 本地和 OPC 变量

上述协议属性也会影响带有 OPC 变量的应用程序：

- 即使伙伴未接受任何数据，发送作业也会被确认，直到发送端的缓冲区溢出。  
在用户程序中阻止内存溢出。
- 无论一个变量是否被监控或者一个读取作业是否被发送，用于接收数据的中间缓冲区始终可用。
- 如果数据在接收端未被足够快速地获取，则在接收缓冲区将会有数据积压。  
因此，接收器未获取最后读取的数据，而是获取非最新数据。  
在用户程序中阻止数据积压。

## 2.16 限制 OPC 变量的访问权限

#### 在 STEP 7 或 NCM 中指定访问权限

在 STEP 7 或 NCM 中，可以在 OPC 服务器的属性对话框中激活访问权限。

默认情况下，通过激活相关复选框，变量的访问权限被设置为 **Read/Write (RW)**，但在必须设置为 **Read (R)**, **Write (W)** 或 **None** 时，也可对其进行更改。

在“数据项特定的访问权限”对话框中，可以定义不同于所有项目设置的访问权限。  
然后，在此对话框中输入的权限将覆盖默认访问权限。

#### 分配 OPC 数据项特定的访问权限

在此，为一个或多个 OPC 数据项输入访问权限。为其它 OPC 数据项定义的默认权限不会受到影响。语法的定义方式如下：

<OPCItem>=<rights>

**<OPCItem>**

根据在 OPC 文档中定义的语法指定一个或多个 OPC 项。可以使用别名。

可以使用下列占位符：

- \* 任意数量的字符
- ? 恰好一个字符

**<rights>**

- |           |                |
|-----------|----------------|
| <i>RW</i> | 读和写访问          |
| <i>R</i>  | 只读访问           |
| <i>W</i>  | 只写访问           |
| 无         | 既不允许写访问也不允许读访问 |

**求值规则**

此处分配的特定权限的优先级高于在访问权限保护区域中分配的默认权限。

对于此处所分配的特定权限，其值会按此处输入的顺序来确定。如果同一个 OPC 数据项有多个分配，则始终是最后一个分配有效。

**示例：**

```
DP:[CP 5611]Slave040_QB*=RW
DP:[CP 5611]Slave040_QB1=R
DP:[CP 5611]Slave040_QB2=W
DP:[CP 5611]Slave040_QB1=W
DP:[CP 5611]Slave040_QB1*=R
```

作为输入的结果，以下访问权限有效：

```
DP:[CP 5611]Slave040_QB2=W
```

## 2.16 限制 OPC 变量的访问权限

DP:[CP 5611]Slave040\_QB1=R

### 说明

访问权限分配的语法中的前导零总是被忽略。

因此，您不得在前导零可能被定位的位置使用通配符 ? 或 \*!

示例：

定义 *Slave0?9M06\_QB1=R* 必须用以下两个定义代替：

*Slave?9M06\_QB1=R*

*Slave9M06\_QB1=R*

### 说明

此处描述的访问权限限制不适用于 OPC UA 服务器。





# 用于 SIMATIC NET 的 OPC 报警和事件服务器

## 3.1 用于 S7 通信的事件服务器

### 3.1.1 功能和报警类别

#### 简介

本部分介绍了由 OPC 服务器返回的事件，以及以属性形式提供的附加信息。

#### SIMATIC NET 的 A&E 服务器

SIMATIC NET 的 OPC 服务器可用作“简单事件服务器”或“条件报警和事件服务器”。简单事件服务器没有关于在操作员监控系统中进行的报警和事件处理的组态信息。

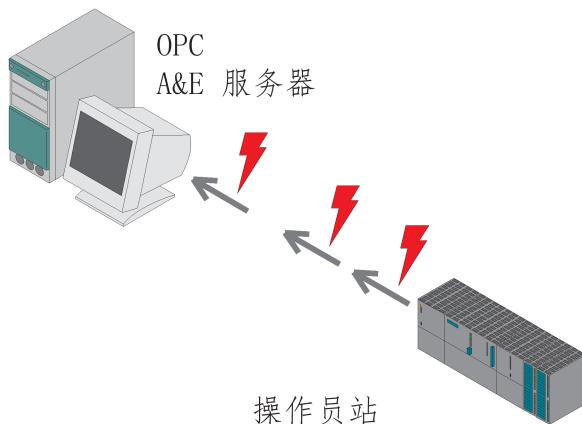


图 3-1 客户端与 A&E 服务器之间的报警和事件通信的原则

“条件相关事件”可能包括子条件。也可以为一次报警定义几个子条件。  
当满足子条件时，将会发生此类事件。

条件相关事件也可评估由服务器和确认选项激活或取消激活的条件。

### 3.1 用于 S7 通信的事件服务器

#### 集成事件服务器

“简单事件服务器”只将简单消息传递给底层组件。

高级报警/事件管理服务器处理低级简单服务器的消息时，将考虑组态信息。

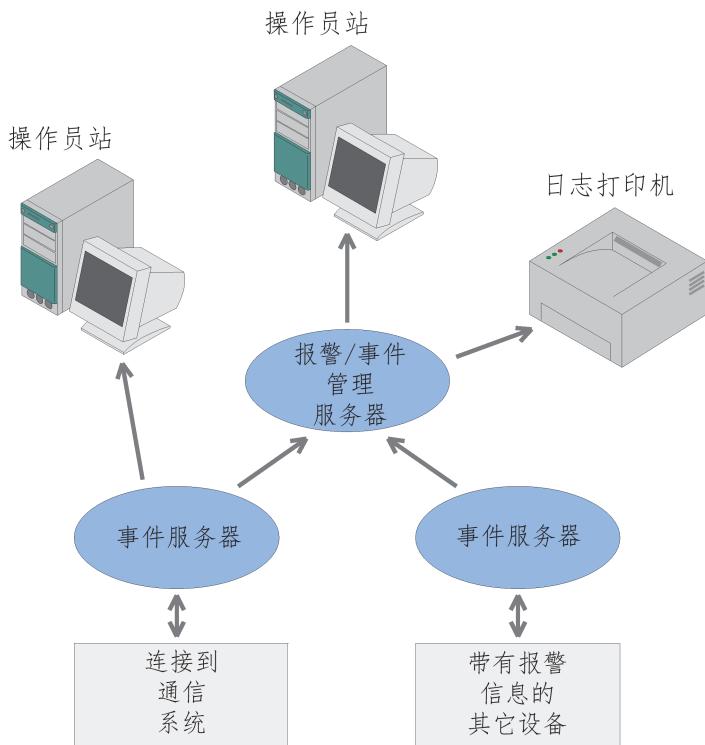


图 3-2 集成事件服务器

#### S7 协议

S7 协议提供协议机制“已编程的报警 (ALARM)”来传送事件。

消息类型由“简单事件服务器”进行使用。

## 报警类别

以下报警和事件类别适用于 S7 通信：

事件类别 值	说明
EVENTCATEGORY_S7SIMPLE 2	以下 S7 报警的简单事件： NOTIFY ALARM ALARM_8 ALARM_8P
EVENTCATEGORY_DIAGNOSIS 12	包含模块诊断缓冲区中的条目（简单事件）。
EVENTCATEGORY_S7CONDITION 13	以下 S7 报警的条件事件： NOTIFY ALARM ALARM_8 ALARM_8P ALARM_S ALARM_SQ
EVENTCATEGORY_STATEPATH 14	指示设备连接中断的报警（条件相关事件）。
EVENTCATEGORY_CAT_LEVEL 40	级别报警 报警指示违反 最小值/最大值
EVENTCATEGORY_CAT_DEVIATION 41	容差报警 报警指示容差偏离
EVENTCATEGORY_CAT_ROC 42	变化率报警 报警指示变化频率
EVENTCATEGORY_CAT_OFFNORMAL 43	不正常报警 报警指示与正常状态有差异 所有 S7 报警 SFB、SFC 和 SCAN 的默认值
EVENTCATEGORY_CAT_TRIP 44	脱扣报警 报警指示断路器脱扣

### 3.1 用于 S7 通信的事件服务器

事件类别 值	说明
EVENTCATEGORY_CAT_COS 45	状态变化报警 报警指示状态变化
EVENTCATEGORY_CAT_DEVICEFAILURE 46	设备故障 报警指示设备出现故障
EVENTCATEGORY_CAT_SYSTEMFAILURE 47	系统故障 报警指示系统出现故障
EVENTCATEGORY_CAT_SYSTEMMESSAGE 60	系统消息 (简单事件)

#### 说明

事件类别 2 到 14 用于通过报警文本文件组态 8.0 以下版本的 OPC 服务器。对于 8.0 及更高版本的 OPC 服务器，类别 40 到 60 适用。

表格 3-1 报警和扫描分配到报警类别 43(CAT\_OFFNORMAL) 或按下表所示设定的 STEP 7 报警类别分配：

STEP 7 报警类别	OPC 报警类别
无报警	禁止报警 (可由用户设置，例如在 Alarm_8p 报警位过多时)。
未指定或未知的报警类别	报警类别 43 (EVENTCATEGORY_CAT_OFFNORMAL) (默认)
报警 - 高	报警类别 40 (EVENTCATEGORY_CAT_LEVEL) 条件“HIHI”
报警 - 低	条件“LOLO”
警告 - 高	条件“HI”
警告 - 低	条件“LO”

STEP 7 报警类别	OPC 报警类别
容差 - 高 容差 - 低	报警类别 41 (EVENTCATEGORY_CAT_DEVIATION) 条件“HI” 条件“LO”
AS 过程控制报警 - 故障 AS 过程控制报警 - 错误 OS 过程控制报警 - 故障 预防性维护 - 常规 过程报警 - 需要确认 操作报警 - 无需确认 操作员提示 - 常规 操作员输入消息 - 常规 状态消息 - AS 状态消息 - OS	报警类别 43 (EVENTCATEGORY_CAT_OFFNORMAL) 条件“CFN”

### 3.1.2 事件参数

#### 所有事件的参数

客户端通过调用 *OnEvent* 回调函数获取事件列表。OPC 事件服务器提供哪些参数取决于事件类型。以下参数与所有事件相关：

参数	含义
szSource	作为消息源，OPC 事件服务器指定发出报告的 S7 设备的连接信息。其组成如下： S7:<连接名称>
ftTime	事件发生的时间。
szMessage	消息名称由消息机制和组态的消息编号组成： ALARM<报警编号> 示例： ALARM55 (请参见下文注释)

## 3.1 用于 S7 通信的事件服务器

参数	含义
dwEventType	OPC 事件服务器支持以下事件： OPC_SIMPLE_EVENT，此常数值为 0x0001。 OPC_CONDITION_EVENT，此常数值为 0x0004。
dwEventCategory	以下值可以用作事件类别： 对于 8.0 以下版本的 OPC 服务器： EVENTCATEGORY_S7SIMPLE (2) EVENTCATEGORY_S7T0 (11) EVENTCATEGORY_DIAGNOSIS (12) EVENTCATEGORY_S7CONDITION (13) EVENTCATEGORY_STATEPATH (14) 对于 8.0 及以上版本的 OPC 服务器： EVENTCATEGORY_CAT_LEVEL (40) EVENTCATEGORY_CAT_DEVIATION (41) EVENTCATEGORY_CAT_ROC (42) EVENTCATEGORY_CAT_OFFNORMAL (43) EVENTCATEGORY_CAT_TRIP (44) EVENTCATEGORY_CAT_COS (45) EVENTCATEGORY_CAT_DEVICEFAILURE (46) EVENTCATEGORY_CAT_SYSTEMFAILURE (47) EVENTCATEGORY_CAT_SYSTEMMESSAGE (60)
dwSeverity	为消息的严重程度返回预定义值。可在 STEP 7 组态数据库中为通信伙伴的各个消息编号修改此值。
dwNumEventAttrs	消息提供的属性数取决于提供的关联值数。
pEventAttributes	此结构包含事件提供的属性。 属性还包括由伙伴设备提供的消息的关联值。

---

## 说明

**报警块消息权重高于为报警消息指定的优先级**

**使用 OPC**

报警和事件服务器“OPC.SimaticNetAlarms”，可接收带报警权重（“严重程度”）的 S7 报警。可在 STEP 7/NetPro 的 OPC 服务器的 S7 连接属性中组态这些消息。请注意：

**S7 程序中 ALARM、ALARM\_8P 或 NOTIFY**

块的预设报警权重优先于为报警消息设置的可组态优先级。

**在 NetPro**

中为特定报警编号组态的报警优先级同样高于报警消息的常规优先级设置和预设报警权重。**ALARM\_S 和 ALARM\_SQ 没有严重程度，所以始终使用组态的报警权重。**

---

## 条件相关事件的参数

为条件相关事件传送以下附加参数：

参数	含义
szConditionName	条件名称。
szSubconditionName	子条件名称。如果没有子条件，此参数采用条件名称。
wChangeMask	指示哪种状态变化导致发送消息。可能值示例包括 OPC_CHANGE_QUALITY 或 OPC_CHANGE_SUBCONDITION。
wNewState	szSubconditionName 参数所含子条件的状态。 可能的值包括： OPC_CONDITION_ACTIVE OPC_CONDITION_ENABLE OPC_CONDITION_ACKED
wQuality	包含消息相关值的质量信息。
bAckRequired	指示消息是否需要确认。
ftActiveTime	激活子条件的时间。 如果不存在子条件，则此参数包含满足条件的起始时间。
dwCookie	服务器为消息分配的 Cookie。
szActorID	包含有关事件起因的应用程序特定信息。

### 3.1.3 事件属性

S7 站发送与消息或报警关联的值。对于 EVENT\_ATTR\_S7\_DATA $n$ 、EVENT\_ATTR\_S7\_DATA $n$ \_DATATYPE 和 EVENT\_ATTR\_S7\_DATA $n$ \_VALUE\_LEN， $n$  值最多可为 10 个关联值。通过 OPC 报警和事件接口在事件属性中提供关联值。

编程消息 (ALARM) 提供以下属性：

- EVENT\_ATTR\_S7\_MSGTEXT
- EVENT\_ATTR\_S7\_ALARMSTATE
- EVENT\_ATTR\_S7\_TYPE
- EVENT\_ATTR\_S7\_SEVERITY
- EVENT\_ATTR\_S7\_CATEGORYID
- EVENT\_ATTR\_S7\_CATEGORYDESC
- EVENT\_ATTR\_S7\_COMMENT
- EVENT\_ATTR\_S7\_ACTIVETIME
- EVENT\_ATTR\_S7\_PCTIME
- EVENT\_ATTR\_S7\_S7TIME
- EVENT\_ATTR\_S7\_STATE
- EVENT\_ATTR\_S7\_ACK\_STATE
- EVENT\_ATTR\_S7\_EVENT\_STATE
- EVENT\_ATTR\_S7\_NO\_DATA
- EVENT\_ATTR\_S7\_DATA $n$
- EVENT\_ATTR\_S7\_DATA $n$ \_DATATYPE
- EVENT\_ATTR\_S7\_DATA $n$ \_VALUE\_LEN
- EVENT\_ATTR\_S7\_EVENT\_EVENTID
- EVENT\_ATTR\_S7\_EVENT\_SUBEVENTID
- EVENT\_ATTR\_S7\_CONNECTION
- EVENT\_ATTR\_S7.Areas
- EVENT\_ATTR\_S7\_INFOTEXT
- EVENT\_ATTR\_S7\_TEXT1

- EVENT\_ATTR\_S7\_TEXT2
- EVENT\_ATTR\_S7\_TEXT3
- EVENT\_ATTR\_S7\_TEXT4
- EVENT\_ATTR\_S7\_TEXT5
- EVENT\_ATTR\_S7\_TEXT6
- EVENT\_ATTR\_S7\_TEXT7
- EVENT\_ATTR\_S7\_TEXT8
- EVENT\_ATTR\_S7\_TEXT9

### **EVENT\_ATTR\_S7\_MSGTEXT**

值:	-1
数据类型:	VT_BSTR

属性值: 组态的报警文本, 与语言相关。如果未组态, 则与 EventID 完全相同

### **EVENT\_ATTR\_S7\_ALARMSTATE**

值:	-2
数据类型:	VT_UI4

属性值: 报警状态 (EVENT\_ATTR\_S7\_ALARMSTATE)

EVENT_ATTR_S7_ALARMSTATE	说明
CHANGE_ACTIVE   ACTIVE   ACKREQUIRED	在 opc_ae.h 头文件中指定各个 #Defines (ACTIVE...) 的值。 ALARM/ALARM_8/ALARM_8P/ALARM_SQ/ALARM_DQ/SCAN 进入的状态
CHANGE_ACTIVE   ACTIVE   ACKED	ALARM_S/ALARM_D/NOTIFY/NOTIFY_8P 进入的状态 (隐式确认) 不得设置 CHANGE_ACKED 标记。
ACTIVE   ACKREQUIRED	ALARM/ALARM_8/ALARM_8P/ALARM_SQ/ALARM_DQ/SCAN 激活 (仅限于刷新)

### 3.1 用于 S7 通信的事件服务器

EVENT_ATTR_S7_ALARMSTATE	说明
ACTIVE   ACKED	ALARM_S/ALARM_D/NOTIFY/NOTIFY_8P 激活（隐式确认） (仅限于刷新)
ACTIVE   CHANGE_ACKED   ACKED	ALARM/ALARM_8/ALARM_8P/ALARM_SQ/ALAR M_DQ/SCAN 激活且确认到达
ACTIVE   ACKED	ALARM/ALARM_8/ALARM_8P/ALARM_SQ/ALAR M_DQ/SCAN 激活并确认 (仅限于刷新)
CHANGE_INACTIVE   ACKREQUIRED	ALARM/ALARM_8/ALARM_8P/ALARM_SQ/ALAR M_DQ/SCAN 退出的状态且未确认
CHANGE_INACTIVE   ACKED	ALARM_S/ALARM_D/NOTIFY/NOTIFY_8P 退出的状态（隐式确认）
CHANGE_INACTIVE   ACKED	ALARM/ALARM_8/ALARM_8P/ALARM_SQ/ALAR M_DQ/SCAN 退出的状态并确认
ACKREQUIRED	ALARM/ALARM_8/ALARM_8P/ALARM_SQ/ ALARM_DQ/SCAN 未激活且未确认 (仅限于刷新)

### EVENT\_ATTR\_S7\_TYPE

值:	-5
数据类型:	VT_UI4

属性值: EVENT\_TYPE\_CONDITION

### EVENT\_ATTR\_S7\_SEVERITY

值:	-6
数据类型:	VT_I4

属性值: 严重程度

严重程度值指明子条件的权重或优先级。值范围介于 1 至 1000。值越高，权重越高。

**EVENT\_ATTR\_S7\_CATEGORYID**

值:	-9
数据类型:	VT_I4

属性值: 类别标识符

可能的值列在“功能和报警类别 (页 445)”部分的“报警和事件类别”表中。

**EVENT\_ATTR\_S7\_CATEGORYDESC**

值:	-10
数据类型:	VT_BSTR

属性值: 类别描述

**EVENT\_ATTR\_S7\_COMMENT**

值:	-13
数据类型:	VT_BSTR

属性值: 组态的附加文本 9, 与语言相关。

**EVENT\_ATTR\_S7\_ACTIVETIME**

值:	-15
数据类型:	VT_DATE

属性值: 信号从“报警进入的状态”开始发生变化的时间（取自时间戳）。

**EVENT\_ATTR\_S7\_PCTIME**

值:	6000
数据类型:	VT_DATE

属性值: OPC 事件服务器收到报警的时间。

### 3.1 用于 S7 通信的事件服务器

#### EVENT\_ATTR\_S7\_S7TIME

值:	6001
数据类型:	VT_DATE

属性值: 在伙伴设备上生成消息的时间。

#### EVENT\_ATTR\_S7\_STATE

值:	6002
数据类型:	VT_UI1 (条件事件) VT_UI2 (简单事件)

属性值: 指示消息是否存在的常规状态。

#### ALARM 消息

位	表示:
0x00H (所有位均为 0)	消息存在
位 0 置位	初始化
位 1 置位	溢出信号
位 2 置位	溢出实例
位 3 到 5	0, 保留
位 6 置位	附加值无法输入 (大小)
位 7 置位	附加值不可获取

#### EVENT\_ATTR\_S7\_ACK\_STATE

值:	6003
数据类型:	VT_UI2

属性值: S7 可编程控制器上的报警确认状态

无法使用 SIMATIC NET 的 OPC 事件服务器确认消息。

但可由其他用户界面系统确认消息。

ALARM 消息:

位	表示:
0	已对“消息 1 进入的状态”执行确认
1 ... 6	已对“消息 2 到 7 进入的状态”执行确认（仅 ALARM_8/ALARM_8P）
7	已对“消息 8 进入的状态”执行确认（仅 ALARM_8/ALARM_8P）
8	已对“消息 1 退出的状态”执行确认
9 ... 14	已对“消息 2 到 7 退出的状态”执行确认（仅 ALARM_8/ALARM_8P）
15	已对“消息 8 退出的状态”执行确认（仅 ALARM_8/ALARM_8P）

使用 SIMATIC NET 的 OPC 事件服务器只能确认“消息 n 进入的状态”。

但可由其他用户界面系统确认消息。

#### **EVENT\_ATTR\_S7\_EVENT\_STATE**

值:	6004
数据类型:	VT_UI1（条件相关事件） VT_UI2（简单事件）

属性值: 事件状态

位	表示:
0	当前状态“消息 1”（1 = 激活）
1 ... 6	当前状态“消息 2 到 7”（1 = 激活）（仅 ALARM_8/ALARM_8P）
7	当前状态“消息 8”（1 = 激活）（仅 ALARM_8/ALARM_8P）

#### **EVENT\_ATTR\_S7\_NO\_DATA**

值:	6005
数据类型:	VT_UI1（条件事件） VT_UI2（简单事件）

属性值: 关联值数 值范围: 1...10

### 3.1 用于 S7 通信的事件服务器

#### EVENT\_ATTR\_S7\_DATA<sub>n</sub>

值:	其中 $n = 0$	6008
	其中 $n = 1$	6012
	其中 $n = 2$	6016
	其中 $n = 3$	6020
	其中 $n = 4$	6024
	其中 $n = 5$	6028
	其中 $n = 6$	6032
	其中 $n = 7$	6036
	其中 $n = 8$	6040
	其中 $n = 9$	6044
数据类型:	VT_ARRAY VT_UI1	

属性值： 关联值数“n”的相关字节（类似于字节数组）。

#### EVENT\_ATTR\_S7\_DATA<sub>n</sub>\_DATATYPE

值:	其中 $n = 0$	6006
	其中 $n = 1$	6010
	其中 $n = 2$	6014
	其中 $n = 3$	6018
	其中 $n = 4$	6022
	其中 $n = 5$	6026
	其中 $n = 6$	6030
	其中 $n = 7$	6034
	其中 $n = 8$	6038
	其中 $n = 9$	6042
数据类型:	VT_UI2:	针对简单事件
	VT_UI1:	针对条件相关事件

属性值： 关联值数“n”的数据类型。

参数值	说明
S7_DATATYPE_ERROR	错误 (0x0)
S7_DATATYPE_BOOLEAN	布尔 (0x03)
S7_DATATYPE_INTEGER	整型 (0x05)
S7_DATATYPE_UNSIGNED	整型 (0x06)
S7_DATATYPE_FLOAT	浮点 (0x07)
S7_DATATYPE_OCTET_STRING	字符串 (0x09)
S7_DATATYPE_BITSTRING	位字符串 (0x04) 注： 长度信息以字节（而不是位）为单位
S7_DATATYPE_DATE	日期 (0x30) 注： 日期从 01.01.1990 开始
S7_DATATYPE_TIME_OF_DAY	时间 (0x31) 注： 从起始日起的 ms
S7_DATATYPE_TIME	时间 (0x32) 注： ms
S7_DATATYPE_S5TIME	时间 (0x33) 注： BCD 编码
S7_DATATYPE_DATE_AND_TIME	日期和时间 (0x34)

### 3.1 用于 S7 通信的事件服务器

#### EVENT\_ATTR\_S7\_DATA<sub>n</sub>\_VALUE\_LEN

值:	其中 $n=0$	6007
	其中 $n=1$	6011
	其中 $n=2$	6015
	其中 $n=3$	6019
	其中 $n=4$	6023
	其中 $n=5$	6027
	其中 $n=6$	6031
	其中 $n=7$	6035
	其中 $n=8$	6039
	其中 $n=9$	6043
数据类型:	VT_UI2: VT_UI2:	针对简单事件 针对条件相关事件

属性值： 关联值数“n”的相关字节数。

#### EVENT\_ATTR\_S7\_EVENT\_EVENTID

值:	6046
数据类型:	VT_UI4 (仅限条件相关事件)

属性值： 报警编号

#### EVENT\_ATTR\_S7\_EVENT\_SUBEVENTID

值:	6047
数据类型:	VT_UI1 (仅限条件相关事件)

属性值： ALARM\_8/ALARM\_8P 类型的消息为 SubEventID (值范围 1 到 8)， 其它情况下为 1。

**EVENT\_ATTR\_S7\_CONNECTION**

值:	6050
数据类型:	VT_BSTR

属性值: S7 连接名称

**EVENT\_ATTR\_S7.Areas**

值:	6051
数据类型:	VT_ARRAY of VT_BSTR

属性值: 枚举诊断事件的区域。

**EVENT\_ATTR\_S7\_INFOTEXT**

值:	6060
数据类型:	VT_BSTR

属性值: 组态的信息文本, 与语言相关。

**EVENT\_ATTR\_S7\_TEXT1**

值:	6061
数据类型:	VT_BSTR

属性值: 组态的附加文本 1, 与语言相关。

**EVENT\_ATTR\_S7\_TEXT2**

值:	6062
数据类型:	VT_BSTR

属性值: 组态的附加文本 2, 与语言相关。

### 3.1 用于 S7 通信的事件服务器

#### EVENT\_ATTR\_S7\_TEXT3

值:	6063
数据类型:	VT_BSTR

属性值: 组态的附加文本 3, 与语言相关。

#### EVENT\_ATTR\_S7\_TEXT4

值:	6064
数据类型:	VT_BSTR

属性值: 组态的附加文本 4, 与语言相关。

#### EVENT\_ATTR\_S7\_TEXT5

值:	6065
数据类型:	VT_BSTR

属性值: 组态的附加文本 5, 与语言相关。

#### EVENT\_ATTR\_S7\_TEXT6

值:	6066
数据类型:	VT_BSTR

属性值: 组态的附加文本 6, 与语言相关。

#### EVENT\_ATTR\_S7\_TEXT7

值:	6067
数据类型:	VT_BSTR

属性值: 组态的附加文本 7, 与语言相关。

**EVENT\_ATTR\_S7\_TEXT8**

值:	6068
数据类型:	VT_BSTR

属性值: 组态的附加文本 8, 与语言相关。

**EVENT\_ATTR\_S7\_TEXT9**

值:	6069
数据类型:	VT_BSTR

属性值: 组态的附加文本 9, 与语言相关。

### 3.1 用于 S7 通信的事件服务器

#### 各种时间戳的含义

属性	新报警从 S7 设备到达:	Refresh
参数TimeStamp	根据时间戳原始组态的设置: 1) 沿变化到达: 信号沿的 S7 时间 2) 确认到达: 确认的 S7 时间 或 1) 沿变化到达: 信号沿的 S7 时间 +/- 偏移 2) 确认到达: 确认的 S7 时间 +/- 偏移 或 1) 沿变化到达: 通知的 PC 时间 2) 确认到达: 通知的 PC 时间	根据时间戳原始组态的设置: 信号沿或确认的最新 S7 时间 或 信号沿或确认的最近 S7 时间 +/- 偏移 或 沿变化或确认的最新通知的 PC 时间
EVENT_ATTR_S7_S7TIME	1) 沿变化到达: 信号沿的 S7 时间 2) 确认到达: 确认的 S7 时间	信号沿或确认的最新 S7 时间
EVENT_ATTR_S7_PCTIME	1) 沿变化到达: 通知的 PC 时间 2) 确认到达: 通知的 PC 时间	沿变化或确认的最新通知的 PC 时间

最初建立或中断连接时，S7 报警更新不提供信号类型为 ALARM/ALARM\_8/ALARM\_8P/ALARM\_SQ/ALARM\_DQ/SCAN 的 S7 时间戳。 ATTR\_S7\_S7TIME 属性之后包含时间戳 01.01.1990（可能的最早 S7 时间），否则根据时间戳 01.01.1990 或 PC 时间形成时间戳，具体取决于时间戳原始组态中的设置。

#### 说明

请注意，OPC 接口上的事件并不转发报警的所有状态变化。如果报警状态快速变化，事件可能不报告某些状态变化，只发送上一状态和当前状态。这取决于 OPC 参数（如“dwBufferTime”）以及组态和计算机性能。

### 3.1.4 模块诊断缓冲区中的条目属性

#### EVENT\_ATTR\_DIAGNOSIS\_MSGTEXT

值:	-1
数据类型:	VT_BSTR

属性值: 组态的报警文本, 与语言相关。如果未组态, 则与 EventID 完全相同。

但在理想情况下, SIMOTION 环境使用自身的报警文本服务器。

#### EVENT\_ATTR\_DIAGNOSIS\_TYPE

值:	-5
数据类型:	VT_I4

属性值: EVENT\_TYPE\_SIMPLE

#### EVENT\_ATTR\_DIAGNOSIS\_SEVERITY

值:	-6
数据类型:	VT_I4

属性值: 严重程度

#### EVENT\_ATTR\_DIAGNOSIS\_CATEGORYID

值:	-9
数据类型:	VT_I4

属性值: 类别标识符

#### EVENT\_ATTR\_DIAGNOSIS\_CATEGORYDESC

值:	-10
数据类型:	VT_I4

属性值: 类别描述

### 3.1 用于 S7 通信的事件服务器

#### EVENT\_ATTR\_DIAGNOSIS\_S7\_PCTIME

值:	6001
数据类型:	VT_DATE

属性值: SIMATIC NET S7 OPC 服务器的时间戳

#### EVENT\_ATTR\_DIAGNOSIS\_S7\_DIAGNOSISID

值:	6048
数据类型:	VT_I4

属性值: 诊断事件编号, 模拟 EventID

#### EVENT\_ATTR\_DIAGNOSIS\_S7\_DIAGNOSISDATA

值:	6049
数据类型:	VT_ARRAY 或 VT_UI1

属性值: 有关诊断事件或其影响的信息, 数据长度取决于事件类别。

---

#### 说明

有关 SFC 52 (S7 诊断数据) 的详细说明, 请参见“S7-300/400 的 STEP 7 系统和标准功能”文档。

---

#### EVENT\_ATTR\_DIAGNOSIS\_S7\_CONNECTION

值:	6050
数据类型:	VT_BSTR

属性值: S7 连接名称

**EVENT\_ATTR\_DIAGNOSIS\_S7.Areas**

值:	6051
数据类型:	VT_ARRAY VT_BSTR

属性值：枚举诊断事件的区域。当前最多有一个区域可用。默认值：空

或

如果有任何有关报警的组态信息：

connections\<connectionname>

或

组态的区域

**EVENT\_ATTR\_DIAGNOSIS\_S7\_INFOTEXT**

值:	6060
数据类型:	VT_BSTR

属性值：组态的信息文本，与语言相关。如果未组态则为空。

**EVENT\_ATTR\_DIAGNOSIS\_S7\_TEXT1**

值:	6061
数据类型:	VT_BSTR

属性值：组态的附加文本 1（可能与源完全相同），与语言相关。如果未组态则为空。

**EVENT\_ATTR\_DIAGNOSIS\_S7\_TEXT2**

值:	6062
数据类型:	VT_BSTR

属性值：组态的附加文本 2（可能用作区域），与语言相关。如果未组态则为空。

### 3.1 用于 S7 通信的事件服务器

#### EVENT\_ATTR\_DIAGNOSIS\_S7\_TEXT3

值:	6063
数据类型:	VT_BSTR

属性值: 组态的附加文本 3, 与语言相关。如果未组态则为空。

#### EVENT\_ATTR\_DIAGNOSIS\_S7\_TEXT4

值:	6064
数据类型:	VT_BSTR

属性值: 组态的附加文本 4, 与语言相关。如果未组态则为空。

#### EVENT\_ATTR\_DIAGNOSIS\_S7\_TEXT5

值:	6065
数据类型:	VT_BSTR

属性值: 组态的附加文本 5, 与语言相关。如果未组态则为空。

#### EVENT\_ATTR\_DIAGNOSIS\_S7\_TEXT6

值:	6066
数据类型:	VT_BSTR

属性值: 组态的附加文本 6, 与语言相关。如果未组态则为空。

#### EVENT\_ATTR\_DIAGNOSIS\_S7\_TEXT7

值:	6067
数据类型:	VT_BSTR

属性值: 组态的附加文本 7, 与语言相关。如果未组态则为空。

**EVENT\_ATTR\_DIAGNOSIS\_S7\_TEXT8**

值:	6068
数据类型:	VT_BSTR

属性值: 组态的附加文本 8, 与语言相关。如果未组态则为空。

**EVENT\_ATTR\_DIAGNOSIS\_S7\_TEXT9**

值:	6069
数据类型:	VT_BSTR

属性值: 组态的附加文本 9, 与语言相关。如果未组态则为空。

**EVENT\_ATTR\_DIAGNOSIS\_S7\_S7TIME**

值:	6100
数据类型:	VT_DATE

属性值: 时间戳

**各种时间戳的含义**

属性	新事件从 S7 设备到达:
参数 TimeStamp	根据 EventTimeBias 的设置: 诊断事件的 S7 时间 或 诊断事件的 S7 时间 + 偏移 或 诊断事件到达的 PC 时间。
EVENT_ATTR_DIAGNOSIS_S7TIME	诊断事件的 S7 时间
EVENT_ATTR_DIAGNOSIS_PCTIME	诊断事件到达的 PC 时间

### 3.1 用于 S7 通信的事件服务器

#### 参见

[指示连接中断的报警属性 \(页 470\)](#)

#### 3.1.5 指示连接中断的报警属性

##### EVENT\_ATTR\_STATEPATH\_MSGTEXT

值:	-1
数据类型:	VT_BSTR

属性值: 组态的报警文本, 与语言相关。如果未组态, 则与 EventID 完全相同。

##### EVENT\_ATTR\_STATEPATH\_ALARMSTATE

值:	-2
数据类型:	VT_UI4

属性值: 报警状态 (EVENT\_ATTR\_STATEPATH\_ALARMSTATE)

EVENT_ATTR_STATEPATH_ALARMSTATE	说明
CHANGE_ACTIVE   ACTIVE	在 opc_ae.h 头文件中指定各个 #Defines (ACTIVE...) 的值。 STATEPATH 进入的状态 (不通过“刷新”)
ACTIVE	STATEPATH 激活
CHANGE_INACTIVE   INACTIVE	STATEPATH 退出的状态 (不通过“刷新”)

##### EVENT\_ATTR\_STATEPATH\_TYPE

值:	-5
数据类型:	VT_I4

属性值: EVENT\_TYPE\_CONDITION

**EVENT\_ATTR\_STATEPATH\_SEVERITY**

值:	-6
数据类型:	VT_I4

属性值: 严重程度

**EVENT\_ATTR\_STATEPATH\_CATEGORYID**

值:	-9
数据类型:	VT_I4

属性值: 类别标识符

**EVENT\_ATTR\_STATEPATH\_CATEGORYDESC**

值:	-10
数据类型:	VT_BSTR

属性值: 类别描述

**EVENT\_ATTR\_STATEPATH\_COMMENT**

值:	-13
数据类型:	VT_BSTR

属性值: 注释

**EVENT\_ATTR\_STATEPATH\_ACTIVETIME**

值:	-15
数据类型:	VT_DATE

属性值: 信号变为“报警进入的状态”的时间（取自时间戳）。

### 3.1 用于 S7 通信的事件服务器

#### EVENT\_ATTR\_STATEPATH\_CONNECTION

值:	6050
数据类型:	VT_BSTR

属性值: S7 连接名称

#### EVENT\_ATTR\_STATEPATH.Areas

值:	6051
数据类型:	VT_ARRAY of VT_BSTR

属性值: 枚举状态路径报警的区域。

### 3.1.6 组态报警文本、源和区域

#### 支持哪些报警?

OPC 报警和事件服务器支持以下报警:

- 符号相关报警 (SCAN)

允许监视与 PLC 用户程序异步的 CPU 中 I、Q、M 和 DB 区域内的位。

- 块相关报警 (报警 SFB、报警 SFC)

可确认的报警 SFB 包括: ALARM (SFB 33)、ALARM\_8 (SFB 34) 和  
ALARM\_8P (SFB 35)

以下 SFB 不可确认: NOTIFY (SFB 36) 和 NOTIFY\_8P (SFB 31)

可确认的报警 SFC 包括: ALARM\_SQ (SFC 17) 和 ALARM\_DQ (SFC 107)

以下 SFC 不可确认: ALARM\_S (SFC 18) 和 ALARM\_D (SFC 108)

- 诊断报警

系统诊断 (ID 0x1000-0x79FF、0xC000-0xFFFF、0xF900-0xF9F)

用户诊断 (ID 0x8000-0xB9FF), 带 WR\_USMSG (SFC 52)

- 状态路径报警

在连接状态变化时由 OPC 报警和事件服务器生成。

#### 利用报警可以指定什么内容?

本部分介绍可在组态中为报警文本、源和区域进行的一些设置。

### 已组态报警文本

- 预设块相关报警（报警 SFB 和 SFC）输入：

在 STEP 7 的“特殊对象属性 > 消息”(Special Object Properties > Message....)

中输入相应背景数据块的报警文本

报警文本在对话框的“消息文本”(Message text) 列中输入。文本内可加入关联值。

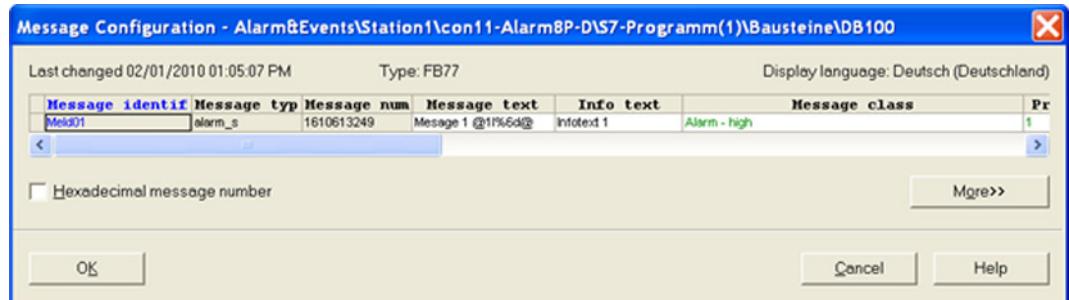


图 3-3 报警 SFC 的组态对话框，使用在 FB1 中调用的 ALARM\_SQ (SFC17)，实例 DB 为 DB1。

- 符号相关报警 (SCAN) 输入：

在 STEP 7 中报警符号的符号编辑器快捷菜单“特殊对象属性 > 消息...”(Special Object Properties > Message...) 中输入报警文本。对于这些符号，已选中“消息”(Message) 复选框。

报警文本在对话框的“消息文本”(Message text) 列中输入。

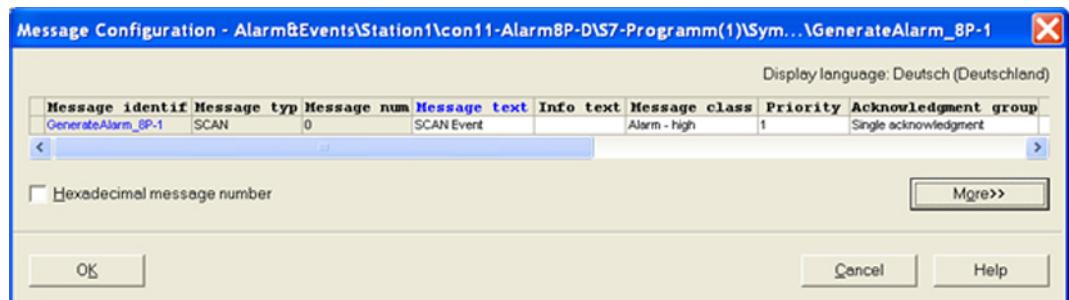


图 3-4 存储器位 3.1 的符号相关报警的组态对话框

### 3.1 用于 S7 通信的事件服务器

- 诊断报警输入:

在“进入的消息文本”(Incoming message text) 和“离开的消息文本”(Outgoing message text) 中输入用户诊断报警 WR\_UMSG (SFC52) 的报警文本进入和退出状态。

输入对话框位于 SIMATIC S7 站 SIMATIC Manager 的 S7

程序快捷菜单“特殊对象属性 > 消息”(Special Object Properties > Message...) 中。

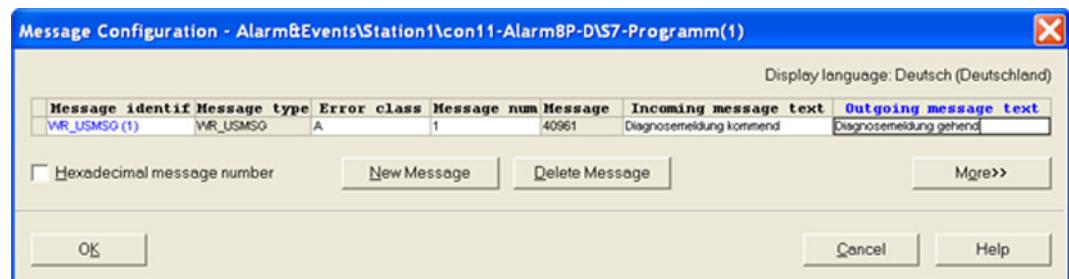


图 3-5 WR\_UMSG (SFC52) 用户诊断报警的组态示例

- 已组态源:

在输入报警文本的对话框中, 还可输入报警源。使用第一条附加文本 (数据块属性设置为 S7\_alarm\_ui=1 时的源) 作为源, 请参见图 3-6。

如果并未如此设置, 则使用在 STEP 7 中分配给报警文本的块路径名称。

例如 SIMATIC400(1)/CPU416.3DP/S7\_program(1)/DB10/DB100。

对于 StatepathAlarm, 使用组态的连接名称作为源。

- 已组态区域:

使用第二条附加文本 (属性设置为 S7\_Alarm\_ui=1 时的 OS 区域) 作为区域, 请参见图 3-6。

如果并未如此设置, 此处使用 STEP 7 项目中的路径名称。

对于 StatepathAlarm, 使用组态的连接名称作为区域。

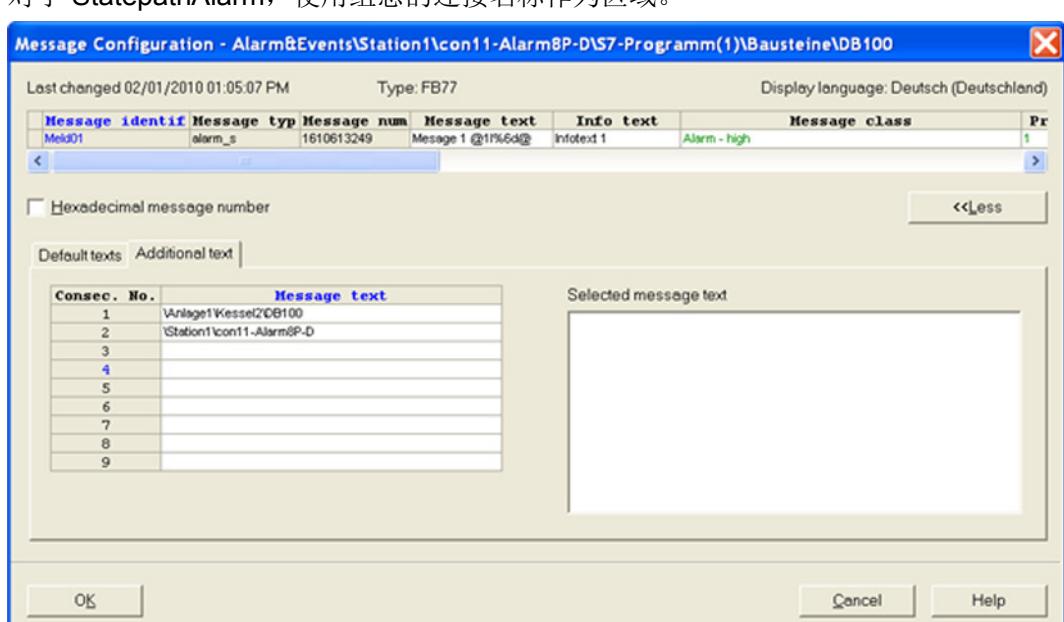


图 3-6 1. 附加文本 (源) 和第二条附加文本 (区域)

## 关联值如何嵌入报警?

报警和信息文本可以包含形式参数。

这些参数中的一部分可以替换为报警的格式化关联值。

有关可能的形式参数的结构和含义, 请参见 STEP 7 中的“帮助 > 插入关联值”(Help > Inserting associated values)。

但是, 用于 S7 通信的 OPC 服务器同样支持下表中由 \*) 标识的其它格式信息。

形式参数只替换为消息和信息文本中的格式化关联值。对于 S7 通信的 OPC 服务器, 不在附加文本中替换形式参数。

### 3.1 用于 S7 通信的事件服务器

形式参数具有以下结构: @<number><type><format>@

<number>	表示用于形式参数的关联值编号。																								
<type>	<p>描述关联值的类型。可能的值包括:</p> <p>Y - 字节 W - 字 D - 双字 I - 整型 D - 双整型 B - 布尔 C - Char R - 实数 &lt;空&gt; - 未指定类型:</p> <p>注意: 无法为“%t”、“%s”和“%Y”格式指定类型。 对于其它格式, 为关联值和格式选择合适的类型。 指定的关联值没有类型转换。</p>																								
<format>	<p>描述文本中关联值的格式。可能的格式如下:</p> <table border="1"> <tr> <td>%[i]X</td><td>以十六进制表示大写字母关联值。</td></tr> <tr> <td>%[i]x</td><td>以十六进制表示小写字母关联值。 *)</td></tr> <tr> <td>%[i]u</td><td>表示为无符号十进制数。</td></tr> <tr> <td>%[i]d</td><td>表示为有符号十进制数。</td></tr> <tr> <td>%[i]i</td><td>表示为有符号十进制数。 *)</td></tr> <tr> <td>%[i]b</td><td>表示为二进制数。</td></tr> <tr> <td>%[i].[y]f</td><td>表示为有 y 个小数位的定点数。</td></tr> <tr> <td>%[i].[y]e</td><td>表示为有 y 个小数位的浮点数, 采用小写字母指数记数法。 *)</td></tr> <tr> <td>%[i].[y]E</td><td>表示为有 y 个小数位的浮点数, 采用大写字母指数记数法。 *)</td></tr> <tr> <td>%[i].[y]g</td><td>表示为有 y 个小数位的定点数或浮点数 (小写字母)。 *)</td></tr> <tr> <td>%[i].[y]G</td><td>表示为有 y 个小数位的定点数或浮点数 (大写字母)。 *)</td></tr> <tr> <td>%[i]s</td><td>关联值解释为以 0 结尾的字符串。</td></tr> </table>	%[i]X	以十六进制表示大写字母关联值。	%[i]x	以十六进制表示小写字母关联值。 *)	%[i]u	表示为无符号十进制数。	%[i]d	表示为有符号十进制数。	%[i]i	表示为有符号十进制数。 *)	%[i]b	表示为二进制数。	%[i].[y]f	表示为有 y 个小数位的定点数。	%[i].[y]e	表示为有 y 个小数位的浮点数, 采用小写字母指数记数法。 *)	%[i].[y]E	表示为有 y 个小数位的浮点数, 采用大写字母指数记数法。 *)	%[i].[y]g	表示为有 y 个小数位的定点数或浮点数 (小写字母)。 *)	%[i].[y]G	表示为有 y 个小数位的定点数或浮点数 (大写字母)。 *)	%[i]s	关联值解释为以 0 结尾的字符串。
%[i]X	以十六进制表示大写字母关联值。																								
%[i]x	以十六进制表示小写字母关联值。 *)																								
%[i]u	表示为无符号十进制数。																								
%[i]d	表示为有符号十进制数。																								
%[i]i	表示为有符号十进制数。 *)																								
%[i]b	表示为二进制数。																								
%[i].[y]f	表示为有 y 个小数位的定点数。																								
%[i].[y]e	表示为有 y 个小数位的浮点数, 采用小写字母指数记数法。 *)																								
%[i].[y]E	表示为有 y 个小数位的浮点数, 采用大写字母指数记数法。 *)																								
%[i].[y]g	表示为有 y 个小数位的定点数或浮点数 (小写字母)。 *)																								
%[i].[y]G	表示为有 y 个小数位的定点数或浮点数 (大写字母)。 *)																								
%[i]s	关联值解释为以 0 结尾的字符串。																								

	%t#<file>	关联值解释为文本库 <file> 中的索引，从文本库插入文本。
	%Y[locale] [#DT format]	关联值解释为 DATE_AND_TIME，采用 DT 格式（考虑区域设置）。*) “locale”表示区域设置名称，例如 German、deu 或 us。如果缺少此设置，将采用当前区域设置。 DT 格式与标准函数 strftime() 的格式规范一致，外加 %s 表示三位毫秒输出这一扩展。如果未指定 DT 格式则使用 %c，相当于日期和时间表示。

方括号 [] 中的信息可选。值“i”代表要用于表示的字符数。

在形式参数外部替换以下字符串：

- \n 替换为字符 <new line> (0x0a)。
- \r 替换为字符 <CR> (0x0d)。
- \t 替换为制表符 (0x09)。
- \" 替换为双引号 (0x22)。
- \@ 替换为 @ 字符 (0x64)。这意味着该字符不引入任何形式参数。

### 3.1 用于 S7 通信的事件服务器

#### 3.1.7 如何在命名空间内浏览和过滤源信息、源和条件?

“IOPC EventAreaBrowser”接口为浏览源区域、源和条件提供方法。

可使用过滤器功能根据子区域进行过滤。

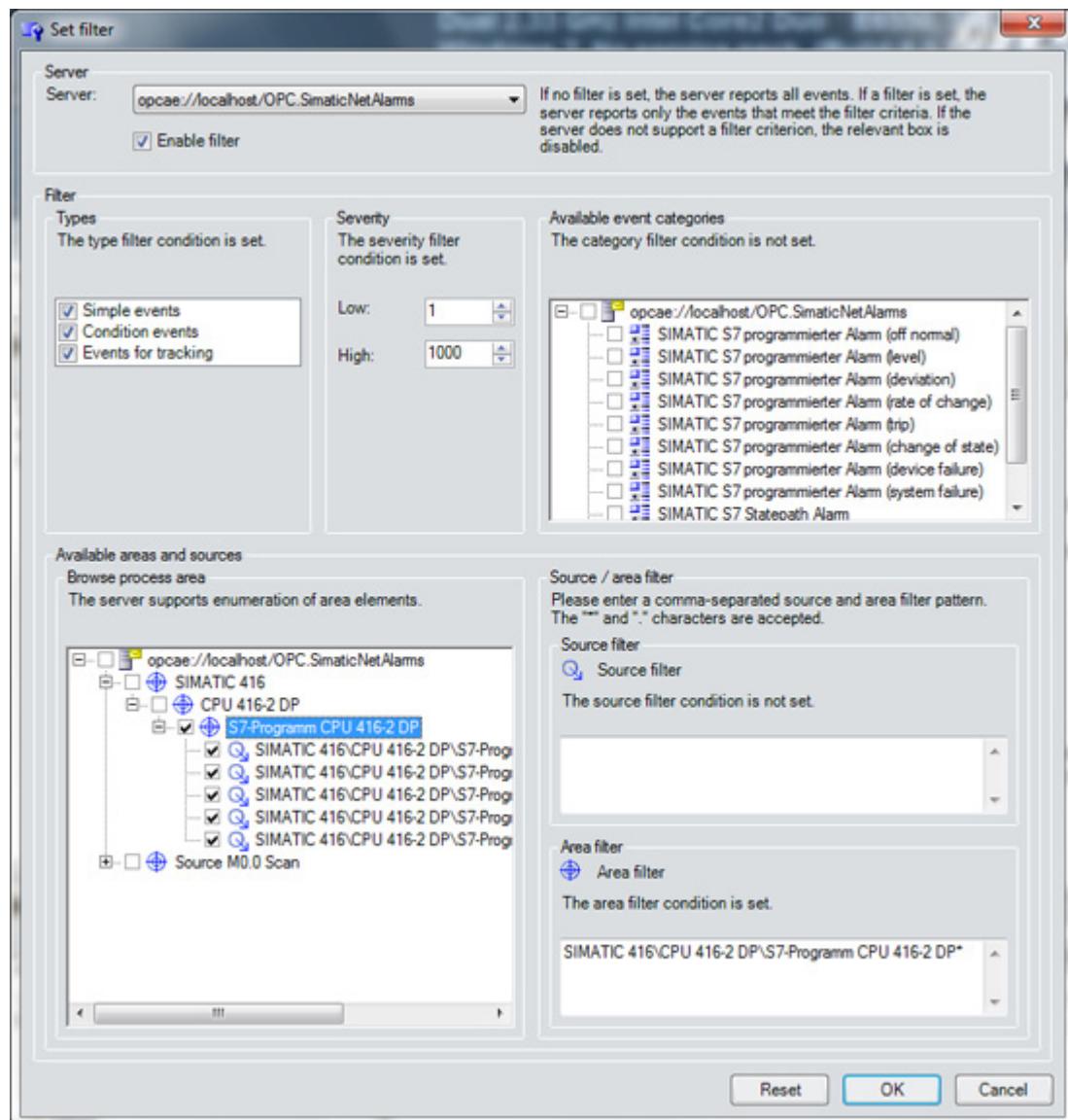


图 3-7 OPC Scout V10 中的过滤器对话框

要同时包括更高级别的区域，必须将字符“\*”附加到过滤器字符串结尾。

例如，使用“SIMATIC 400(1)"/CPU 416-3DP\*”过滤器获得 CPU 416 的所有报警。

## 3.1 用于 S7 通信的事件服务器

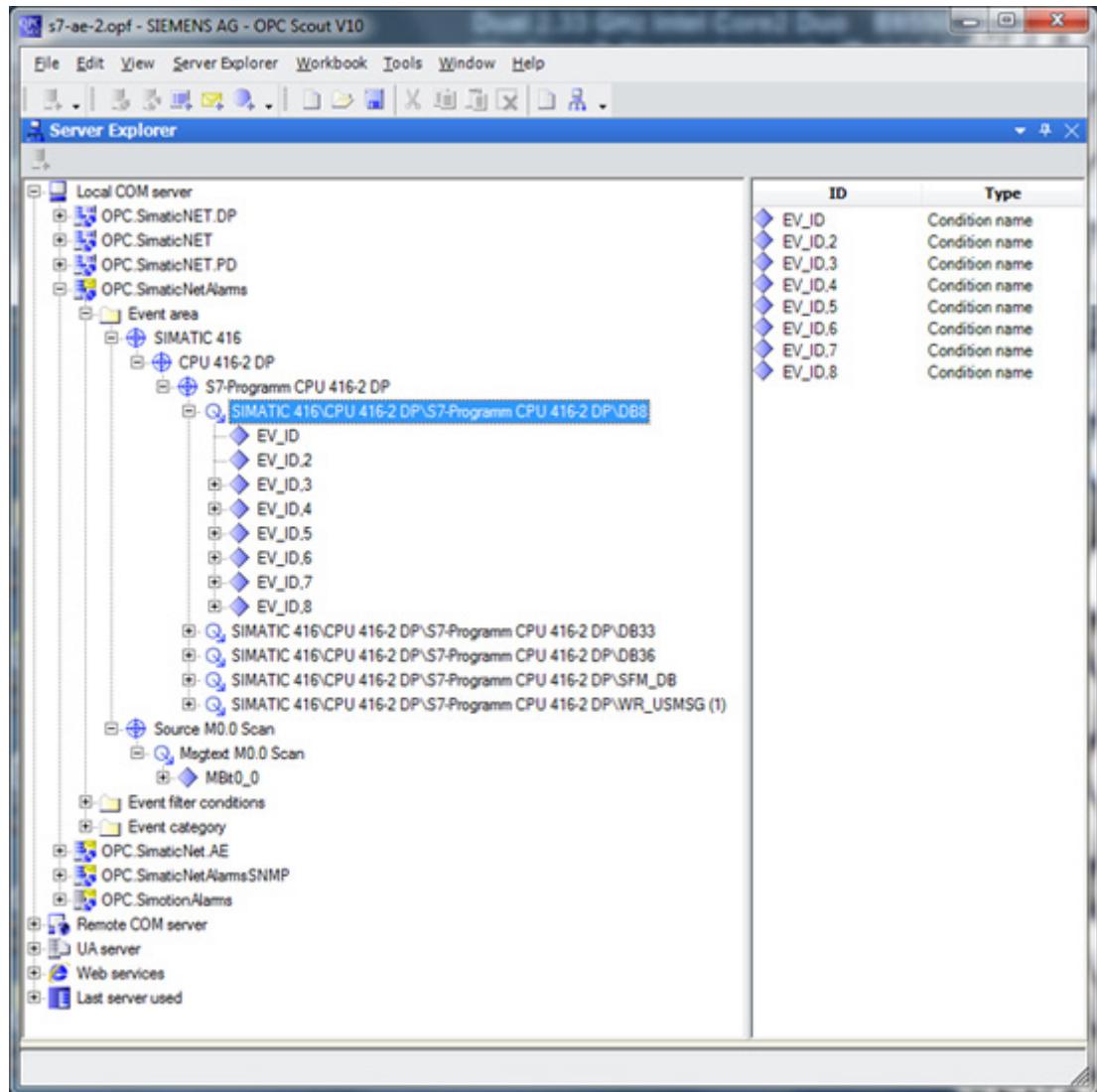


图 3-8 CPU 416 的源命名空间。可使用第一条附加文本对源进行专门组态。

## 3.1 用于 S7 通信的事件服务器

## 3.1.8 将 STEP 7 组态值映射到 OPC S7 报警和事件参数

下表显示组态数据（如“EventID”）从 STEP 7 到 OPC A&E 参数（如“条件”）的映射：

组态的参数	OPC 值	含义
EventID ⇒ 条件 ⇒ 组态的 EventID 映射到“条件”	“HIHI” “HI”（默认）	“HIHI”状态 “HI”状态
EVENTCATEGORY_CAT_LEVEL	“LO”	“LO”状态
EVENTCATEGORY_CAT_DEVIATION	“LOLO”	“LOLO”状态
EVENTCATEGORY_CAT_ROC		用户组态的 EventID 只能采用四个可能值中的一个。
EventID ⇒ 条件 EVENTCATEGORY_CAT_OFFNORMAL	“CFN”	“CFN”状态
EventID ⇒ 条件 EVENTCATEGORY_CAT_TRIP	“TRIP”	“TRIP”状态
EventID ⇒ 条件 EVENTCATEGORY_CAT_COS	“COS”	“COS”状态
EventID ⇒ 条件 EVENTCATEGORY_CAT_DEVICEFAILURE EVENTCATEGORY_CAT_SYSTEMFAILURE	“FAILURE”	“FAILURE”状态
来源/源 ⇒ 源	<connectionname>/ ALARM"<event ID> {,<subevent ID>} 或 <connectionname>/ STATEPATH 例如“s7_conn1/ALARM10,5” 例如“s7_conn1/STATEPATH”	连接名称与 EventID 或 STATEPATH 的唯一组合  备选： 组态的源，与语言相关。 源必须唯一！
消息权重 ⇒ 严重程度	1 ... 1000 请参见下表	组态或报警数据的严重 程度

组态的参数	OPC 值	含义
消息类别 ⇒ 类别	40	类别 CAT_LEVEL
	41	类别 CAT_DEVIATION
	42	类别 CAT_ROC
	43	类别
	44	CAT_OFFNORMAL
	45	类别 CAT_TRIP
	46	类别 CAT_COS
	47	类别 CAT_DEVICEFAILURE
		类别 CAT_SYSTEMFAILUR E
时间戳的已组态源 ⇒ 时间戳	00.00.0000 00:00:00.0000	时间戳 (CPU、CPU + OFFSET、PCtime)

优先级：优先级按下表转换为 OPC 报警严重程度：

S7 优先级	OPC 报警严重程度
0	1
1	63
2	125
3	188
4	250
5	313
6	375
7	438
8	500
9	563
10	625
11	688
12	750

### 3.1 用于 S7 通信的事件服务器

S7 优先级	OPC 报警严重程度
13	813
14	875
15	938
16	1000

#### 3.1.9 S7 演示报警

演示连接使命名空间内的数据块对象可用于数据访问。演示连接旨在使您熟悉 SIMATIC NET OPC 系统，它可通过组态激活。

还可以仿真 S7 消息。

可通过设置和复位演示数据块“db20”首字节的各个位来触发这些消息。

##### COM DA S7

ItemID:

S7: [DEMO] DB20, X0.0

S7: [DEMO] DB20, X0.1

...

S7: [DEMO] DB20, X0.7

##### OPC UA S7

NameSpaceID

S7:

NodeId:

...

DEMO.db20.0, x7

将位设置从“False”更改为“True”将触发 OPC

报警进入状态，将位从“True”复位为“False”将触发 OPC 报警退出状态。

演示数据块“db20”第三个字节中相应的位设置为“True”时，隐式确认报警：

S7: [DEMO] DB20, X2.0 ... 或

S7: DEMO.db20.2,x0 ...

可通过将这些位复位为“False”来设置可确认的报警。

通过数据访问触发报警时生成：

- 对于 S7 通信的 OPC 报警和事件  
类别 EVENTCATEGORY\_S7OFFNORMAL = 43 的条件报警
- 对于 S7 通信的 OPC UA 报警和条件  
EventType = Nodeld(ns=S7TYPES:, i=43) 的 UA 报警  
-> S7OffNormalAlarmType 的引用 Nodeld

## 3.2 SNMP 通信简单事件服务器

### SNMP 陷阱

陷阱是无需 OPC 服务器请求便可向其发送的消息。许多 SNMP 兼容设备都有七种通用陷阱。在 MIB 文件中同样介绍了设备特定陷阱。

---

#### 说明

必须使用 MIB 编译程序（STEP 7 的一部分）集成设备特定 MIB。

必须在相关设备上激活陷阱，且必须组态陷阱的目标站（这里是指：SNMP OPC 服务器）。

---

### 通用陷阱

#### warmStart

如果计算机已建立网络连接，将其重新启动  
设备暖启动后发送。

#### coldStart

如果计算机未建立网络连接，将其启动  
设备冷启动后发送。

#### linkDown

设备连接终止后发送。

#### linkUp

设备连接建立后发送。

## 3.2 SNMP 通信简单事件服务器

### **authenticationFailure**

设备受到未授权访问后发送。

### **egpNeighborLoss**

设备的 EGP 邻站（EGP = 外部网关协议）未工作。

外部网关协议用于在两个相邻网关主机之间交换路由信息。

### **enterpriseSpecific**

发送设备特定的陷阱后发送。

## 转发陷阱

陷阱以所谓的“简单事件”形式转发到 OPC 报警和事件服务器。

发生频率和陷阱描述也可用作 OPC DA 数据项。

### 要求

- 必须在 SNMP 兼容设备上输入装有 OPC SNMP 服务器的 PC 的 IP 地址作为陷阱接收方，且必须激活陷阱。  
使用设备特定组态工具进行转换（请参见手册“调试 PC 站”；基于 Web 的 OSM/ESM/SCALANCE 管理）。
- 必须安装 Windows SNMP 服务。

OPC 报警服务名称： OPC.SimaticNetAlarmsSNMP

## 报警和事件类别

以下报警和事件类别可用，下文有更详细的介绍：

类别编号	名称	说明
20	EVENTCATEGORY_SNMP_GENERICTRAP	SNMP 陷阱事件（通用陷阱）
21	EVENTCATEGORY_SNMP_SPECIFICTRAP	SNMP 陷阱事件（特定陷阱）

通用陷阱由各项标准指定，对所有设备都有相同意义。特定陷阱特定于设备。

设备配置文件会通知 SNMP OPC 服务器一个陷阱是通用还是特定陷阱。

类别“通用”或“特定”要放在术语“类别”的上下文中进行理解。

每个类别的报警和事件都可以发生在任何伙伴设备上（源）。

同一个伙伴设备中同一类别的不同报警和事件由各自 EventID 标识。

陷阱由通信伙伴触发，并按 SNMP 协议转发到 SIMATIC NET SNMP OPC 服务器。

通信负载较高时，SNMP 协议可能丢弃陷阱，所以 SIMATIC NET SNMP OPC 服务器不一定能接收到所有陷阱。这是 SNMP 协议的普遍限制。

陷阱没有状态也不被确认，因此 SIMATIC NET SNMP OPC 服务器不为陷阱管理状态机。

客户端通过调用 *OnEvent* 回调函数获取事件列表。SNMP OPC 事件服务器返回以下服务特定参数：

参数	含义
dwEventCategory	以下值可以用作事件类别： EVENTCATEGORY_SNMP_GENERICTRAP = 20 EVENTCATEGORY_SNMP_SPECIFICTRAP = 21
dwEventType	SNMP OPC 事件服务器仅支持以下类型： OPC_SIMPLE_EVENT 此常数值为 0x0001。
dwNumEventAttrs	消息提供的属性数取决于提供的关联值数。
dwSeverity	为错误严重程度返回预定义值。 可在组态数据库中为通信伙伴的各个消息编号修改此值。
pEventAttributes	此结构包含事件提供的属性。 属性还包括由伙伴设备提供的消息的关联值。
szMessage	设备配置文件中的陷阱事件名称： <名称文本> 例如 TrapPowerDown
szSource	作为消息源，SNMP OPC 事件服务器指定发出报告的 SNMP 设备的设备信息。这与 OPC 数据访问的 SNMP 变量的设备名称一致： SNMP:<设备> 例如 SNMP:\Switch33

## 3.3 S7 和 SNMP 通信的报警和事件服务器

除此事件外，同时还能以属性形式发送附加信息。SNMP 陷阱提供以下属性：

属性 ID 值	属性	含义
EVENT_ATTR_SNMP_ PCTIME 6100	VT_DATE	OPC 事件服务器收到报警的时间。
EVENT_ATTR_SNMP_ TIMESTAMP 6101	AsnTimeticks (VT_UI4)	在伙伴设备上生成消息的时间。
EVENT_ATTR_SNMP_ EVENTID 6102	LONG (VT_I4)	设备中的唯一陷阱编号
EVENT_ATTR_SNMP_ ENTERPRISE 6103	AsnObjectIdentifier (VT_BSTR)	设备特定陷阱
EVENT_ATTR_SNMP_ AGENTADDRESS 6104	AsnNetworkAddress (VT_BSTR)	代理的 IP 地址
EVENT_ATTR_SNMP_ SOURCEADDRESS 6105	AsnNetworkAddress (VT_BSTR)	陷阱源的 IP 地址
EVENT_ATTR_SNMP_ VARBINDINGS 6106	SnmpVarBindList (VT_ARRAY   VT_VARIANT)	已发送陷阱的变量绑定

## 3.3 S7 和 SNMP 通信的报警和事件服务器

### 3.3.1 SIMATIC NET 的报警和事件服务器

**哪些报警和事件服务器可用于 SIMATIC NET?**

类似于 OPC

数据访问 (“OPC.SimaticNET”服务器结合所有通信协议), “OPC.SimaticNET.AE”服务器也可用于 OPC 报警和事件, 这结合了 SNMP 和 S7 通信协议。  
这允许在一台服务器内接收 S7 报警和 SNMP 陷阱。

单独的报警和事件服务器 (如“OPC.SimaticNetAlarms”和“OPC.SimaticNetAlarmsSNMP”) 可以彼此并行使用。

### 3.3 S7 和 SNMP 通信的报警和事件服务器

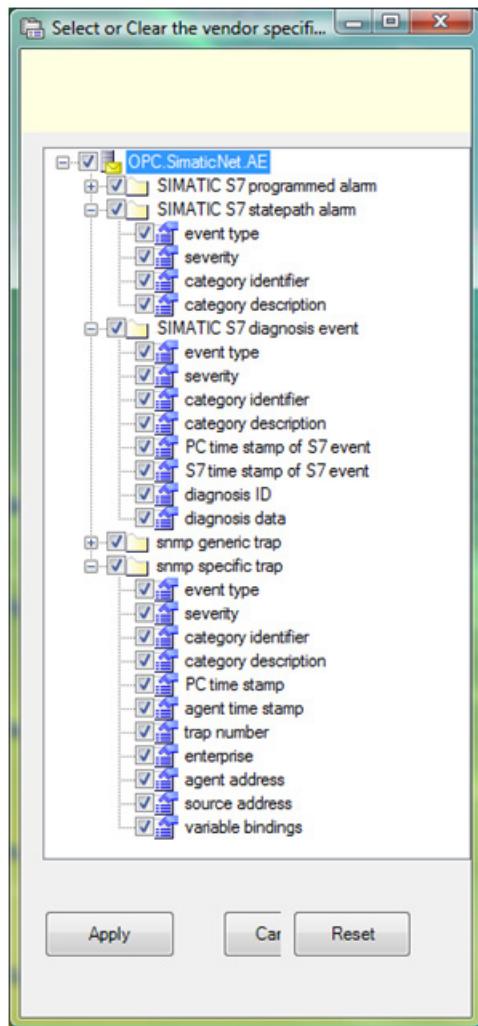


图 3-9 报警和事件服务器“OPC.SimaticNET.AE”；OPC Scout V10 中的视图

#### 3.3.2 服务器的名称 (ProgID) 是什么？

##### ProgID

SIMATIC NET OPC 报警和事件服务器的 ProgID 如下：

- OPC.SimaticNET.AE
- OPC.SimaticNETAlarms
- OPC.SimaticNETAlarmsSNMP

### 3.3.3 如何检测报警源的协议？

#### 如何查找报警源？

“IOPCEventAreaBrowser”界面可用于浏览可能的报警源。此界面支持 OPC 区域和源参数的过滤和查询方法。

返回以下固定区域和源前缀：

- \S7::
- \SNMP::

可按区域“\S7::”和“\SNMP::”进行过滤。

### 3.3 S7 和 SNMP 通信的报警和事件服务器

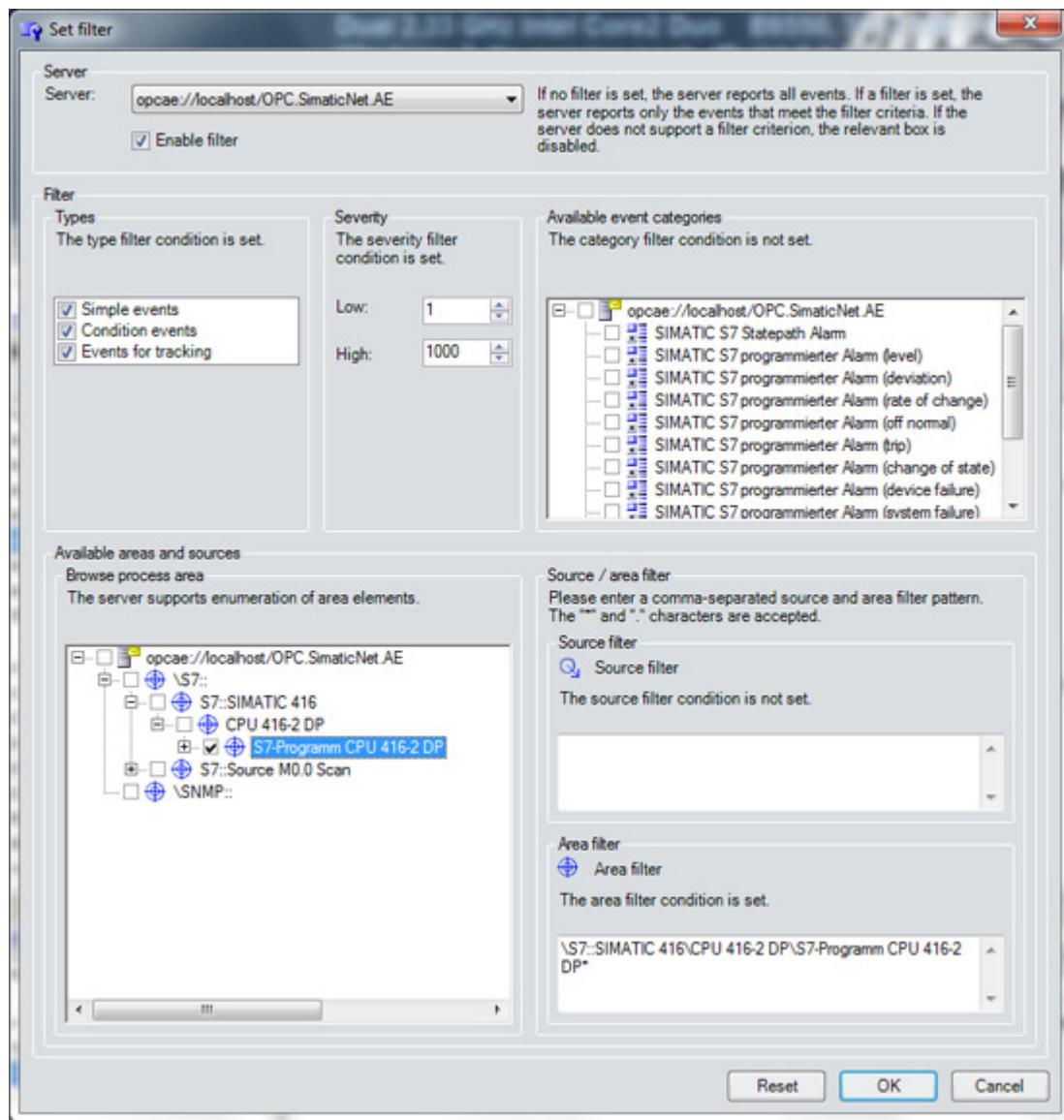


图 3-10 在 OPC Scout V10 中过滤报警源

### “OPC.SimaticNET.AE”服务器的特性

如果使用“OPC.SimaticNET.AE”服务器，在报警的源参数方面其行为有所不同：

- 除协议特定源参数外（如“\S7:\S7\_connection\_8”），还有区域前缀“\S7::”。这会生成以下源参数：  
“\S7::\S7:\S7\_connection\_8”
- 对于 S7 协议的可组态源文本“tank\_pressure”，它也可以显示为如下形式：  
“\S7::tank\_pressure”

在其它方面，“OPC.SimaticNET.AE”服务器与单独的“OPC.SimaticNetAlarms”和“OPC.SimaticNetAlarmsSNMP”服务器行为类似。



# 使用 OPC 服务器

本部分介绍在使用 OPC 服务器时为您提供的各种选项。

## 4.1 编程自动化接口

自动化接口是对自定义接口的补充。有了自动化接口，在对 OPC 进行编程时可便捷的使用现代化开发系统和脚本语言。

既有用于访问过程变量的自动化接口（数据访问），也有用于处理事件和报警的自动化接口（报警和事件）。

### 应用程序

可在想要基于办公应用程序或使用 **Visual Basic**

通过合理数目的变量和介质数据吞吐量创建应用程序时使用自动化接口。

### 4.1.1 对用于数据访问的自动化接口进行编程

对于数据访问，有一个简单的类模型，它可将接口及其方法进行分类。

自动化接口进一步细化了对自定义接口有效的类模型，以提供结构化开发系统（如 **Visual Basic**）的优势和选项。

---

#### 说明

用于 OPC 数据访问的 OPC 自动化接口已在 **Siemens** 提供的版本中发布，该版本基于 OPC 基金会所提供的版本，并更正了其中的错误。

---

#### 4.1.1.1 OPC 数据访问的对象模型都提供了什么？

数据访问类模型的类别包含以下对象：

- OPCServer
- OPCGroup
- OPCItem

## 4.1 编程自动化接口

可向自动化接口添加更多的对象。为了管理 OPCGroup 和 OPCItem 对象，还提供了单独的集合对象 OPCGroups 和 OPCItems。集合对象提供了用于管理所分配对象的功能。

还提供了一个包含浏览功能的 OPCBrowser 集合对象。

### 对象模型

下图说明了各个对象以及各个对象之间的关系。

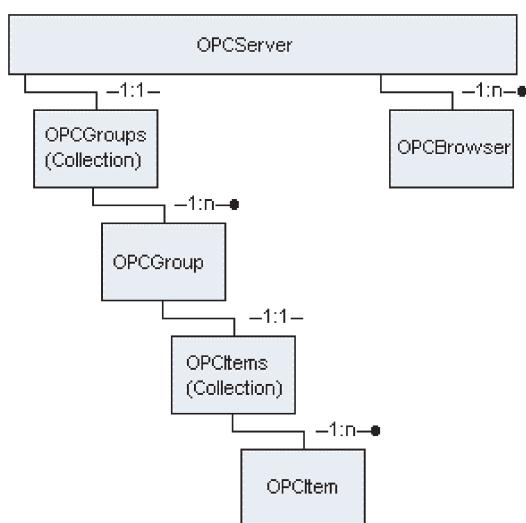


图 4-1 OPC 数据访问的对象模型

#### 4.1.1.2 编程时需记住的要点

- 使用自动化接口时，必须将可选参数作为变量传送。但在 Visual Basic 中，则应使用所需类型的可选变量声明可选参数。这样可确保变量包含正确的数据类型。
- 异步函数  
必须将 *IsSubscribed* 组特定属性设置为 *True*，以允许监视变量。
- 在 Visual Basic 中，必须使用补充 *withEvents* 声明用于接收事件的对象。

**示例：**

*Dim WithEvents MyOPCGroup As OPCGroup*

- 按照 OPC 规范，数组始终以索引 1 开始。在程序中可按如下方式对其进行定义：

```
Option Base 1 'ARRAYs are always Option Base 1
```

### 4.1.1.3 用于数据访问的自动化接口的对象

您将在后续部分看到一个有关数据访问的属性、方法和事件的列表。其中仅介绍了特定于 SIMATIC NET 的特性。有关这些属性、方法和事件的详细说明，请参见相关的 OPC 规范。

## 对象

用于数据访问的自动化接口存在以下对象：

### OPCServer 对象

OPC 服务器类的 OPCServer 对象由客户端创建。

OPCServer 对象的属性包含有关服务器的常规信息。在创建对象时，还可将 OPCGroups 集合对象创建为 OPCServer 对象的属性。

### OPCServer 的属性

下面有一个 OPCServer 对象属性列表。其中仅介绍了特定于 SIMATIC NET 的特性。

有关各属性的说明，请参见下面的 OPC 规范：

数据访问自动化接口

版本 2.02

1999 年 2 月 4 日

属性	含义
Bandwidth	返回服务器的带宽。SIMATIC NET 的 OPC 服务器不支持带宽。
BuildNumber	返回服务器的版本号。
ClientName	指定客户端的名称。ClientName 主要用于测试。
CurrentTime	返回当前时间 (UTC)。
LastUpdateTime	返回服务器上次向客户端发送数据的时间 (UTC)。
LocaleID	指定显示文本的语言。对于 SIMATIC NET，提供德语和英语。
MajorVersion	返回服务器的主版本号。
MinorVersion	返回服务器的次要版本号。
OPCGroups	指定 OPCGroup 对象的集合

## 4.1 编程自动化接口

属性	含义
PublicGroupNames	返回 OPC 服务器的公共组的名称。 SIMATIC NET 的 OPC 服务器不支持可选公共组。
ServerName	返回所连接 OPC 服务器的名称。
ServerNode	返回执行 OPC 服务器的网络节点的名称。
ServerState	返回服务器的状态。
StartTime	返回服务器启动时间 (UTC)。
VendorInfo	返回供应商信息。 SIMATIC NET 的 OPC 服务器将“SIMATIC NET OPC 服务器”作为供应商信息返回。

## OPCServer 的方法

下面列出了 OPCServer 对象的方法。其中仅介绍了特定于 SIMATIC NET 的特性。  
您将在以下 OPC 规范中找到方法的说明：

数据访问自动化接口

版本 2.02

1999 年 2 月 4 日

方法	含义
Connect	建立与 OPC 服务器的连接。例如，OPC 服务器的 <i>ProgID</i> 是“OPC.SimaticNET”。
CreateBrowser	创建 OPCBrowser 集合对象。
Disconnect	终止与 OPC 服务器的连接。 删除最后一个链接后，SIMATIC NET 的 OPC 服务器会终止与 OPC 客户端的所有通信连接。
GetErrorString	获取错误代码的错误消息。SIMATIC NET 的 OPC 服务器支持德语和英语错误消息。Windows 操作系统的错误消息以操作系统安装的语言显示。
GetItemProperties	返回所请求属性的值列表。
GetOPCServers	返回已注册 OPC 服务器的名称。例如，SIMATIC NET 的 OPC 服务器的名称为“OPC.SimaticNET”。
LookupItemIDs	返回与 <i>PropertyID</i> 相对应的 <i>ItemID</i> 的列表。SIMATIC NET 的 OPC 服务器不返回可显示为 <i>ItemID</i> 的 <i>PropertyID</i> 。

方法	含义
QueryAvailableLocaleIDs	返回可用语言代码。 SIMATIC NET 的 OPC 服务器支持德语和英语错误消息。 Windows 操作系统的错误消息以操作系统安装的语言显示。
QueryAvailableProperties	返回 OPC 数据项的属性和属性代码。

## OPCServer 的事件

下面将对 OPCServer 对象的事件进行说明。其中仅介绍了特定于 SIMATIC NET 的特性。有关事件的说明，请参见下面的 OPC 规范：

数据访问自动化接口

版本 2.02

1999 年 2 月 4 日

事件	含义
ServerShutDown	OPC 服务器关闭时触发此事件。 组态程序发送关闭命令时，或者 PC 站接收新的组态数据时，SIMATIC NET 的 OPC 服务器触发此事件。

## 集合对象 OPCBrowser

使用 OPCBrowser 集合对象可以浏览 OPC 服务器的命名空间。

OPCBrowser 类的对象由 OPCServer 对象的 *CreateBrowser* 方法创建。  
可为一个服务器创建多个 OPCBrowser 对象。

## OPCBrowser 的属性

下面有一个 OPCBrowser 集合对象属性列表。其中仅介绍了特定于 SIMATIC NET 的特性。有关各属性的说明，请参见下面的 OPC 规范：

## 4.1 编程自动化接口

数据访问自动化接口

版本 2.02

1999 年 2 月 4 日

属性	含义
AccessRights	决定 <i>ShowLeafs</i> 方法的访问权限。
Count	返回条目数。
CurrentPosition	返回命名空间树中的当前位置。
DataType	决定 <i>ShowLeafs</i> 方法的数据类型。
Filter	决定 <i>ShowLeafs</i> 和 <i>ShowBranches</i> 方法的过滤器。
Organization	返回命名空间的组织结构。 SIMATIC NET 的 OPC 服务器的命名空间为分层结构。

## OPCBrowser 的方法

下面有一个 OPCBrowser 集合对象的方法列表。其中仅介绍了特定于 SIMATIC NET 的特性。您将在以下 OPC 规范中找到方法的说明：

数据访问自动化接口

版本 2.02

1999 年 2 月 4 日

方法	含义
GetAccessPaths	获取 ItemID 的访问路径。 对于 SIMATIC NET 的 OPC 服务器，不应使用 AccessPath 参数。
GetItemID	如果适用，此方法将为 OPC 服务器的命名空间中的元素返回 ItemID。
Item	指定条目名称
MoveDown	将命名空间中的当前位置向下移动一个层级
MoveTo	将命名空间中的当前位置移动到指定位置
MoveToRoot	将命名空间中的当前位置移动到根级
MoveUp	将命名空间中的当前位置向上移动一个层级
ShowBranches	指定当前浏览位置的分支名称
ShowLeafs	指定当前浏览位置的叶名称

## 集合对象 OPCGroups

**OPCGroups** 集合对象是 **OPCGroup** 对象的集合，用于创建和管理 OPC 组。

**OPCGroups** 的标准属性可指定所有新创建 OPC 组的标准值。

如果 OPC 服务器对象成功执行 *Connect* 调用，则会将 **OPCGroups** 集合对象作为 OPC 服务器对象的属性进行创建。

---

### 说明

SIMATIC NET 的 OPC 服务器不支持可选公共组。

---

## OPCGroups 的属性

下面有一个 **OPCGroups** 集合对象属性列表。其中仅介绍了特定于 SIMATIC NET 的特性。有关各属性的说明，请参见下面的 OPC 规范：

数据访问自动化接口

版本 2.02

1999 年 2 月 4 日

属性	含义
Count	返回组数。
DefaultGroupDeadband	为新生成的 <b>OPCGroup</b> 对象指定 <i>Deadband</i> 的初始值。
DefaultGroupIsActive	为新生成的 <b>OPCGroup</b> 对象指定 <i>ActiveState</i> 的初始值。
DefaultGroupLocaleID	为新生成的 <b>OPCGroup</b> 对象指定 <i>LocaleID</i> 的初始值。
DefaultGroupTimeBias	为新生成的 <b>OPCGroup</b> 对象指定 <i>TimeBias</i> 的初始值。
DefaultGroupUpdateRate	为新生成的 <b>OPCGroup</b> 对象指定 <i>UpdateRate</i> 的初始值。
Parent	返回相应 <b>OPCServer</b> 对象的引用。

## OPCGroups 的方法

下面有一个 **OPCGroups** 集合对象的方法列表。其中仅介绍了特定于 SIMATIC NET 的特性。您将在以下 OPC 规范中找到方法的说明：

#### 4.1 编程自动化接口

数据访问自动化接口

版本 2.02

1999 年 2 月 4 日

方法	含义
Add	创建新的 OPCGroup 对象，并将其添加至集合。
GetOPCGroup	指定对 OPCGroup 对象的名称或服务器句柄的引用。
Item	返回对集合中索引对象的引用。
Remove	删除服务器的一个组。
RemoveAll	删除服务器的所有组和项。

#### OPCGroups 的事件

下面将对 OPCGroups 集合对象的事件进行说明。其中仅介绍了 SIMATIC NET 的特性。有关事件的详细说明，请参见下面的 OPC 规范：

数据访问自动化接口

版本 2.02

1999 年 2 月 4 日

事件	含义
GlobalDataChange	通知值更改以及所有激活组的激活数据项的状态。

#### OPCGroup 对象

OPC 组类可管理各个过程变量，即 OPC 数据项。使用 OPCGroup 对象，客户端可以根据语义方面的考虑将 OPC 数据项放入某一组，然后将该组作为单个单元进行处理。

监视变量以及读取和写入访问权限是组特定内容。

例如，可通过调用一个函数激活监视一个组中的所有 OPC 数据项。

例如，应将显示在操作员控制和监视站上同一屏幕中的所有过程变量放在同一个组中，然后在打开屏幕时激活监视变量。

---

#### 说明

SIMATIC NET 的 OPC 服务器不支持可选公共组。

---

## OPCGroup 的属性

下面有一个 OPCGroup 对象属性列表。其中仅介绍了特定于 SIMATIC NET 的特性。  
有关各属性的说明，请参见下面的 OPC 规范：

数据访问自动化接口

版本 2.02

1999 年 2 月 4 日

属性	含义
ClientHandle	指定用于本地化数据的句柄。
DeadBand	指定发生值更改时不会发出通知的带宽。
IsActive	指定组的激活状态。 如果想要监视该组的变量，请确保将 <i>IsActive</i> 设置为 <i>True</i> 。
IsPublic	指示该组是否为公共组。 SIMATIC NET 的 OPC 服务器不支持可选公共组。
IsSubscribed	指示是否监视组变量。 如果想要监视该组的变量，请确保将 <i>IsSubscribed</i> 设置为 <i>True</i> 。
LocaleID	指定服务器返回的文本字符串的语言。
Name	指定组的名称。
OPCItems	指定用于管理组中数据项的集合对象。
Parent	返回相应 OPCServer 对象的引用。
ServerHandle	返回组的唯一句柄。
TimeBias	指定用于将时间戳转换为本地时间的时间偏移。
UpdateRate	指定值更改或数据项状态更改时通知客户端的时间间隔。

## OPCGroup 的方法

下面有一个 OPCGroup 对象的方法列表。其中仅介绍了特定于 SIMATIC NET 的特性。  
您将在以下 OPC 规范中找到方法的说明：

数据访问自动化接口

版本 2.02

1999 年 2 月 4 日

方法	含义
AsyncCancel	取消异步作业。
AsyncRead	发送异步读取命令。
AsyncRefresh	使用缓存中的当前值为每个激活的 OPC 数据项创建一个事件。
AsyncWrite	发送异步写入命令。
SyncRead	启动同步读取某一组中一个或多个数据项的值、状态信息和时间戳。
SyncWrite	启动同步写入某一组中一个或多个数据项的值。

## OPCGroup 的事件

OPC 自动化接口使用事件返回激活数据项的值更改以及异步操作的结果。

Visual Basic 中用于接收事件的对象必须使用 *withEvents* 声明。

下面有一个 OPCGroup 对象的事件列表。其中仅介绍了特定于 SIMATIC NET 的特性。  
有关事件的说明，请参见下面的 OPC 规范：

数据访问自动化接口

版本 2.02

1999 年 2 月 4 日

事件	含义
AsyncCancelComplete	“取消”作业完成后触发此事件。
AsyncReadComplete	“读取”作业完成后触发此事件。
AsyncWriteComplete	“写入”作业完成后触发此事件。
DataChange	一个或多个数据项中的值发生更改或质量发生更改时触发此事件。

## 集合对象 OPCItems

OPCItems 集合对象是 OPCItem 对象的集合，用于创建和管理 OPC 数据项。

OPCItems 的属性可用于指定所创建的所有新 OPC 数据项的标准值。

创建 **OPCGroup** 对象后会自动创建 **OPCItems** 集合对象。 **OPCItems** 始终作为组的属性存在并用于监视 OPC 变量。

## OPCItems 的属性

下面有一个 **OPCItems** 对象属性列表。 其中仅介绍了特定于 SIMATIC NET 的特性。  
有关各属性的说明，请参见下面的 OPC 规范：

数据访问自动化接口

版本 2.02

1999 年 2 月 4 日

属性	含义
Count	返回组中数据项的数目。
DefaultAccessPath	为新添加的 OPC 数据项指定 <i>AccessPath</i> 的初始值。 对于 SIMATIC NET 的 OPC 服务器来说， <i>DefaultAccessPath</i> 应为空。
DefaultIsActive	为新添加的 OPC 数据项指定 <i>ActiveState</i> 的初始值。
DefaultRequestedDataType	为新添加的 OPC 数据项指定 <i>RequestedDataType</i> 的初始值。
Parent	返回相应 <b>OPCGroup</b> 对象的引用。

## OPCItems 的方法

下面有一个 **OPCItems** 对象的方法列表。 其中仅介绍了特定于 SIMATIC NET 的特性。  
您将在以下 OPC 规范中找到方法的说明：

数据访问自动化接口

版本 2.02

1999 年 2 月 4 日

方法	含义
AddItem	向集合对象添加一个新的 OPC 数据项。
AddItems	向集合对象添加若干个 OPC 数据项。
GetOPCItem	指定对由 <i>AddItem</i> 创建的服务器句柄的引用。
Item	指定对集合数据项的引用。
Remove	从组中删除一个或多个数据项。

方法	含义
<b>SetActive</b>	设置组中一个或多个数据项的激活状态。
<b>SetClientHandles</b>	更改一个或多个数据项的客户端句柄。
<b>SetDataTypes</b>	设置一个或多个数据项的数据类型。
<b>Validate</b>	检查一个或多个 OPC 数据项的有效性。

## OPCItem 对象

OPC 数据项类的对象表示过程变量，例如可编程控制器的输入模块。过程变量是过程 I/O 的可写入和/或可读取数据项，如罐温度。

每个过程变量都与一个值、质量或时间戳关联。

## OPCItem 的属性

下面有一个 OPCItem 对象属性列表。其中仅介绍了特定于 SIMATIC NET 的特性。  
有关各属性的说明，请参见下面的 OPC 规范：

数据访问自动化接口

版本 2.02

1999 年 2 月 4 日

属性	含义
<b>AccessPath</b>	返回数据项的访问路径。对于 SIMATIC NET 的 OPC 服务器来说，AccessPath 应为空。
<b>AccessRights</b>	返回变量的访问权限。
<b>CanonicalDataType</b>	返回数据项的原始数据类型。
<b>ClientHandle</b>	指定用于将过程变量更加简单地分配到客户端内部数据结构的句柄。
<b>EUInfo</b>	返回有关值单位的信息（可选）。SIMATIC NET 的 OPC 服务器不支持单位（工程单位）。
<b>EUType</b>	返回返回值的单位（可选）。SIMATIC NET 的 OPC 服务器不支持单位（工程单位）。
<b>IsActive</b>	指定是否为数据项生成通知事件。
<b>ItemID</b>	返回数据项的唯一名称。
<b>Parent</b>	返回对父 OPCGroup 对象的引用。

属性	含义
Quality	返回上次所读取值的质量。
RequestedDataType	为数据项的值指定所请求的数据类型。
ServerHandle	返回用于标识数据项的服务器句柄。
TimeStamp	返回获取上一个值的时间。
值	返回数据项的上一个有效值。

## OPCItem 的方法

下面有一个 **OPCItem** 对象的方法列表。其中仅介绍了特定于 **SIMATIC NET** 的特性。  
您将在以下 OPC 规范中找到方法的说明：

数据访问自动化接口

版本 2.02

1999 年 2 月 4 日

方法	含义
Read	同步读取变量的值、质量和/或时间戳。
Write	同步设置变量的值。

## 4.1.2 对用于报警和事件的自动化接口进行编程

对于报警和事件，有一个简单的类模型，它可将接口及其方法进行分类。

自动化接口进一步细化了对自定义接口有效的类模型，以提供结构化开发系统（如 Visual Basic）的优势和选项。

### 4.1.2.1 OPC 报警和事件的对象模型都提供了什么？

报警和事件的类模型的类包含以下对象：

- OPCEventServer
- OPCEventSubscriptions
- OPCEventSubscription
- OPCEventAreaBrowsers
- OPCEventAreaBrowser

## 4.1 编程自动化接口

- OPCEvents
- OPCEvent
- OPCEventCondition
- OPCEventSubConditions
- OPCEventSubCondition

可向自动化接口添加更多的对象。

### 说明

因为 SIMATIC NET 的事件服务器是简单事件服务器，所以不支持以下对象：  
OPCEventAreaBrowsers、OPCEventAreaBrowser、OPCEventCondition、OPCEventSubConditions、OPCEventSubCondition。

## 对象模型

下图说明了各个对象以及各个对象之间的关系。

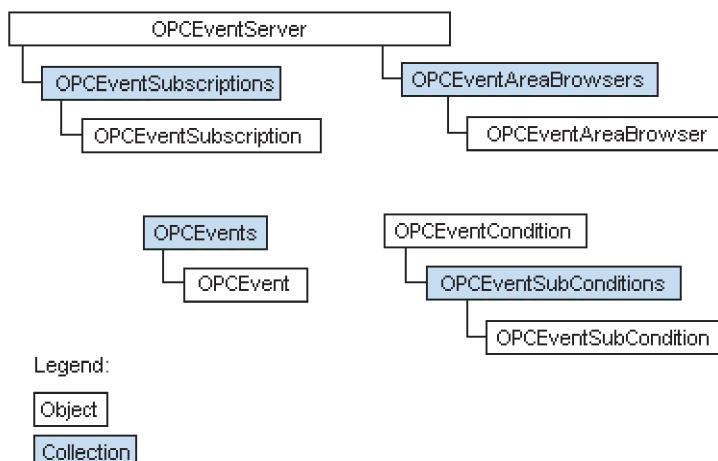


图 4-2 OPC 报警和事件的对象模型

### 4.1.2.2 编程时需记住的要点

使用自动化接口时，必须将可选参数作为变量传送。但在 Visual Basic 中，应使用目标类型声明可选参数。这样可确保变量包含正确的数据类型。

- 异步函数  
必须将 *IsSubscribed* 组特定属性设置为 *True*，以允许监视变量。
- 在 Visual Basic 中，必须使用补充 *withEvents* 声明用于接收事件的对象。  
示例：  
`Dim WithEvents MyOPCGroup As OPCGroup`
- 按照 OPC 规范，数组始终以索引 1 开始。在程序中可按如下方式对其进行定义：  
*Option Base 1 'ARRAYs are always Option Base 1*

### 4.1.2.3 用于报警和事件的自动化接口的对象

您将在后续部分看到一个有关报警和事件的属性、方法和事件的列表。

其中仅介绍了特定于 SIMATIC NET 的特性。

有关这些属性、方法和事件的详细说明，请参见相关的 OPC 规范。

## 对象

用于报警和事件的自动化接口存在以下对象：

### OPCEventServer 对象

OPC 事件服务器类的 OPCEventServer 对象由客户端创建。客户端必须先创建 OPCEventServer，然后才能访问报警和事件的其它对象。

OPCEventServer 对象的属性包含有关事件服务器的常规信息。

在创建对象的同时还创建了 OPCEventServerSubscription 集合对象。

OPCEventServer 通过 *Connect* 方法连接到事件服务器。

### OPCEventServer 的属性

下面有一个 OPCEventServer 对象属性列表。其中仅介绍了特定于 SIMATIC NET 的特性。有关各属性的说明，请参见下面的 OPC 规范：

报警和事件自动化接口标准

版本 1.01

1999 年 12 月 15 日

属性	含义
BuildNumber	返回服务器的版本号。
ClientName	指定客户端的名称。
CurrentTime	返回当前时间 (UTC)。
FiltersByArea	指示服务器是否能够按照区域进行过滤。 OPC 事件服务器不支持区域。
FiltersByCategory	指示服务器是否能够按照事件类别进行过滤。
FiltersByEventType	指示服务器是否能够按照事件类型进行过滤。
FiltersBySeverity	指示服务器是否能够按照事件的严重程度进行过滤。
FiltersBySource	指示服务器是否能够按照来源进行过滤。
LastUpdateTime	返回服务器上次向客户端发送数据的时间 (UTC)。
LocaleID	指定显示文本的语言。
MajorVersion	返回服务器的主版本号。
MinorVersion	返回服务器的次要版本号。
OPCEventAreaBrowsers	指定 <i>OPCAutoEventAreaBrowser</i> 对象的集合。 SIMATIC NET 的 OPC 事件服务器不支持 <i>OPCEventAreaBrowsers</i> 。
OPCEventSubscriptions	指定 <i>OPCEventSubscription</i> 对象的集合。
ServerName	返回所连接 OPC 服务器的名称。
ServerNode	返回执行 OPC 服务器的网络节点的名称。
ServerState	返回服务器状态。
StartTime	返回 OPC 服务器启动时间。
VendorInfo	返回供应商信息。

## OPCEventServer 的方法

下面有一个 OPCEventServer 对象的方法列表。其中仅介绍了特定于 SIMATIC NET 的特性。您将在以下 OPC 规范中找到方法的说明：

报警和事件自动化接口标准

版本 1.01

1999 年 12 月 15 日

方法	含义
AckCondition	确认事件服务器上的一个或多个条件。
Connect	建立与 OPC 服务器的连接。
Disconnect	终止与 OPC 服务器的连接。
EnableConditionsByArea	将所有区域的所有条件设置为已定义的状态。
EnableConditionBySrc	将所有来源的所有条件设置为已定义的状态。
GetConditionState	获取条件的当前状态信息。
GetErrorString	将错误代码转换为错误文本。
GetOPCEventServers	返回所有已注册事件服务器的名称。
QueryAvailableLocaleIDs	返回可用语言代码。
QueryConditionNames	返回对特定事件类别有效的条件名称。
QueryEventAttributes	返回供应商特定的属性
QueryEventCategories	返回受事件服务器支持的类别。
QuerySourceConditions	返回与特定来源相链接的条件的名称。
QuerySubConditionNames	返回与特定来源相链接的子条件的名称。

## OPCEventServer 的事件

下面有一个 OPCEventServer 对象的事件列表。其中仅介绍了特定于 SIMATIC NET 的特性。有关事件的说明，请参见下面的 OPC 规范：

报警和事件自动化接口标准

版本 1.01

1999 年 12 月 15 日

事件	含义
EventServerShutdown	OPC 服务器关闭时触发此事件。 组态程序发送关闭命令时, 或者 PC 站接收新的组态数据时, SIMATIC NET 的 OPC 服务器触发此事件。

### 集合对象 OPCEventSubscriptions

OPCEventSubscriptions 集合对象是 OPCEventSubscription 对象以及用于创建、删除和管理这些对象的方法的集合。

OPCEventSubscriptions 的属性可为所有新创建的 OPCEventSubscription 对象指定标准值。

OPCEventServer 对象的 *Connect* 调用成功执行后, 会自动创建 OPCEventSubscriptions 集合对象。OPCEventSubscriptions 始终作为 OPCEventServer 对象的属性存在并用于管理事件消息。

### OPCEventSubscriptions 的属性

下面有一个 OPCEventSubscriptions 对象属性列表。其中仅介绍了特定于 SIMATIC NET 的特性。有关各属性的说明, 请参见下面的 OPC 规范:

报警和事件自动化接口标准

版本 1.01

1999 年 12 月 15 日

属性	含义
Count	返回条目数。
DefaultIsActive	为新生成的 OPCEventSubscription 对象指定激活状态的初始值。
DefaultbufferTime	为新生成的 OPCEventSubscription 对象指定事件通知发送频率的初始值。
DefaultMaxSize	为新生成的 OPCEventSubscription 对象指定可通过单个事件通知发送的最大事件数量的初始值。

## OPCEventSubscriptions 的方法

下面有一个 **OPCEventSubscriptions** 对象的方法列表。其中仅介绍了特定于 SIMATIC NET 的特性。您将在以下 OPC 规范中找到方法的说明：

报警和事件自动化接口标准

版本 1.01

1999 年 12 月 15 日

方法	含义
Add	创建新的 OPCEventSubscription 对象，并将其添加至集合。
Item	指定对集合中索引对象的引用。
Remove	删除 OPCEventSubscription 对象。
RemoveAll	删除所有 OPCEventSubscription 对象。

## OPCEventSubscription 对象

**OPCEventSubscription** 对象代表特定的事件订阅。订阅是一组事件的季票。

客户端启动定期向事件服务器发送事件的作业。

## OPCEventSubscription 的属性

下面有一个 **OPCEventSubscription** 对象属性列表。其中仅介绍了特定于 SIMATIC NET 的特性。有关各属性的说明，请参见下面的 OPC 规范：

报警和事件自动化接口标准

版本 1.01

1999 年 12 月 15 日

属性	含义
bufferTime	确定 OPCEventSubscription 对象的事件通知的发送频率。
IsActive	指定 OPCEventSubscription 对象的激活状态。
MaxSize	指定可通过单个事件通知发送的最大事件数。
Name	指定 OPCEventSubscription 对象的名称。

**OPCEventSubscription 的方法**

下面有一个 **OPCEventSubscription** 对象的方法列表。其中仅介绍了特定于 **SIMATIC NET** 的特性。您将在以下 OPC 规范中找到方法的说明：

报警和事件自动化接口标准

版本 1.01

1999 年 12 月 15 日

方法	含义
GetFilter	<p>返回 <b>OPCEventSubscription</b> 对象的当前过滤器。 <b>SIMATIC NET</b> 的 S7 OPC 事件服务器的参数具有以下含义：</p> <p><i>EventType</i> 支持事件类型 <b>OPC_SIMPLE_EVENT</b> 和 <b>OPC_CONDITION_EVENT</b>。</p> <p><i>EventCategory</i> 事件类别将在“用于 S7 通信的事件服务器 (页 445)”部分中介绍</p> <p><i>LowSeverity</i></p> <p><i>HighSeverity</i></p> <p><i>Areas</i> OPC 事件服务器不支持区域。</p> <p><i>Sources</i> 可输入连接名称作为源。</p>
GetReturnedAttributes	返回服务器通过事件通知发送的每个事件类别的属性。
Refresh	刷新所有条件。
RefreshCancel	取消执行 <i>Refresh</i> 方法。 <b>SIMATIC NET</b> 的 S7 OPC 事件服务器不支持 <i>RefreshCancel</i> 。

方法	含义
SelectReturnedAttributes	此方法为每个事件类别指定由 <i>OnEvent</i> 方法通过事件通知返回的属性。
SetFilter	对所有过滤器进行设置以使过滤器属性与所创建事件的属性相匹配。 SIMATIC NET 的 S7 OPC 事件服务器的参数具有以下含义：  <i>EventType</i> 支持事件类型 OPC_SIMPLE_EVENT 和 OPC_CONDITION_EVENT。  <i>EventCategory</i> S7_PROCESS_ALARM  <i>LowSeverity</i>  <i>HighSeverity</i>  <i>Areas</i> S7 OPC 事件服务器不支持区域。  <i>Sources</i> 可输入连接名称作为源。

### OPCEventSubscription 的事件

下面有一个 OPCEventSubscription 对象的事件列表。其中仅介绍了特定于 SIMATIC NET 的特性。有关事件的说明，请参见下面的 OPC 规范：

报警和事件自动化接口标准

版本 1.01

1999 年 12 月 15 日

事件	含义
ConditionEvent	服务器发送条件事件时触发此事件。
RefreshCancel	中断 <i>Refresh</i> 方法时触发此事件。 SIMATIC NET 的 OPC 事件服务器不支持 <i>RefreshCancel</i> 。
RefreshComplete	完成 <i>Refresh</i> 方法时触发此事件。 SIMATIC NET 的 OPC 事件服务器不支持 <i>RefreshComplete</i> 。
RefreshConditionEvent	服务器发送与刷新条件相关的事件时触发此事件。

事件	含义
SimpleEvent	服务器发送一组简单事件时触发此事件。
TrackingEvent	服务器发送跟踪事件时触发此事件。 SIMATIC NET 的 OPC 事件服务器不支持 <i>TrackingEvent</i> 。

## 集合对象 OPCAutoEventAreaBrowsers

OPCAutoEventAreaBrowsers 集合对象是 OPCAEventAreaBrowser 对象以及用于创建、删除和管理这些对象的方法的集合。

---

### 说明

OPC 事件服务器不支持 OPCAEventAreaBrowsers。

---

## OPCAutoEventAreaBrowser 对象

利用 OPCAEventAreaBrowser 对象，客户端可以浏览服务器的整个区域和来源。区域组合在一起构成 OPCEventAreas，来源组合在一起构成 OPCEventSources。区域和来源可用于过滤事件。

## 服务器的命名空间

服务器具有平面或分层命名空间。如果命名空间为平面命名空间，则 OPCEventAreas 和 OPCEventSources 在服务器上具有相同的区域和来源。

如果命名空间是分层命名空间，则区域可视为树的分支，而来源可视为树叶。OPCAutoEventAreaBrowser 对象如指针般沿着区域移动。OPCEventAreas 和 OPCEventSources 仅包含其中存在对象的区域和来源。

---

### 说明

OPC 事件服务器不支持 OPCAEventAreaBrowser。

---

## 集合对象 OPCEvents

OPCEvents 集合对象是用于处理事件的方法的传递参数。它包含 OPCEvent 对象的集合以及用于创建、删除和管理这些对象的方法。

OPCEvents 基于服务器发送的事件通知而生成，因此可提供已触发的事件。

## OPCEvents 的属性

下面有一个 **OPCEvents** 对象属性列表。其中仅介绍了特定于 **SIMATIC NET** 的特性。  
有关各属性的说明，请参见下面的 OPC 规范：

报警和事件自动化接口标准

版本 1.01

1999 年 12 月 15 日

属性	含义
Count	返回条目数。
LastRefresh	指定 <b>OPCEvents</b> 集合对象是一系列刷新通知中的最后一个。
Refresh	指定 <b>OPCEvents</b> 集合对象属于刷新通知。

## OPCEvents 的方法

下面有一个 **OPCEvents** 对象的方法列表。其中仅介绍了特定于 **SIMATIC NET** 的特性。  
您将在以下 OPC 规范中找到方法的说明：

报警和事件自动化接口标准

版本 1.01

1999 年 12 月 15 日

方法	含义
Add	创建新的 <b>OPCEvent</b> 对象，并将其添加至集合。
Item	指定条目名称。

## OPCEvent 对象

**OPCEvent** 对象包含特定事件的通知。

## OPCEvent 的属性

下面有一个 **OPCEvent** 对象属性列表。其中仅介绍了特定于 **SIMATIC NET** 的特性。  
有关各属性的说明，请参见下面的 OPC 规范：

## 4.1 编程自动化接口

报警和事件自动化接口标准

版本 1.01

1999 年 12 月 15 日

属性	含义
AckRequired	指示条件需要确认。
ActiveTime	返回激活条件的时间。
ActorID	返回需要确认的跟踪和条件事件。
ChangeAckState	返回条件的 <i>Acknowledge</i> 属性更改的事件通知。
ChangeActiveState	返回条件的 <i>ActiveState</i> 属性更改的事件通知。
ChangeAttribute	返回条件的 <i>Attribute</i> 属性更改的事件通知。
ChangeEnableState	返回条件的 <i>Enable</i> 属性更改的事件通知。
ChangeMessage	返回条件的 <i>Message</i> 属性更改的事件通知。 SIMATIC NET 的 OPC 事件服务器不支持 <i>ChangeMessage</i> 。
ChangeQuality	返回条件的 <i>Quality</i> 属性更改的事件通知。 SIMATIC NET 的 OPC 事件服务器不支持 <i>ChangeQuality</i> 。
ChangeSeverity	返回条件的 <i>Severity</i> 属性更改的事件通知。 SIMATIC NET 的 OPC 事件服务器不支持 <i>ChangeSeverity</i> 。
ChangeSubCondition	返回条件的 <i>SubCondition</i> 属性更改的事件通知。
ConditionAcknowledged	指示条件的新状态为 <i>Acknowledged</i> 。
ConditionActive	指示条件的新状态为 <i>Active</i> 。
ConditionName	返回涉及事件通知的条件的名称。
Cookie	返回由链接到事件通知的服务器定义的 <i>cookie</i> 。 SIMATIC NET 的 OPC 事件服务器不支持 <i>Cookie</i> 。
EventCategory	返回事件的事件类别的代码。
Message	返回用于描述通知的文本。
OPCEventAttributes	返回由事件通知返回的供应商特定的事件属性。
Quality	返回与条件状态相关联的质量。 SIMATIC NET 的 OPC 事件服务器不支持 <i>Quality</i> 。
Severity	返回事件的严重程度。

属性	含义
Source	返回事件通知的来源。
SubConditionName	返回当前子条件的名称。 SIMATIC NET 的 OPC 事件服务器不支持 <i>SubConditionName</i> 。
Time	返回触发事件的时间 (UTC)。

### OPCEventCondition 对象

OPCEventCondition 对象可用于描述条件的当前状态。

### 集合对象 OPCEventSubConditions

OPCEventSubConditions 集合对象是 OPCEventSubCondition 对象的集合。该集合代表 OPC 事件服务器上条件的各种状态。

### OPCEventSubCondition 对象

OPCEventSubCondition 对象代表 OPC 事件服务器的某一特定子条件。  
OPCEventSubCondition 包含子条件的属性。

## 4.2 对自定义接口进行编程

按照满足给定要求的理想工作方式设计自定义接口。

对于使用脚本语言进行的访问，自定义接口并不适合。

对于此用途，可由自动化接口来完成。

既有用于访问过程变量的自定义接口（数据访问），也有用于处理事件和报警的自定义接口（报警和事件）。

### 应用程序

如果想要使用 C++

创建具有大量变量和高数据吞吐量的应用程序，则可对自定义接口进行编程。

## CLSID

每个 COM 类都可通过标识代码唯一标识。此代码长度为 128 位，称为 CLSID。使用 CLSID，操作系统可查找要在其中实施 COM 类的“\*.dll”或“\*.exe”文件。

如果客户端想要使用类的对象，也可以使用 CLSID 进行引用。

## ProgID

为简化 OPC 服务器的标识，可将可读 ProgID 分配给 CLSID。CLSID 和 ProgID 由 OPC 服务器的供应商指定。

对于 SIMATIC NET 的 OPC 服务器，以下几点适用于所有协议：

接口	ProgID
数据访问	<b>OPC.SimaticNET</b> (标准 OPC 服务器，适用于多协议操作) <b>OPC.SimaticNET</b> (高性能 OPC 服务器，适用于 S7、SR、DP 单协议模式； 还允许使用符号和 DX) <b>OPC.SimaticNET.DP</b> (OPC 服务器，适用于高效 DP 访问)
报警和事件，适用于 S7	<b>OPC.SimaticNetAlarms</b> (报警和事件服务器，适用于 S7) <b>OPC.SimaticNetAlarms</b> (高性能报警和事件服务器，适用于 S7 单协议模式；还允许使用符号和 DX) <b>OPC.SimaticNetAlarmsSNMP</b> (报警和事件服务器，适用于 SNMP)

## 可按以下方式操作

创建 COM 对象并调用其方法。

---

### 说明

Windows 对象是 COM 类的一个实例。COM 类与 C++ 类不同。C++ 中的类是类型定义。COM 类是对象描述，不包含类型。

### 4.2.1 创建 COM 对象并查询 OPC 服务器的状态

有关使用自定义接口进行 OPC 数据访问的详细说明，请参见示例说明。

下列描述仅以基本顺序进行了概述，未涉及错误处理。

可通过五个步骤创建 COM 对象并查询状态。

有关使用 Visual C++ 创建 COM 对象的示例，请参见“示例程序 (页 729)”部分。

### 4.2.2 用于数据访问的自定义接口的对象

本部分列出了对象的接口及其方法。其中仅介绍了特定于 SIMATIC NET 的特性。

有关接口的详细说明，请参考相关 OPC 规范。

#### 接口方法的返回值

所有方法都返回 *HResult* 类型的结果。

#### 4.2.2.1 OPCServer 对象

OPC 服务器类的对象具有各种属性，例如，其中包含有关 OPCServer 对象的状态或版本信息的属性。该类还含有客户端用于管理 OPC 组类的对象的方法。

客户端应用程序仅对直接使用 COM 机制的 OPC 服务器对象寻址。所有其它对象由相关 OPC 方法创建。

#### OPCServer 的接口

OPCServer 对象具有以下接口：

- IOPCServer
- IOPCBrowse (从 3.00 开始的新增内容)
- IOPCItemIO (从 3.00 开始的新增内容)
- IOPCBrowseServerAddressSpace (2.0)
- IOPCCCommon
- IConnectionPointContainer
- IOPCItemProperties

括号中显示了支持接口的 OPC DA 规范。

下图显示了 OPCServer 对象的接口。

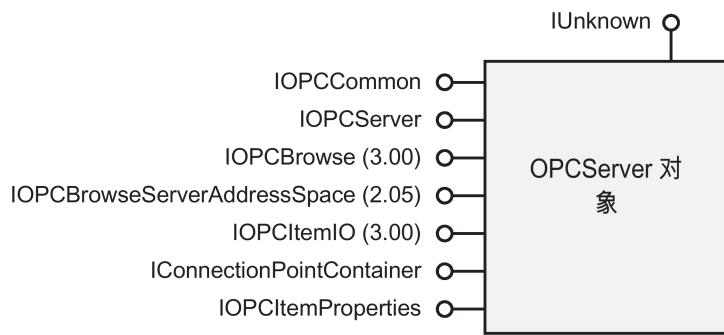


图 4-3 OPCServer 对象

### IOPCServer 接口

**IOPCServer** 接口包含用于管理 OPC 组类的对象的方法。

还可以获得有关服务器当前状态的信息。

下文提供了 **IOPCServer** 接口的方法列表。其中仅介绍了特定于 **SIMATIC NET** 的特性。

方法	含义
AddGroup	<p>在服务器对象中生成组。 不使用 <b>SIMATIC NET</b> 的 OPC 服务器评估 <i>LCID</i> 参数。 <i>pTimeBias</i> 参数可用于指定相对于本地时区的时差。 <b>SIMATIC NET</b> 的 OPC 服务器所使用的更新速率是组态期间指定的循环时间的倍数。最小更新速率与循环时间相同。 如果 <i>szName</i> 为空，则生成以 ~ 字符开头的名称（例如 ~GROUP_1）。因此，用户定义的名称不应以 ~ 开头。 如果组态包括上限 (EUHigh) 和下限 (EULow) 且数据项值更改按 (<i>pPercentDeadband</i>/100) * (EU High - EU Low) 计算，则 <i>pPercentDeadband</i> 参数会导致更改消息限制在 OPC 客户端。 此规则仅适用于模拟量数据 (EUType =1) 和标准类型 VT_I2、I4、R4、R8、BOOL、UI1。</p>
CreateGroupEnumerator	<p>为组创建各种枚举器。 <b>SIMATIC NET</b> 的 OPC 服务器不支持公共组。 因此，方法的返回值应与参数 <i>dwScope</i> 的输入值 ...PRIVATE 和 ...PUBLIC（用于公共组）相同。</p>

方法	含义
GetErrorString	获取错误代码的错误消息。 SIMATIC NET 的 OPC 服务器支持德语和英语错误文本。 <b>Windows</b> 操作系统的错误消息以操作系统安装的语言显示。
GetGroupByName	返回指向专用组名称的附加接口指针；参考计数器递增。
GetStatus	返回有关服务器的状态信息。 返回值是 OPC 服务器的名称和版本。
RemoveGroup	删除服务器中的组。 SIMATIC NET 的 OPC 服务器不支持 <i>bForce</i> 的使用。 如果对这些组的引用仍然存在，则不能删除这些组。

您将在以下 OPC 规范中找到方法的说明：

数据访问自定义接口标准

版本 3.00

2003 年 3 月 4 日

### IOPCBrowse 接口

这个新浏览接口的主要优势在于，即使在简单浏览命名空间时也可以读出 **ItemID** 和属性。这样能够显著减少 OPC 服务器的调用次数。此功能与 OPC XML-DA 的功能相同。

方法	含义
Browse	此方法可浏览名称空间中的分支，不标识或部分标识命名空间元素。命名空间应具有层级结构，无子元素的 OPC 客户端才显示平面结构。 分层命名空间具有分支子元素和叶子元素。也可以读出 <b>ItemID</b> 和属性。
GetProperties	此方法可为每个 <b>ItemID</b> 返回一个 <b>OPCItemProperties</b> 数组。

您将在以下 OPC 规范中找到方法的说明：

数据访问自定义接口标准

版本 3.00

2003 年 3 月 4 日

**IOPCItemIO 接口**

使用 IOPCItemIO 可在不使用组的情况下读取/写入数据项。

这样更为简单和高效，对于单个调用而言尤为如此（读取组态变量）。

您将在以下 OPC 规范中找到方法的说明：

数据访问自定义接口标准

版本 3.00

2003 年 3 月 4 日

方法	含义
Read	同步读取一个或多个值、质量和时间戳。可以将此函数与 IOPCSyncIO::Read 方法相比较。
WriteVQT	同步写入一个或多个值。 不支持写入质量和时间戳；拒绝时显示消息 OPC_E_NOTSUPPORTED。

**IOPCServerPublicGroups 接口**

SIMATIC NET 的 OPC 服务器不支持公共组，因此不支持可选 IOPCServerPublicGroups 接口。

**IOPCBrowseServerAddressSpace 接口**

使用 OPC DA -2.0 接口 IOPCBrowseServerAddressSpace，可浏览服务器的命名空间。

命名空间包含服务器已知的所有 OPC 数据项。

下文提供了 IOPCBrowseServerAddressSpace 接口的方法列表。其中仅介绍了特定于 SIMATIC NET 的特性。您将在以下 OPC 规范中找到方法的说明：

数据访问自定义接口标准

版本 2.05

2001 年 12 月 17 日

方法	含义
BrowseAccessPaths	获取 ItemID 的访问路径。 使用 SIMATIC NET 的 OPC 服务器时，不需要 BrowseAccessPaths。
BrowseOPCItemIDs	<p>返回 <i>IEnumString</i> 类型的字符串，其中的内容通过调用的参数指定。</p> <p>如果将输入参数 <i>dwBrowseFilterType</i> 设置为 <i>OPC_BRANCH</i>，则参数 <i>vtDataTypeFilter</i> 和 <i>dwAccessRightsFilter</i> 无效。</p> <p>创建过滤器的规则如下：</p> <ul style="list-style-type: none"> <li>*: 包括空字符串在内的任意字符串</li> <li>+: 任意字符串，但必须包含至少一个字符</li> <li>?: 恰好一个字符</li> <li>/ (方括号) : 来自指定集的一个确切字符 如果使用过滤器字符，则前面必须有反斜线字符 (\)。</li> </ul>
ChangeBrowsePosition	较高级别或命名空间中分支的更改。
GetItemID	获取分层命名空间中的完整 ItemID。 只有在单叶结构时，SIMATIC NET 的 OPC 服务器才支持 GetItemID。
QueryOrganization	返回命名空间的结构。SIMATIC NET 的 OPC 服务器的命名空间为分层结构。

## IOPCItemProperties 接口

IOPCItemProperties 接口包含用于查询有关数据项的服务器特定信息的方法。

下文提供了 IOPCItemProperties 接口的方法列表。其中仅介绍了特定于 SIMATIC NET 的特性。您将在以下 OPC 规范中找到方法的说明：

数据访问自定义接口标准

版本 2.05

2001 年 12 月 17 日

方法	含义
QueryAvailableProperties	返回数据项的可用属性列表。
GetItemProperties	返回以 PropertyID 列表形式传送的数据项的属性值。
LookupItemIDs	返回 PropertyID 列表的 ItemID 列表。

### IConnectionPointContainer 接口

IConnectionPointContainer 接口是一个用于通过连接点报告异步事件的标准 COM 接口。有关连接点的使用的更详细信息，请参考有关 COM 的文献。

### IOPCCommon 接口

IOPCCommon 接口包含将语言设置和客户端名称告知服务器的方法。

下文提供了 IOPCCommon 接口的方法列表。其中仅介绍了特定于 SIMATIC NET 的特性。您将在以下 OPC 规范中找到方法的说明：

数据访问自定义接口标准

版本 2.05

2001 年 12 月 17 日

方法	含义
SetLocaleID	设置服务器的语言代码。SIMATIC NET 的 OPC 服务器支持德语和英语。
GetLocaleID	获取服务器的语言代码。SIMATIC NET 的 OPC 服务器支持德语和英语。
QueryAvailableLocaleIDs	返回服务器所有可用语言代码。SIMATIC NET 的 OPC 服务器支持德语和英语。
GetErrorString	返回错误代码的错误文本。
SetClientName	向服务器传送描述客户端的文本。

### IPersistFile 接口

SIMATIC NET 的 OPC 服务器不支持可选 IPersistFile 接口。

### 4.2.2.2 OPCGroup 对象

OPC 组类的对象可管理各个过程变量，即 OPC 数据项。利用 OPCGroup 对象，客户端可以形成有意义的 OPC 数据项单元并使用它们执行操作。

#### OPCGroup 的接口

OPCGroup 对象具有以下接口：

- IOPCItemMgt
- IOPCGroupStateMgt / IOPCGroupStateMgt2 (从 3.00 开始的新增内容)
- IOPCSyncIO2
- IOPCAsyncIO3 (从 3.00 开始的新增内容)
- IOPCItemSamplingMgt (3.00 新增内容，可选)
- IConnectionPointContainer
- IOPCSyncIO
- IOPCSyncIO2 (从 3.00 开始的新增内容)

括号中显示了支持接口的 OPC DA 规范。

下图显示 OPCGroup 的接口。

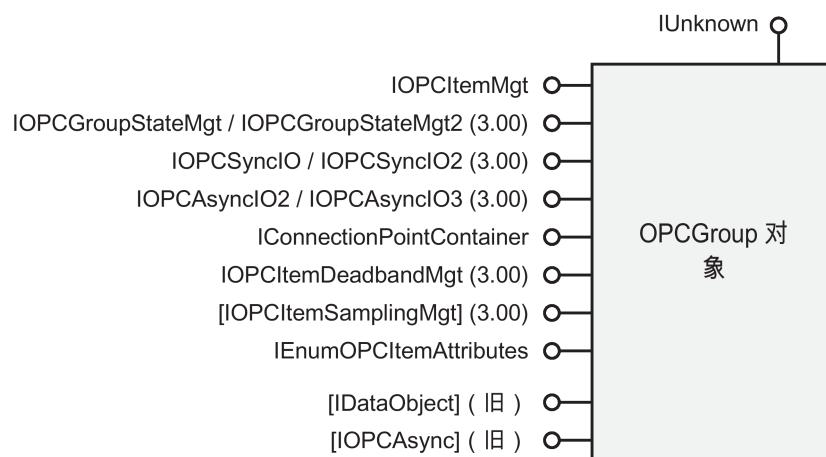


图 4-4 OPCGroup 对象 - [可选接口括在括号中]

**IOPCGroupStateMgt 接口**

**IOPCGroupStateMgt** 接口提供了用于管理组的方法。可编辑组特定参数并复制组。

下文提供了 **IOPCGroupStateMgt** 接口的方法列表。其中仅介绍了特定于 **SIMATIC NET** 的特性。您将在以下 OPC 规范中找到方法的说明：

数据访问自定义接口标准

版本 2.05

2001 年 12 月 17 日

方法	含义
CloneGroup	创建组的副本。复制组属性，以下几种情况例外： 将激活状态设置为 <b>false</b> 。 分配了新服务器句柄 <b>szName</b> 可为空。在这种情况下，生成唯一名称。
GetState	获取组的状态。 参数 <b>pTimeBias</b> 和 <b>pLCID</b> 对 <b>SIMATIC NET</b> 的 OPC 服务器没有意义。
SetName	更改组的名称
SetState	更改组的属性。 参数 <b>pTimeBias</b> 和 <b>pLCID</b> 对 <b>SIMATIC NET</b> 的 OPC 服务器没有意义。 <b>SIMATIC NET</b> 的 OPC 服务器所使用的更新速率是组态期间指定的循环时间的倍数。 最小更新速率与循环时间相同。

## IOPCGroupStateMgt2 接口

自 OPC DA 3.00 起，利用此 IOPCGroupStateMgt 扩展，可以为每个 OPC 组设置监视时间。然后 OPC 服务器周期性发送保持连接。  
如果激活数据项未发生更改，则其既可以是数据，也可以是空回调（保持连接）。

方法	含义
SetKeepAlive	使用此方法，可以通过周期性回调方式设置 OPC 服务器的生命迹象监视。OPC 客户端不需要 GetStatus() 监视。
GetKeepAlive	读出当前值

您将在以下 OPC 规范中找到方法的说明：

数据访问自定义接口标准

版本 3.00

2003 年 3 月 4 日

## IOPCPublicGroupStateMgt 接口

SIMATIC NET 的 OPC 服务器不支持公共组。因此，可选 IOPCPublicGroupStateMgt 接口没有意义。

## IOPCAsyncIO2 接口

IOPCAsyncIO2 接口包含用于异步读取和写入数据项的方法。

异步意味着客户端先启动读取或写入操作，然后再继续执行。接口使用连接点。这可简化所传送数据的处理。

OPC 为每个所读取的值都提供一个时间戳。由于 SIMATIC 系统不管理时间戳，所以将服务器接收值的时间用作时间戳。

下文提供了 IOPCAsyncIO2 接口的方法列表。其中仅介绍了特定于 SIMATIC NET 的特性。您将在以下 OPC 规范中找到方法的说明：

数据访问自定义接口标准

版本 2.05

2001 年 12 月 17 日

方法	含义
Read	发送异步读取命令。 在服务器上监视此调用是否超时。 如果在组态中设置的超时时间已过，则会发送通知，显示状态 E_ABORT。
Write	发送异步写入命令。 在服务器上监视此调用是否超时。 如果在组态中设置的超时时间已过，则会发送通知，显示状态 E_ABORT。
Cancel2	取消未决作业
Refresh2	为每个激活数据项请求缓存中的当前值
SetEnable	允许取消激活 <i>OnDataChange</i> 通知
GetEnable	返回 <i>OnDataChange</i> 通知的当前值

## IOPCAsyncIO3 接口

自 OPC DA 3.00 起，IOPCAsyncIO3 接口扩展 IOPCAsyncIO2，包括了基于 MaxAge 异步读取或写入质量和时间戳的选项。但是，OPC 服务器拒绝写入质量和时间戳，并显示 OPC\_E\_NOTSUPPORTED。

### 说明

请注意，两个接口 *IOPCAsyncIO2* 和 *IOPCAsyncIO3* 使用相同的连接点，使用相同的 cookie 和回调函数。因此，如果两个接口同时使用，应只设置一次回调。

方法	含义
ReadMaxAge	基于 MaxAge 参数异步读取一个或多个值、质量和时间戳
WriteVQT	异步写入一个或多个值。 不支持写入质量和时间戳；拒绝时显示消息 OPC_E_NOTSUPPORTED。
RefreshMaxAge	强制所有激活数据项的回调函数 IOPCDataCallback::OnDataChange 并考虑 MaxAge

您将在以下 OPC 规范中找到方法的说明：

数据访问自定义接口标准

版本 3.00

2003 年 3 月 4 日

## IOPCAsyncIO 接口

此接口包含用于异步读取和写入数据项的方法。

异步意味着客户端先启动读取或写入操作，然后再继续执行。

异步操作返回事务 ID。服务器完成读取或写入操作后，客户端会收到来自其 **IAdviseSink** 接口的通知消息（有关客户端的 **IAdviseSink** 接口的详细信息，请参见 **IDataObject** 接口的描述）。

---

### 说明

在版本 2 中，**IOPCAsyncIO** 接口由 **IOPCAsyncIO2** 替代。

**IOPCAsyncIO2** 和更新的 **IOPCAsyncIO3** 使用连接点，因此更容易处理。

在未来的项目中，使用 **IOPCAsyncIO2** 或 **IOPCAsyncIO3**。

---

OPC 为每个所读取的值都提供一个时间戳。由于 SIMATIC 系统不管理时间戳，所以将服务器接收值的时间用作时间戳。

下文提供了 **IOPCAsyncIO** 接口的方法列表。其中仅介绍了特定于 SIMATIC NET 的特性。您将在以下 OPC 规范中找到方法的说明：

数据访问自定义接口标准

版本 2.05

2001 年 12 月 17 日

方法	含义
Cancel	取消未决作业。
Read	发送异步读取命令。 在服务器上监视此调用是否超时。 相应的参数是 <i>Read/Write Timeout</i> 。 如果超时监视中止，将执行回调，并显示状态 <i>E_ABORT</i> 。
Refresh	请求每个激活 OPC 数据项的当前值。
Write	发送异步写入命令。 在服务器上监视此调用是否超时。 相应的参数是 <i>Read/Write Timeout</i> 。 如果超时监视中止，将执行回调，并显示状态 <i>hrStatus=E_ABORT</i> 。

**IOPCItemMgt 接口**

IOPCItemMgt 接口包含用于管理组中若干数据项的方法。

下文提供了 IOPCItemMgt 接口的方法列表。 其中仅介绍了特定于 SIMATIC NET 的特性。 您将在以下 OPC 规范中找到方法的说明：

数据访问自定义接口标准

版本 2.05

2001 年 12 月 17 日

方法	含义
AddItems	向组中添加一个或多个数据项。
CreateEnumerator	创建枚举器。 此为 <b>EnumOPCItemAttributes</b> 对象。
RemoveItems	从组中删除一个或多个数据项。
SetActiveState	设置组中一个或多个数据项的激活状态。
SetClientHandles	设置组中一个或多个数据项的客户端句柄。
SetDataTypes	设置组中一个或多个数据项的所需数据类型。
ValidateItems	检查 OPC 数据项的有效性。

## IOPCItemDeadbandMgt 接口

使用 OPC DA 3.00 的这个新接口，可以更改组中特定数据项的 **PercentDeadband** 参数。 **PercentDeadband**

组参数可用于所有带有组设置值的数据项，或数据项的设定值现在使用数据项 **PercentDeadband** 值进行评估，条件是该值使用此方法进行设置（默认值 = 0）。

方法	含义
SetItemDeadband	覆盖特定数据项的组值
GetItemDeadband	读取当前数据项特定值
ClearItemDeadband	将数据项特定值重置为组值

您将在以下 OPC 规范中找到方法的说明：

数据访问自定义接口标准  
版本 3.00  
2003 年 3 月 4 日

## IOPCItemSamplingMgt 接口

支持 OPC DA 3.00 的此可选接口。

使用此接口，OPC 客户端可设置采样速率 (RevisedSamplingRate)。

采样速率可高于或低于组更新速率 (RevisedUpdateRate)。

如果采样速率与组更新速率相同，则相对于上次响应没有更改。因此，OPC 组中各个数据项的采样速率能够更为精确地适应各数据项的实际更改速率。

采样速率不会更改客户端的组更新速率。

如果采样速率低于组更新速率，则 OPC 服务器可缓冲若干个值。应 OPC 客户端的请求，可在服务器上缓冲若干数据项更改。在 OPC 数据访问 3.00 中，该机制称为数据项缓冲。缓冲的默认设置为 0 = "OFF"。

方法	含义
SetItemSamplingRate	覆盖特定数据项的组更新速率
GetItemSamplingRate	读取当前数据项特定值
ClearItemSamplingRate	将数据项特定值重置为组值
SetItemBufferEnable	更改速率（采样速率）比组更新速率快时，启用或禁用数据项的数据项缓冲。
GetItemBufferEnable	读取特定数据项的状态

您将在以下 *OPC* 规范中找到方法的说明：

数据访问自定义接口标准

版本 3.00

2003 年 3 月 4 日

### IOPCSyncIO 接口

**IOPCSyncIO** 接口包含用于同步读取和写入的方法。

同步意味着客户端等待读取或写入操作完成后再继续执行。

客户端使用读取或写入操作的结果进行进一步处理。

由于 **SIMATIC NET** 的 *OPC*

服务器为每个客户端启动了单独的线程，所以客户端在等待时并不阻碍其它客户端。

下文提供了 **IOPCSyncIO** 接口的方法列表。其中仅介绍了特定于 **SIMATIC NET** 的特性。您将在以下 *OPC* 规范中找到方法的说明：

数据访问自定义接口标准

版本 2.05

2001 年 12 月 17 日

方法	含义
Read	读取某一组中的一个或多个数据项的值、状态信息和时间戳。 在服务器上监视此调用是否超时。 组态期间为每个特定协议均设置了相应的超时。
Write	写入组中一个或多个数据项的值。 在服务器上监视此调用是否超时。 组态期间为每个特定协议均设置了相应的超时。

### IOPCSyncIO2 接口

自 OPC DA 3.00 起, IOPCSyncIO2 接口扩展 IOPCSyncIO, 包括了基于 MaxAge 同步读取或写入质量和时间戳的选项。但是, OPC 服务器拒绝写入质量和时间戳, 并显示 OPC\_E\_NOTSUPPORTED。

方法	含义
ReadMaxAge	基于 MaxAge 参数同步读取一个或多个值、质量和时间戳
WriteVQT	同步写入一个或多个值。 不支持写入质量和时间戳; 拒绝时显示消息 OPC_E_NOTSUPPORTED。

您将在以下 OPC 规范中找到方法的说明:

数据访问自定义接口标准

版本 3.00

2003 年 3 月 4 日

### IDataObject 接口

IDataObject 接口是用于数据传送的标准 COM 接口。

它包含用于建立客户端与服务器组之间通知连接的方法。

当服务器向客户端发送通知时, 可通过调用 IAdviseSink 的 *OnDataChange* 方法经 IAdviseSink 客户端接口访问客户端。

下图说明了客户端上的 IAdviseSink 接口与服务器上的 IDataObject 接口之间的交互关系。

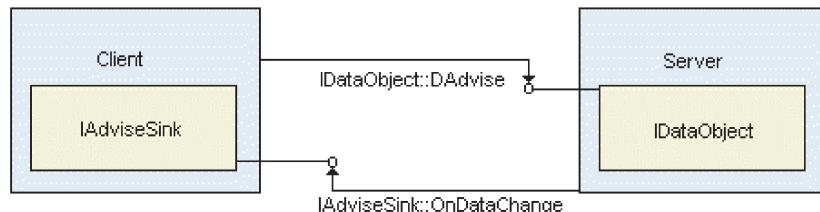


图 4-5 IAdviseSink 和 IDataObject 的交互

**说明**

*IDataObject* 在数据访问接口的版本 1 中用于异步通信。从版本 2 起，开始使用连接点，这样更为简单和灵活。  
在未来的项目中，将使用版本 2。

下文提供了 *IDataObject* 接口的方法列表。其中仅介绍了特定于 SIMATIC NET 的特性。您将在以下 OPC 规范中找到方法的说明：

数据访问自定义接口标准  
版本 2.05  
2001 年 12 月 17 日

方法	含义
DAdvise	建立服务器与客户端之间的连接
DUnadvise	终止服务器与客户端之间的连接

**IEnumOPCItemAttributes 接口**

*IEnumOPCItemAttributes* 接口基于标准 *IEnum* 接口。它返回组中的数据项。该接口由 *IOPCItemMgt* 接口的 *CreateEnumerator* 方法提供。它不能通过 *QueryInterface* 获得。

下文提供了 *IEnumOPCItemAttributes* 接口的方法列表。其中仅介绍了特定于 SIMATIC NET 的特性。您将在以下 OPC 规范中找到方法的说明：

数据访问自定义接口标准  
版本 2.05  
2001 年 12 月 17 日

方法	含义
Clone	创建与 <i>IEnumOPCItemAttributes</i> 对象完全相同的副本
Next	获取组中的下一个 OPC 数据项
Reset	将列表重置为组中的第一个数据项
Skip	跳过列表中一定数目的数据项

### 4.2.3 用于报警和事件的自定义接口的对象

下面列出了这些接口及其方法。其中仅介绍了特定于 SIMATIC NET 的特性。  
有关接口的详细说明，请参考相关 OPC 规范。

#### 接口方法的返回值

所有方法都返回 *HResult* 类型的结果。

##### 4.2.3.1 OPCEventServer 对象

OPC 事件服务器类的对象由客户端进行创建。

然后，客户端在其需要报警和事件的服务时使用此对象。

可以使用 OPCEventServer 对象执行以下任务：

- 创建消息对象
- 执行查询
- 激活事件
- 设置显示文本的语言
- 注册以接收服务器特定的事件

#### OPCEventServer 的接口

下图显示了 OPCEventServer 的接口。

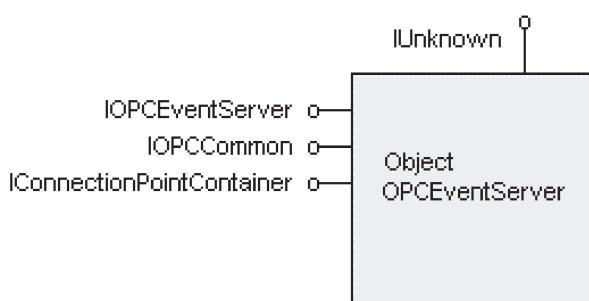


图 4-6 OPCEventServer 对象

#### IOPCCommon 接口

IOPCCommon 接口包含将语言设置和客户端名称告知服务器的方法。

下文提供了 **IOPCCCommon** 接口的方法列表。其中仅介绍了特定于 **SIMATIC NET** 的特性。您将在以下 OPC 规范中找到方法的说明：

报警和事件自定义接口标准

版本 1.10

2002 年 10 月 2 日

方法	含义
<b>SetLocaleID</b>	设置服务器的语言代码。 <b>SIMATIC NET</b> 的 OPC 服务器支持德语和英语。
<b>GetLocaleID</b>	获取服务器的语言代码。 <b>SIMATIC NET</b> 的 OPC 服务器支持德语和英语。
<b>QueryAvailableLocaleIDs</b>	返回服务器所有可用语言代码。 <b>SIMATIC NET</b> 的 OPC 服务器支持德语和英语。
<b>GetErrorString</b>	获取错误代码的错误消息
<b>SetClientName</b>	向服务器传送描述客户端的文本

## IOPCEventServer

**IOPCEventServer** 接口是报警和事件的中央接口。它的任务如下：

- 创建订阅对象
- 创建区域浏览器
- 浏览事件类别
- 管理条件

### 说明

**SIMATIC NET** 不支持可选区域，因此不会创建任何区域浏览器。

下文提供了 **IOPCEventServer** 接口的方法列表。其中仅介绍了特定于 **SIMATIC NET** 的特性。您将在以下 OPC 规范中找到方法的说明：

报警和事件自定义接口标准

版本 1.10

2002 年 10 月 2 日

方法	含义
GetStatus	获取 OPC 服务器的当前状态信息。
CreateEventSubscription	创建一个消息对象以通知客户端。 该消息对象称为订阅，如同一张通往某个事件组的票。 访问消息对象所需的接口将被返回。
QueryAvailableFilters	返回有关事件服务器所支持的过滤器选项的消息。 <b>SIMATIC NET 的 OPC</b> 事件服务器不支持以下过滤器： OPC_FILTER_BY_EVENTS: 0x01 OPC_FILTER_BY_CATEGORY: 0x02 OPC_FILTER_BY_SEVERITY: 0x04 OPC_FILTER_BY_SOURCE: 0x16
QueryEventCategories	返回由事件服务器提供的事件类别。 <b>SIMATIC NET 的 OPC</b> 事件服务器只有在参数 <i>dwEventType</i> 值为 OPC_SIMPLE_EVENT 或 OPC_CONDITION_EVENT 时，才会返回事件类别。
QueryConditionNames	返回由特定事件类别的事件服务器提供的条件。
QuerySubConditionNames	返回由特定事件类别的事件服务器提供的子条件。
QuerySourceConditions	返回由特定源的 OPC 事件服务器提供的条件。
QueryEventAttributes	返回由特定事件类别的事件服务器提供的属性。 <b>SIMATIC NET 的 OPC</b> 事件服务器提供了特殊属性。 不得通过数据访问将这些属性用作 <i>ItemID</i> 。
TranslateToItemID	获取对应于事件属性的 OPCItem，以与关联的 OPC 数据访问服务器配合使用。 <b>SIMATIC NET 的 OPC</b> 事件服务器不支持 <i>TranslateToItemID</i> 。
GetConditionState	返回有关源的条件状态的信息。

方法	含义
EnableConditionByArea	激活特定区域内的所有源的所有条件。
EnableConditionBySource	激活所有指定源的所有条件。
DisableConditionByArea	取消激活指定区域内所有源的所有条件。 SIMATIC NET 的 OPC 事件服务器不支持条件。 返回 E_NOTIMPL。
DisableConditionBySource	取消激活所有指定源的所有条件。
AckCondition	将事件确认传送到客户端。只能确认条件事件。
CreateAreaBrowser	创建一个 OPCEventAreaBrowser 对象以浏览进程空间。 SIMATIC NET 的 OPC 事件服务器不支持区域。 返回 E_NOTIMPL。

### IConnectionPointContainer

IConnectionPointContainer 接口是一个用于通过连接点报告异步事件的标准 COM 接口。  
有关连接点的使用的更详细信息，请参考有关 COM 的文献。

#### 4.2.3.2 OPCEventSubscription 对象

OPC 事件订阅类的一个对象将事件消息发送到使用该对象的 IConnectionPointContainer 接口的客户端。

一个客户端可以使用多个 OPCEventSubscription 对象。  
它可以为不同对象定义不同的过滤器。

OPCEventSubscription 相当于已定义事件的一张票。

### OPCEventSubscription 的接口

下图显示了 OPCEventSubscription 的接口。

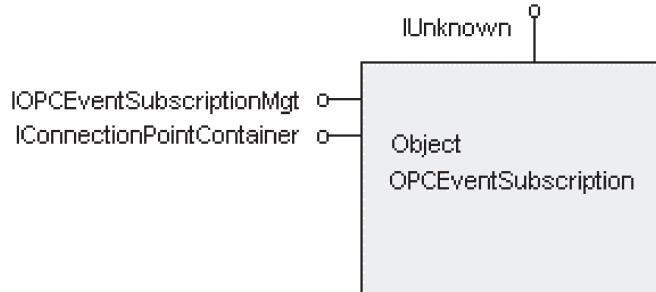


图 4-7 OPCEventSubscription 对象

### IOPCEventSubscriptionMgt 接口

IOPCEventSubscriptionMgt 接口是管理有关特定事件订阅的信息的中央接口。例如，通过该接口可以选择与客户端相关的事件。

下文提供了 IOPCEventSubscriptionMgt 接口的方法列表。其中仅介绍了特定于 SIMATIC NET 的特性。您将在以下 OPC 规范中找到方法的说明：

报警和事件自定义接口标准

版本 1.10

2002 年 10 月 2 日

方法	含义
SetFilter	<p>设置用于为该事件订阅选择特定事件的过滤器。</p> <p>过滤参数对于 SIMATIC NET 的 OPC 事件服务器具有以下含义：</p> <ul style="list-style-type: none"> <li><b>事件类型</b></li> <li>OPC 事件服务器支持 OPC_SIMPLE_EVENT 和 OPC_CONDITION_EVENT 事件类型。</li> <li><b>事件类别</b></li> <li>事件类别将在“用于 S7 通信的事件服务器 (页 445)”部分中进行介绍</li> <li><b>严重性</b></li> <li>严重性可在 STEP 7 或 SIMATIC NCM PC/S7 中进行组态。例如，S7_PROCESS_ALARM 的默认值是 500。</li> <li><b>区域</b></li> <li>SIMATIC NET 的 OPC 事件服务器不支持区域。</li> <li><b>源</b></li> <li>可以输入连接名称。</li> </ul>
GetFilter	返回事件订阅的当前使用的过滤器。 请参见 <i>SetFilter</i> 。
SelectReturnedAttributes	指定属性在返回时带有事件类别的事件消息。
GetReturnedAttributes	返回在返回时带有事件类别的事件消息的属性列表。
刷新	将与当前过滤器设置相匹配的所有激活的及未激活且未确认的条件消息发送到客户端。
CancelRefresh	取消执行当前刷新。 由于没有与 <i>Refresh</i> 一起发送的事件消息，因此 <i>CancelRefresh</i> 不起作用。
GetState	返回事件订阅的当前状态。
SetState	设置事件订阅的各种属性。

## IConnectionPointContainer 接口

**IConnectionPointContainer** 接口是一个用于通过连接点报告异步事件的标准 COM 接口。有关连接点的使用的更详细信息，请参考有关 COM 的文献。

## 参见

用于数据访问的自动化接口的对象 (页 495)

### 4.2.3.3 OPCEventAreaBrowser 对象

OPC 事件服务器不支持区域，因此不能使用 OPCEventAreaBrowser 对象。

## 4.2.4 报警和事件客户端的接口

### 事件通知

通过连接点将事件告知客户端。为此，客户端必须提供带有 **IUnknown** 接口的 COM 对象和调用特定的 **IOPCEventSink** 接口，才能接收调用。客户端在连接点注册后将指向 **IUnknown** 接口的指针传递给服务器。

### 客户端对象的接口

客户端提供的用于接收消息的对象必须具有如下所示的结构：

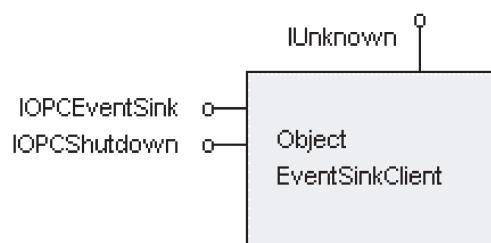


图 4-8 客户端的接口

### 4.2.4.1 IOPCEventSink 接口

**IOPCEventSink** 接口是客户端的用于接收消息的中央接口。它包含一个由服务器调用的传送事件的方法。

下文提供了 **IOPCEventSink** 接口的方法列表。其中仅介绍了特定于 **SIMATIC NET** 的特性。您将在以下 OPC 规范中找到方法的说明：

报警和事件自定义接口标准

版本 1.10

2002 年 10 月 2 日

方法	含义
<b>OnEvent</b>	<p>向客户端传送一个或多个事件消息。</p> <p><i>*pEvents</i> 结构包含一个或多个事件。 OPC 事件服务器在以下结构元素中输入特定值：</p> <p><i>SzSource</i> 连接信息。 连接信息无法使用 <code>TranslateToItemID</code> 转换为 <code>ItemID</code> 并随后由数据访问使用。</p> <p><i>ftTime</i> 伙伴设备上发生事件的时间。在 <code>EVENT_ATTR_S7_PCTIME</code> 属性中输入在 OPC 事件服务器上接收消息的时间。</p> <p><i>szMessage</i> ALARM 消息号</p> <p><i>dwEventType</i> <code>OPC_SIMPLE_EVENT</code></p> <p><i>dwEventCategory</i> <code>S7_PROCESS_ALARM</code></p> <p><i>pEventAttributes</i> 此结构包含随事件提供的属性。 属性还包括由伙伴设备提供的消息的关联值。 所有其它结构元素都是不相关的。</p>

#### 4.2.4.2 IOPCSshutdown 接口

通过这个基于连接点的调用接口，服务器可以在其被关闭或自行关闭前通知客户端。这样一来，客户端便可做出响应，并在需要时关闭。

下文提供了 **IOPCSshutdown** 接口的方法列表。其中仅介绍了特定于 **SIMATIC NET** 的特性。您将在以下 OPC 规范中找到方法的说明：

报警和事件自定义接口标准

版本 1.10

2002 年 10 月 2 日

方法	含义
ShutdownRequest	服务器将告知客户端它正在关闭。 可以在“通信设置”组态程序中设置有关关闭原因的文本 ( <i>szReason</i> )。

## 4.2.5 OPC DA 过程变量的错误消息

与协议无关的错误代码

错误代码	含义
OPC_E_INVALIDHANDLE (0xC0040001L)	句柄的值无效。
IDS_DUPLICATE (0xC0040002L)	多次传送相同的值。
IDS_UNKNOWNLCID (0xC0040003L)	服务器不支持指定的本地 <i>ID</i> 。
OPC_E_BADTYPE (0xC0040004L)	服务器不支持 <i>anonicalDatatype</i> 与 <i>requestedDatatype</i> 之间的转换。
OPC_E_PUBLIC (0xC0040005L)	不能为公共组执行所需的操作。
OPC_E_BADRIGHTS (0xC0040006L)	所需的操作（读取或写入）受数据项访问权限的保护。
OPC_E_UNKNOWNITEMID (0xC0040007L)	服务器的命名空间中不存在名称（数据项定义）。
OPC_E_INVALIDITEMID (0xC0040008L)	名称（数据项定义）具有非法语法。
OPC_E_INVALIDFILTER (0xC0040009L)	不允许使用过滤器的字符序列。

错误代码	含义
OPC_E_UNKNOWNPATH (0xC004000AL)	不允许将连接名称指定为 AccessPath。
OPC_E_RANGE (0xC004000BL)	值超出了允许的取值范围。
OPC_S_UNSUPPORTEDRATE (0x0004000DL)	服务器不支持所需的更新率。 使用最近的允许值。
OPC_S_CLAMP (0x0004000EL)	在写入中传送的值已被接受，但却在输出时被截断。
OPC_S_INUSE (0x0004000FL)	由于仍然存在对象引用，所以不能完全执行此操作。
OPC_E_NOTFOUND (0xC0040011L)	GetErrorString 方法以 <i>dwError=%lx</i> 形式返回字符串，其中 %lx 是 OPC 服务器未知的十六进制错误。
0xC0048003L 0x00048003L	例如，由于连接中断，发生超时。
0xC0048004L 0x00048004L	在内部使用的服务已关闭。
0xC0048005L 0x00048005L	所需的操作（读取或写入）受数据项访问权限的保护。
0xC0048006L 0x00048006L	已发生通信错误。 检查通信伙伴和电缆连接。
0xC0048007L 0x00048007L	值高于允许的取值范围。
0xC0048008L 0x00048008L	值低于允许的取值范围。
0xC0048009L 0x00048009L	转换过程中出错。
0x85270181L 0x05270181L	回调缓冲区已溢出。

错误代码	含义
CONNECT_E_NOCONNECTION (0x80040200)	无连接。该消息与 OPC 服务器的内部 COM 连接相关，但却与 S7 或传输连接无关。
CONNECT_E_CANNOTCONNECT (0x80040202)	无法建立连接。 检查异步连接点的数目（只有 1 个异步连接点可用）。

### DP 协议的错误代码

错误代码	含义
0x85270101L 0x05270101L	DP 主站无法运行。
0x85270102L 0x05270102L	DP 从站无法运行。
0x05270165L	DP 主站处于 CLEAR 或 AUTOCLEAR 模式。

### S7 协议的错误代码

错误代码	含义
0x85270201L 0x05270201L	缓冲区发送/接收服务： r_id 无效。
0x85270202L 0x05270202L	域服务： 访问权限无效。
0x85270203L 0x05270203L	域服务： 发生块错误。
0x85270204L 0x05270204L	域服务： 发生文件错误。
0x85270205L 0x05270205L	针对 SIMOTION 项目的一致性检查失败。
0x85270206L 0x05270206L	未与 S7 通信伙伴建立任何连接，已组态为立即返回错误。

## PROFINET 协议的错误代码

错误代码	含义
0x85270601L 0x05270601L	请求的值不是永久性的。
0x85270602L 0x05270602L	未建立连接。
0x85270603L 0x05270603L	无连接
0x85270605L 0x05270605L	值不确定。
0x85270606L 0x05270606L	无效描述（过长、非法字符、无分隔符、语法错误）。
0x8527060aL 0x0527060aL	无效枚举。
0x8527060bL 0x0527060bL	无效 ID。
0x8527060cL 0x0527060cL	无效 epsilon 类型或值。
0x8527060dL 0x0527060dL	替换值无效。
0x8527060eL 0x0527060eL	与其本身的非法连接。
0x8527060fL 0x0527060fL	无效 cookie 值。
0x85270610L 0x05270610L	不受支持的时间。
0x85270611L 0x05270611L	不受支持的 QoS 类型。
0x85270614L 0x05270614L	已连接目标。
0x85270604L 0x05270604L	只缓冲该值。

错误代码	含义
0x85270607L 0x05270607L	未知对象。
0x85270608L 0x05270608L	未知属性。
0x85270609L 0x05270609L	返回的类型与预期的类型不匹配。
0x85270612L 0x05270612L	不支持 QoS 值（服务质量）。
0x85270613L 0x05270613L	系统当前正在保存，组态当前无法进行更改。
0x85270615L 0x05270615L	该操作当前无法应用。
0x85270616L 0x05270616L	访问遭到拒绝。
0x85270617L 0x05270617L	检测到硬件故障。
0x85270A01L 0x05270A01L	无法解析地址信息。
0x85270A02L 0x05270A02L	数据长度无效。
0x05270A64L	已缓冲提供者或使用者的状态，但尚未传送到设备。
0x05270A65L	已缓冲控制器的初始值，但尚未传送到设备（例如，因设备故障所致）。

### SEND/RECEIVE 协议的错误代码

错误代码	含义
0x85270301L 0x05270301L	未与 SR 通信伙伴建立任何连接，已组态为立即返回错误。

## Windows 错误代码

错误代码	含义
0x80070005	访问遭到拒绝。
0x80070057	参数无效。

---

### 说明

首字节为零 (0x0....) 的应答消息是有条件的成功消息。

然后，用户便可假定：即使质量“不良”或“不确定”，也会定义变量值、质量和时间戳。

---

## 4.3 对 XML 接口进行编程

---

### 说明

从版本 6.1 开始，SIMATIC NET 软件提供 OPC XML 接口。

---

## 简介

本部分介绍 XML DA 接口的方法。

以图形方式介绍方法的变量结构并描述各个元素和属性。

首先会显示方法描述的语法。接着显示对基本模式的描述。它们是 OPC XMLDA 模式定义中多次出现的元素。随后，我们会查看各个方法。

## 数据模型的 XML 模式定义

与 XMLSchema2001 的定义一致，OPC XML-DA 的 WSDL

模式段描述了用途各个方法的参数的复杂数据类型。可在 OPC XMLDA 规范中找到精确描述。

在 Web 服务的代码中，从 XML 模式为每个复杂的数据类型创建了一个 C# 类。Web 服务的方法将在接口中进行描述。

## XML DA 接口使用的相关注意事项

---

### 说明

#### 对于与 OPC XML DA

相关的产品特定错误消息，不会返回错误文本，而仅返回错误编号。可在“OPC DA 过程变量的错误消息 (页 543)”部分查阅相应的错误文本。

---

---

### 说明

若要操作 OPC XML DA，需在“通信设置”组态程序的“安全性”(Security) 中启用远程 OPC 通信。

---

---

### 说明

为利用相应的符号，OPC XML DA Web 服务会使用其它服务。

这些服务需具备足够的权限才能访问存有 STI 或 ATI 符号文件的文件夹以及自身文件。

因此，请确保以下用户具有针对这些文件和文件夹的“完全访问”权限：

**操作系统： Windows 7 或 Windows 8**

**德语操作系统的用户：“NETZWERKDienst”用户**

**英语操作系统的用户：“NETWORK SERVICE”用户**

---

---

### 说明

OPC XML DA 服务分配了以下用于创建跟踪文件的附加服务：

- Windows 7 和 Windows 8 中的网络服务

请为这些服务分配针对跟踪文件存储文件夹的读写权限。

---

### 说明

由于可对 Internet 信息服务器进行组态，因此当发生某些特定事件时它可以自动复位。在 Windows 7 和 Windows 8 中，自动复位默认设定为 29 小时的操作时间。请注意：

- 这些设定可能导致回复延迟。
  - 每次复位后会丢失活动的订阅，必须重新发送。
- 

## 4.3.1 元素的描述

### 元素描述中使用的约定

OPCXML 元素的描述前面有变量结构的描述。

- 变量结构中的元素以**粗体文本**指示。
- 变量结构中的属性以**斜体**指示。

标签的嵌套以其缩进深度表示。标签的属性也会缩进并超出相应的标签。

### 示例

变量结构可能如下所示：

```
<tag1>
    <tag2 att1="yes" att2="no">content of tag2
    </tag2>
    <tag3>content of tag3
    </tag3>
</tag1>
```

在 `<tag1>` 的描述中，可能会发现如下所示的结构：

```
tag1
  tag2
    att1
    att2
  tag3
```

因为元素 `tag2` 和 `tag3` 是 `tag1` 的子元素，所以它们比 `tag1` 缩进得更远。在 `tag2` 后，您还可立即看到该元素的属性 `att1` 和 `att2`。

### 4.3.2 基本模式

#### OPC XMLDA 的模式定义

OPCXMLDA 的所有元素都有模式定义。

在这些描述中，我们通常指的是基本的常用元素的定义。

以下部分描述了这些元素中的某些元素：

##### **ItemProperty**

用于定义数据项属性的元素。

##### **ItemValue**

用于指定数据项的值及任何相关附加信息（如时间戳）的元素。

##### **OPCError**

包含 OPC 错误的错误代码和描述。

##### **ReplyBase**

带服务器响应基本信息（如响应发送时间）的元素。

##### **RequestOptions**

包含详细指定客户端请求的信息。

### 4.3.2.1 ItemProperty

#### 变量结构

##### ItemProperty

*Name*

*Description*

*ItemPath*

*ItemName*

*ResultID*

**Value**

#### 元素和属性

名称	说明
名称	包含属性名称。
说明	属性的描述。
ItemPath	元素的路径信息。 SIMATIC NET 不支持该属性。
ItemName	此元素可以通过 <b>ItemName</b> 属性在服务器的命名空间中进行唯一标识。
ResultID	如果发生错误或非关键例外，该属性将包含 OPC 错误的名称。
值	属性的当前值。

### 4.3.2.2 ItemValue

#### 变量结构

**ItemValue**

- ValueTypeQualifier*
- ItemPath*
- ItemName*
- ClientItemHandle*
- Timestamp*
- ResultID*
- DiagnosticInfo**
- Value**
- Quality**

  - QualityField*
  - LimitField*
  - VendorField*

#### 元素和属性

名称	说明
<b>ValueTypeQualifier</b>	该属性仅与类型时间、日期和持续时间一起使用来指定数据类型。 对于所有其它数据类型，该属性不存在或者被忽略（如果存在）。
<b>ItemPath</b>	元素的路径信息。 SIMATIC NET 不支持该属性。
<b>ItemName</b>	此元素可以通过 <b>ItemName</b> 属性在服务器的命名空间中进行唯一标识。
<b>ClientItemHandle</b>	服务器在其回复内返回的请求中的、由客户端指定的字符串。 客户端将其用于更复杂的系统中，以标识回复并将其分配给正确的请求。
<b>Timestamp</b>	上一次从服务器获取值的时间。
<b>ResultID</b>	如果发生错误或非关键例外，该属性将包含 OPC 错误的名称。

名称	说明
DiagnosticInfo	详细的服务器特定的诊断信息。
值	数据项的值。 由于该属性涉及多形态信息，因此需要附加属性 <code>xsi:type</code> (例如, <code>:xsi:type=xsd:float</code> )。
Quality	有关数据质量的信息。 如果服务器没有返回该属性，则质量良好。 如果质量不良或不确定，则返回该属性。
QualityField	如果质量的值为良好，则不返回该属性。 如果质量不良，并且项的先前值已知，则该属性的值为 <code>badLastKnownValue</code> 。
LimitField	包含对 OPC 限制位字段进行寻址时所使用的名称。 除非限制状态为无，否则总是传送该属性。
VendorField	与 OPC 供应商位字段相对应的数值。 是否传送该属性取决于供应商。

#### 4.3.2.3 OPCError

##### 变量结构

###### OPCError

*ID*

**Text**

##### 元素和属性

名称	说明
ID	包含 OPC 错误的名称。
文本	文本形式的错误的描述。字符串的内容取决于 <code>LocaleID</code> 属性。

#### 4.3.2.4 ReplyBase

##### 变量结构

###### ReplyBase

*RcvTime*  
*ReplyTime*  
*ClientRequestHandle*  
*RevisedLocaleID*  
*ServerState*

##### 元素和属性

名称	说明
RcvTime	指示服务器接收请求的时间。 RcvTime 是一个强制属性。
ReplyTime	指示服务器发送回复的时间； 强制属性。
ClientRequestHandle	如果该属性包含在客户端的请求中，则将由服务器返回并带有响应。
RevisedLocaleID	如果客户端为服务器所支持的 <i>LocaleID</i> 属性指定一个值，服务器将返回 <i>LocaleID</i> 属性的默认值并具有 <i>RevisedLocaleID</i> 属性。
ServerState	指定服务器的状态，并总是被返回； 强制属性。

## 4.3.2.5 RequestOptions

## 变量结构

**RequestOptions**

*ReturnErrorText*  
*ReturnDiagnosticInfo*  
*ReturnItemTime*  
*ReturnItemName*  
*ReturnItemPath*  
*RequestDeadline*  
*ClientRequestHandle*  
*LocaleID*

## 元素和属性

名称	说明
ReturnErrorText	此属性的默认值为 TRUE。 这种情况下，服务器返回详细的错误描述。
ReturnDiagnosticInfo	如果该属性的值为 TRUE，则服务器将返回详细的诊断信息。
ReturnItemTime	指定是否为每个数据项返回时间戳。 默认值为 FALSE，不返回任何值。
ReturnItemName	指定是否为每个数据项返回 <b>ItemName</b> 属性。 默认值为 FALSE，不返回任何值。
ReturnItemPath	指定是否为每个数据项返回 <b>ItemPath</b> 属性。 如果 <b>ReturnItemPath=true</b> 或 <b>false</b> ，则总是返回路径（非零）。 默认值为 FALSE。

名称	说明
RequestDeadline	<p>指定客户端等待来自服务器的响应的最迟时间。</p> <p>此时，不可用的项的数据将作为错误返回。</p> <p>如果指定的时间比服务器上的当前时间早，整个请求将失败。客户端必须提供 UTC 时间格式的时间信息。</p> <p>要使用该值，必须在传送参数时设置 <code>RequestDeadlineSpecified=true</code>。</p> <p>例外</p> <p>使用 Microsoft Development Environment 2003 开发的 OPC XML DA 客户端必须以 <code>RequestOptions.RequestDeadline</code> 形式传送本地时间，而不是 UTC 时间。Visual Studio 2003 的标准 SOAP 编写器总是将传送时间转换为本地时间 + 时间偏移量格式的 UTC 时间。</p>
ClientRequestHandle	<p>如果该属性包含在客户端的请求中，则将由服务器返回并带有响应。</p> <p>在庞大而复杂的系统中，该信息有助于客户端为请求分配响应。指定该属性为可选操作。</p>
LocaleID	一个可选属性，客户端通过该属性为某些返回的数据指定语言。

### 4.3.3 读取和写入作业

#### 4.3.3.1 读取

##### 变量结构

###### Read

###### Options

*ReturnErrorText*  
*ReturnDiagnosticInfo*  
*ReturnItemTime*  
*ReturnItemName*  
*ReturnItemPath*  
*RequestDeadline*  
*ClientRequestHandle*  
*LocaleID*

###### ItemList

*ItemPath*  
*ReqType*  
*MaxAge*

###### Items

*ItemPath*  
*ReqType*  
*ItemName*  
*ClientItemHandle*  
*MaxAge*

##### 元素和属性

###### 读取

<Read> 元素包含读取作业的所有信息。

###### 选项

<Options> 元素包含可用于 XMLDA 请求的选项。该元素类型为 RequestOptions。有关该元素属性的信息，请参见本文档中的相关部分。

###### ItemList

各数据项的容器元素。一个请求可包含多个数据项。

*ItemPath*、*ReqType* 和 *MaxAge* 属性可用于 *ItemList* 和 *Items* 元素。项的属性覆盖 *ItemList* 的同名属性。

#### ItemPath

元素的路径。SIMATIC NET 不支持该属性。

#### MaxAge

类型为 *xsd:int* 的一段时间，其指定数据的最大老化时间。

数据不得超过此处指定的持续时间。

#### ReqType

为客户端请求的数据项的值指定数据类型。支持规范中所列的所有数据类型。

#### Items

指定单个数据项所使用的元素。

#### ItemName

数据项的名称。

#### ClientItemHandle

由客户端分配的字符串。如果该属性由客户端指定，则服务器必须将其返回。

### 示例

```
<soap:Body>
    <Read
        xmlns="http://opcfoundation.org/webservices/XMLDA/1.0/">
        <Options
            ReturnErrorText="false"
            ReturnItemTime="true"
            ReturnItemName="true"
            LocaleID="en" />
        <ItemList>
            <Items ItemName="Simple Types/UInt" />
            <Items ItemName="Simple Types/Int" />
            <Items ItemName="Simple Types/Float" />
        </ItemList>
    </Read>
</soap:Body>
```

## 同步读取 web 方法的 C# 接口

使用 C# 用户程序时，可以使用以下方法读取值。仅需指定上面所列出的部分参数：

```
ReplyBase Read(RequestOptions Options,  
                ReadRequestItemList ItemList,  
                out ReplyItemList ItemValues,  
                out OPCError[] Error);
```

### 4.3.3.2 ReadResponse

#### 变量结构

```
ReadResponse  
  ReadResult  
    RcvTime  
    ReplyTime  
    ClientRequestHandle  
    RevisedLocaleID  
    ServerState  
  RItemList  
    Reserved  
    Items  
      ValueTypeQualifier  
      ItemPath  
      ItemName  
      ClientItemHandle  
      Timestamp  
      ResultID  
      DiagnosticInfo  
    Value  
    Quality  
      QualityField  
      LimitField  
      VendorField  
  Errors  
    ID  
    Text
```

## 元素和属性

### **ReadResponse**

使用 `<ReadResponse>` 元素，服务器可返回读取作业的结果。

### **ReadResults**

`<ReadResults>` 元素包含有关服务器响应的基本信息。

该元素属于 `ReplyBase` 类型。有关该元素属性的信息，请参见本文档中的相关部分。

### **RItemList**

`RItemList` 是响应的各个数据项的容器元素。一个响应可包含多个数据项。

### **Reserved**

该属性阻止基于 WSDL 的程序以代码生成数据项的数组形式表示返回列表。

### **Items**

指定单个数据项所使用的元素。

该元素属于 `ItemValue` 类型。有关该元素属性的信息，请参见本文档中的相关部分。

### **Errors**

与此响应一起出现的错误的数组。

该元素属于 `OPCError` 类型。有关该元素属性的信息，请参见本文档中的相关部分。

## 示例

```
<soap:Body>
    <ReadResponse
        xmlns="http://opcfoundation.org/webservices/XMLDA/1.0/">
        <ReadResult
            RcvTime="2003-05-27T00:15:36.6400000-07:00"
            ReplyTime="2003-05-27T00:15:36.7500000-07:00"
            ServerState="running"
        />
        <RItemList>
            <Items
                ItemName="Simple Types/UInt"
                Timestamp="2003-05-27T00:15:36.7343750-07:00">
                    <Value xsi:type="xsd:unsignedInt">4294967295</Value>
                </Items>
                <Items
                    ItemName="Simple Types/Int"
                    Timestamp="2003-05-27T00:15:36.7343750-07:00">
                    <Value xsi:type="xsd:int">2147483647</Value>
                </Items>
                <Items
                    ItemName="Simple Types/Float"
                    Timestamp="2003-05-27T00:15:36.7343750-07:00">
                    <Value xsi:type="xsd:float">3.402823E+38</Value>
                </Items>
            </RItemList>
        </ReadResponse>
    </soap:Body>
```

### 4.3.3.3 写入

#### 变量结构

##### Write

*ReturnValuesOnReply*  
**Options**  
    *ReturnErrorText*  
    *ReturnDiagnosticInfo*  
    *ReturnItemTime*  
    *ReturnItemName*  
    *ReturnItemPath*  
    *RequestDeadline*  
    *ClientRequestHandle*  
    *LocaleID*  
**ItemList**  
    *ItemPath*  
**Items**  
        *ValueTypeQualifier*  
        *ItemPath*  
        *ItemName*  
        *ClientItemHandle*  
        *Timestamp*  
        *ResultID*  
        **DiagnosticInfo**  
        **Value**  
        **Quality**  
            *QualityField*  
            *LimitField*  
            *VendorField*

#### 元素和属性

##### Write

<Write> 元素包含写入作业的所有信息。可为一个或多个数据项执行写入作业。

##### ReturnValuesOnReply

通过该属性，客户端可以指定是否应为每个数据项返回一个值。

返回服务器为该写入作业处理的值。

如果在写入后立即进行读取，服务器将返回相同值。

如果写入作业失败，则不会返回任何值。通过只写值，服务器返回 E\_WRITEONLY。

### Options

<Options> 元素包含可用于 XML DA 请求的选项。该元素属于 RequestOptions 类型。有关该元素属性的信息，请参见本文档中的相关部分。

### ItemList

各数据项的容器元素。一个请求可包含多个数据项。

### ItemName

此元素可以通过 ItemName 属性在服务器的命名空间中进行唯一标识。

### Items

指定单个数据项所使用的元素。该元素类型为 ItemValue。

有关该元素属性的信息，请参见本文档中的相关部分。

---

### 说明

当写入并读回 *DateTime* 数据项时，会发生由 .NET 互操作层引起的长达 2 毫秒的错误。

## 示例

```
<soap:Body>
    <Write
        xmlns="http://opcfoundation.org/webservices/XMLDA/1.0/">
        <Options
            ReturnErrorText="false"
            ReturnItemName="true"
            LocaleID="en"
        />
        <ItemList>
            <Items ItemName="Simple Types/UInt">
                <Value xsi:type="xsd:unsignedInt">4294967295</Value>
            </Items>
            <Items ItemName="Simple Types/Int">
                <Value xsi:type="xsd:int">2147483647</Value>
            </Items>
            <Items ItemName="Simple Types/Float">
                <Value xsi:type="xsd:float">3.402823E+38</Value>
            </Items>
        </ItemList>
    </Write>
</soap:Body>
```

## 同步写入 web 方法的 C# 接口

使用 C# 用户程序时，可以使用以下方法写入值。仅需指定上面所列出的部分参数：

```
ReplyBase Write(RequestOptions Options,
                 WriteRequestItemList ItemList,
                 bool ReturnItemVal,
                 out ReplyItemList ItemValues,
                 out OPCError[] Error);
```

#### 4.3.3.4 WriteResponse

##### 变量结构

```
WriteResponse
  WriteResult
    RcvTime
    ReplyTime
    ClientRequestHandle
    RevisedLocaleID
    ServerState
  RItemList
    Reserved
    Items
      ValueTypeQualifier
      ItemPath
      ItemName
      ClientItemHandle
      Timestamp
      ResultID
      DiagnosticInfo
      Value
      Quality
        QualityField
        LimitField
        VendorField
  Errors
    ID
    Text
```

##### 元素和属性

###### WriteResponse

使用 <WriteResponse> 元素，服务器可返回写入作业的结果。

###### WriteResults

<WriteResults> 包含有关服务器响应的基本信息。

该元素属于 ReplyBase 类型。有关该元素属性的信息，请参见本文档中的相关部分。

**RItemList**

各数据项的容器元素。一个响应可包含多个数据项。

**Reserved**

该属性阻止基于 WSDL 的程序以代码生成数据项的数组形式表示返回列表。

**Items**

指定单个数据项所使用的元素。

该元素属于 **ItemValue** 类型。有关该元素属性的信息，请参见本文档中的相关部分。

**Errors**

与此响应一起出现的错误的数组。

该元素属于 **OPCError** 类型。有关该元素属性的信息，请参见本文档中的相关部分。

**示例**

```
<soap:Body>
    <WriteResponse
        xmlns="http://opcfoundation.org/webservices/XMLDA/1.0/">
        <WriteResult
            RcvTime="2003-05-27T05:19:26.3687500-07:00"
            ReplyTime="2003-05-27T05:19:26.4687500-07:00"
            ServerState="running"
        />
        <RItemList>
            <Items ItemName="Simple Types/UInt" />
            <Items ItemName="Simple Types/Int" />
            <Items ItemName="Simple Types/Float" />
        </RItemList>
    </WriteResponse>
</soap:Body>
```

#### 4.3.4 使用订阅

##### 通过多个方法调用支持的关联

与其它大多数 Web 服务相比，OPC XMLDA Web 服务支持通过几个方法调用与已订阅的 Web 客户端之间的宽松关联。该宽松的关联将在下文中称为轮询订阅。

数据项的值可以使用轮询订阅循环读取。

## 客户端登录和退出

客户端使用 OPC XML-DA Web 服务登录订阅。

然后，这将指出客户端感兴趣的数据包项，并查询它们的值。

当来自客户端的轮询请求到达时，返回当前的值。客户端终止该订阅。

## 使用句柄进行标识

为标识订阅，可通过以后的每次调用将由服务器分配的句柄作为参数进行传送。

此外，还可以指定一个句柄列表，以便每次进行 `SubscriptionPolledRefresh` 时多个订阅是相关的。这将提高轮询的效率。

订阅对象的地址将以订阅句柄的形式返回给客户端，以便下次进行

`SubscriptionPolledRefresh` 和 `SubscriptionCancel` 时能够唯一标识该订阅。

下图说明了订阅的使用：

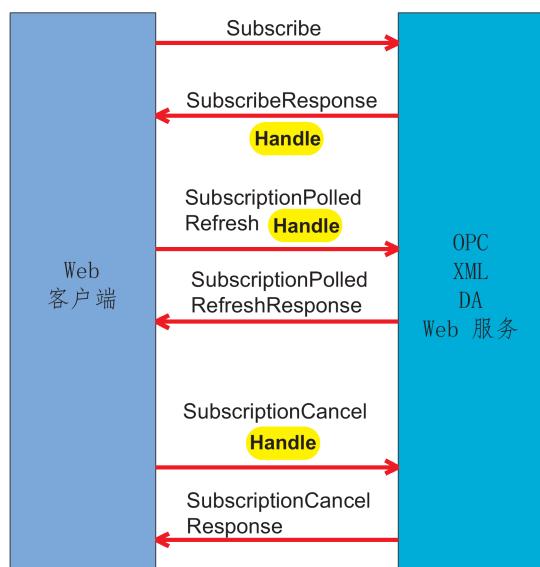


图 4-9 订阅的使用

## 监控数据项的有效期

`SubscriptionPolledResponse` 方法将通过 `SubscriptionPolledRefresh` 方法请求的值返回到客户端。`SubscriptionCancel` 放弃订阅及其数据项。

如果指定时间内没有来自客户端的进一步的 `SubscriptionPolledRefresh` 请求到达，OPC DA Web 服务将以与 `SubscriptionCancel` 方法相同的方式自动终止订阅。

#### 4.3.4.1 订阅

##### 变量结构

```
Subscribe
  ReturnValuesOnReply
  SubscriptionPingRate
  Options
    ReturnErrorText
    ReturnDiagnosticInfo
    ReturnItemTime
    ReturnItemName
    ReturnItemPath
    RequestDeadline
    ClientRequestHandle
    LocaleID
  ItemList
    ItemPath
    ReqType
    Deadband
    RequestedSamplingRate
    EnableBuffering
  Items
    ItemPath
    ReqType
    ItemName
    ClientItemHandle
    Deadband
    RequestedSamplingRate
    EnableBuffering
```

##### 元素和属性

###### 订阅

<Subscribe> 元素包含订阅作业的所有信息。

###### ReturnValuesOnReply

如果该属性的值设为 TRUE，服务器将返回可用于 SubscribeResponse 的值。如果值为 FALSE，服务器将不返回任何值以及 SubscribeResponse。

**SubscriptionPingRate**

该属性的值指定服务器检查客户端是否存在的时间间隔。

如果在指定时间内客户端尚未与服务器通信，服务器会释放客户端的订阅所需的所有资源。

**Options**

**<Options>** 元素包含可用于 XML DA 请求的选项。该元素属于 **RequestOptions** 类型。有关该元素属性的信息，请参见本文档中的相关部分。

**ItemList**

各个数据项的容器元素。一个请求可包含多个数据项。

*ItemPath*、*ReqTyp*、*Deadband*、*RequestedSamplingRate* 和 *EnableBuffering* 属性可用于 *ItemList* 和 *Items* 元素。*item* 的属性将覆盖 *ItemList* 的同名属性。

**ItemPath**

元素的路径。 SIMATIC NET 不支持该属性。

**ReqType**

为客户端请求的数据项的值指定数据类型。

**Deadband**

指定不触发 **SubscriptionPolledRefresh** 的数据项值更改的阀值上限。

该值将以数据项的最大值范围的百分比形式进行指定。因此，该参数值应在 0% 到 100% 之间，并且只能与整型和浮点型数据项一起使用。

**RequestedSamplingRate**

由客户端指定的时间（以毫秒为单位），在此时间之后服务器将检查值是否发生更改。

**EnableBuffering**

如果该属性的值设为 TRUE，服务器将根据 *RequestedSamplingRate* 参数将所有的值更改保存在缓冲区中。服务器将在下一次 *PolledRequest* 时将保存的数据返回到客户端。

**ItemName**

数据项的名称。

**ClientItemHandle**

由客户端分配的字符串。如果该属性由客户端指定，则服务器必须将其返回。

#### 4.3.4.2 SubscribeResponse

##### 变量结构

```
SubscribeResponse
  ServerSubHandle
  SubscribeResult
    RcvTime
    ReplyTime
    ClientRequestHandle
    RevisedLocaleID
    ServerState
  RItemList
    RevisedSamplingRate
  Items
    RevisedSamplingRate
    ItemValue
      ValueTypeQualifier
      ItemPath
      ItemName
      ClientItemHandle
      Timestamp
      ResultID
      DiagnosticInfo
      Value
      Quality
        QualityField
        LimitField
        VendorField
  Errors
    ID
    Text
```

##### 元素和属性

###### SubscribeResponse

<SubscribeResponse> 元素包含 SubscribeResponse 的所有信息。

### **ServerSubHandle**

服务器为该值指定的值必须与 `SubscriptionPolledRefresh` 请求和 `SubscriptionCancel` 请求配合使用。该属性标识发送请求的客户端。

### **SubscribeResult**

`<SubscribeResult>` 元素包含有关服务器响应的基本信息。

该元素属于 `SubscribeResult`

类型。有关该元素属性的信息，请参见本文档中的相关部分。

### **RItemList**

包含数据项元素。只有在客户端通过 `ReturnValuesOnReply`

属性的对应值请求数据项的可用值时，才会将这些值发送给客户端（请参见订阅）。

### **RevisedSamplingRate**

服务器支持的最快更新周期。该值由服务器返回到客户端。

### **Items**

指定单个数据项所使用的元素。

该元素属于 `ItemValue` 类型。有关该元素属性的信息，请参见本文档中的相关部分。

### **Errors**

与此响应一起出现的错误的数组。

该元素属于 `OPCError` 类型。有关该元素属性的信息，请参见本文档中的相关部分。

#### 4.3.4.3 SubscriptionPolledRefresh

##### 变量结构

###### **SubscriptionPolledRefresh**

*HoldTime*  
*WaitTime*  
*ReturnAllItems*  
**Options**  
*ReturnErrorText*  
*ReturnDiagnosticInfo*  
*ReturnItemTime*  
*ReturnItemName*  
*ReturnItemPath*  
*RequestDeadline*  
*ClientItemHandle*  
*LocaleID*  
**ServerSubHandles**

##### 元素和属性

###### **SubscriptionPolledRefresh**

客户端通过该元素请求服务器刷新先前订阅请求中已定义的项。

###### **HoldTime**

该属性指示服务器在服务器内部时间等于或大于该属性的值之前，不要向客户端发送任何刷新的值。

###### **WaitTime**

达到 HoldTime 属性中指定的时间后，服务器会在返回响应前等待 WaitTime 中所指定的时间。如果数据项值在 WaitTime 期间发生更改，则立即发送响应。

###### **ReturnAllItems**

如果该属性的值设为 FALSE，服务器只返回自上次 SubscriptionPolledRefresh 请求和当前 SubscriptionPolledRefresh 请求之间发生更改的数据项。

如果该属性的值设为 TRUE，服务器将返回在相应订阅请求中定义的所有数据项。

###### **Options**

<Options> 元素包含可用于 XML DA 请求的选项。该元素属于 RequestOptions 类型。有关该元素属性的信息，请参见本文档中的相关部分。

**ServerSubHandles**

服务器使用此属性来标识要轮询的订阅。

**4.3.4.4 SubscriptionPolledRefreshResponse****变量结构****SubscriptionPolledRefreshResponse**

*DataBufferOverflow*

**SubscriptionPolledRefreshResult**

*RcvTime*

*ReplyTime*

*ClientRequestHandle*

*RevisedLocaleID*

*ServerState*

**InvalidServerSubHandles****RItemList**

*SubscriptionHandle*

**Items**

*ValueTypeQualifier*

*ItemPath*

*ItemName*

*ClientItemHandle*

*Timestamp*

*ResultID*

**DiagnosticInfo****Value****Quality**

*QualityField*

*LimitField*

*VendorField*

**Errors**

*ID*

**Text**

## 元素和属性

### **SubscriptionPolledRefreshResponse**

该元素包含服务器将其作为对 **SubscriptionPolledRefresh** 请求的响应发送到客户端的所有信息。

### **DataBufferOverflow**

如果该属性的值为 TRUE，则数据项中的更改将会因缺少资源而无法保存。各数据项指示它们是否因缺少资源而受到影响。

### **SubscriptionPolledRefreshResult**

该元素包含有关服务器响应的基本信息。它属于 **ReplyBase** 类型。有关该元素属性的信息，请参见本文档中的相关部分。

### **InvalidServerSubHandles**

**ServerSubHandles** 被服务器视为无效。

### **RItemList**

如果客户端通过 **SubscriptionPolledRefresh** 请求传送的 **ReturnAllItems** 属性的值为 TRUE，**<RItemList>** 元素将包含所有数据项的值。否则，服务器只返回已更改的值。

### **SubscriptionHandle**

客户端在订阅调用中指定的 **RItemList** 的句柄。

### **Items**

指定单个数据项所使用的元素。

该元素属于 **ItemValue** 类型。有关该元素属性的信息，请参见本文档中的相关部分。

### **Errors**

与此响应一起出现的错误的数组。

该元素属于 **OPCError** 类型。有关该元素属性的信息，请参见本文档中的相关部分。

## 4.3.4.5 **SubscriptionCancel**

### 变量结构

#### **SubscriptionCancel**

*ServerSubHandle*

*ClientRequestHandle*

## 元素和属性

### **SubscriptionCancel**

客户端使用 `<SubscriptionCancel>` 元素结束订阅，并且服务器可释放相应的资源。

#### **ServerSubHandle**

由服务器分配给该元素的值标识发送请求的订阅。

#### **ClientRequestHandle**

如果该属性包含在客户端的请求中，则将由服务器返回并带有

#### **SubscriptionCancelResponse**

在庞大而复杂的系统中，该信息有助于客户端为请求分配响应。

指定该属性为可选操作。

### 4.3.4.6 **SubscriptionCancelResponse**

## 变量结构

### **SubscriptionCancelResponse**

*ClientRequestHandle*

## 元素和属性

### **SubscriptionCancelResponse**

服务器使用该元素确认客户端的 `SubscriptionCancel` 请求。

#### **ClientRequestHandle**

如果客户端在其请求中使用该属性，服务器会在响应中将其返回。

## 4.3.5 更多查询

### 4.3.5.1 Browse

#### 变量结构

##### **Browse**

*LocaleID*  
*ClientRequestHandle*  
*ItemPath*  
*ItemName*  
*ContinuationPoint*  
*MaxElementsReturned*  
*BrowseFilter*  
*ElementNameFilter*  
*VendorFilter*  
*ReturnAllProperties*  
*ReturnPropertyValues*  
*ReturnErrorText*  
**PropertyNames**

#### 元素和属性

##### **Browse**

<Browse> 元素包含在层级地址空间中进行浏览所必需的所有信息。

##### **LocaleID**

一个可选属性，客户端通过该属性为某些返回数据指定语言。

##### **ClientRequestHandle**

如果该属性包含在客户端的请求中，将由服务器返回并带有响应。

在庞大而复杂的系统中，该信息有助于客户端为请求分配响应。指定该属性为可选操作。

##### **ItemPath**

在地址空间中进行浏览的起始数据项的路径。

如果有第二次浏览请求，该属性的值须与上一个请求的值相同。

##### **ItemName**

在地址空间中进行浏览的起始数据项的名称。如果有第二次浏览请求，该属性的值须与上一个请求的值相同。

### **ContinuationPoint**

如果不是初始浏览请求，可在此指定一个点作为重新启动浏览请求的起点。

### **MaxElementsReturned**

返回值的最大数。

如果在此传递负值，将发生异常。

### **BrowseFilter**

枚举数据类型为

`browseFilter (all、branch、item)` 的该属性用于指定将要返回的元素子集。

### **ElementNameFilter**

用于选择元素的常规表达式。

### **VendorFilter**

用于选择供应商特定信息的供应商特定的表达式。未定义对 **ElementNameFilter** 元素的影响。

### **ReturnAllProperties**

如果该属性的值设为 TRUE，服务器将为每个返回的元素返回所有可用属性。

在这种情况下，**PropertyNames** 属性值随后会被忽略。

### **ReturnPropertyValue**

如果该属性的值设为 TRUE，则服务器不仅会返回属性名称，也会返回属性值。

### **ReturnErrorText**

如果该属性的值为 TRUE，服务器将返回详细的错误描述。

### **PropertyNames**

与每个元素一起返回的属性的名称。如果 **ReturnAllProperties** 属性的值设为 TRUE，则无论 **PropertyNames** 如何，服务器都将返回所有属性。

#### 4.3.5.2 BrowseResponse

##### 变量结构

```
BrowseResponse
  ContinuationPoint
  MoreElements
  BrowseResult
    RcvTime
    ReplyTime
    ClientRequestHandle
    RevisedLocaleID
    ServerState
  Elements
    Name
    Item Path
    ItemName
    IsItem
    HasChildren
  Properties
    Name
    Description
    ItemPath
    ItemName
    ResultID
    Value
  Errors
    ID
    Text
```

##### 元素和属性

###### BrowseResponse

包含浏览请求响应的所有信息。

###### ContinuationPoint

在此，服务器可以指定后续浏览请求的起始点。SIMATIC NET 不支持该属性。

**MoreElements**

如果服务器不支持 ContinuationPoint 属性或者返回值的数量超出了 MaxElementsReturned 值，服务器会将该属性设为 TRUE。

**BrowseResult**

<BrowseResult> 元素包含有关服务器响应的基本信息。

该元素属于 ReplyBase 类型。有关该元素属性的信息，请参见本文档中的相关部分。

**Elements**

包含有关结构树的解析元素的信息。

**Name**

命名空间中已显示给用户（例如，在结构视图中）的元素的名称。

**ItemPath**

元素的路径。SIMATIC NET 不支持该属性。

**ItemName**

此元素可以通过 ItemName 属性在服务器的命名空间中进行唯一标识。

**IsItem**

如果将该属性的值设为 TRUE，则该元素是可用于读取、写入和订阅请求的数据项。

**HasChildren**

如果将该属性的值设为 TRUE，则相应元素将具有子元素。

**Properties**

<Properties> 元素包含有关可通过 Browse 或 GetProperties

调用进行访问的属性的信息。

该元素属于 ItemProperty 类型。有关该元素属性的信息，请参见本文档中的相关部分。

**Errors**

与此请求一起出现的错误的数组。

该元素属于 OPCError 类型。有关该元素属性的信息，请参见本文档中的相关部分。

### 4.3.5.3 GetProperties

#### 变量结构

##### GetProperties

*LocaleID*  
*ClientRequestHandle*  
*ItemPath*  
*ReturnAllProperties*  
*ReturnPropertyValues*  
*ReturnErrorText*  
**ItemIDs**  
*ItemPath*  
*ItemName*  
**PropertyNames**

#### 元素和属性

##### GetProperties

使用 **<GetProperties>** 元素，您可以查询属性。

##### LocaleID

一个可选属性，客户端通过该属性为某些返回数据指定语言。

##### ClientRequestHandle

如果该属性包含在客户端的请求中，将由服务器返回并带有响应。

在庞大而复杂的系统中，该信息有助于客户端为请求分配响应。指定该属性为可选操作。

##### ItemPath

元素的路径。SIMATIC NET 不支持该属性。

##### ReturnAllProperties

如果该属性的值设为 TRUE，服务器将返回每个已返回元素的所有属性。

这种情况下，PropertyNames 元素的值将被忽略。

##### ReturnPropertyValues

如果该属性的值设为 TRUE，则服务器不仅会返回属性名称，也会返回属性值。

##### ReturnErrorText

如果该属性的值为 TRUE，服务器将返回详细的错误描述。

##### ItemIDs

包含属性待查询的数据项的列表。

**ItemPath**

元素的路径。SIMATIC NET 不支持该属性。

**ItemName**

此元素可以通过 **ItemName** 属性在服务器的命名空间中进行唯一标识。

**PropertyNames**

与每个元素一起返回的属性的名称。如果 **ReturnAllProperties** 属性的值设为 TRUE，则无论 **PropertyNames** 如何，服务器都将返回所有属性。

#### 4.3.5.4 GetPropertiesResponse

##### 变量结构

**GetPropertiesResponse****GetPropertiesResult***RcvTime**ReplyTime**ClientRequestHandle**RevisedLocaleID**ServerState***PropertyLists***ItemPath**ItemName**ResultID***Properties***Name**Description**ItemPath**ItemName**ResultID**Value***Errors***ID**Text*

## 元素和属性

### **GetPropertiesResponse**

服务器将通过该元素响应 GetProperties 请求。

### **GetPropertiesResult**

<GetPropertiesResult> 元素包含有关服务器响应的基本信息。

该元素属于 ReplyBase 类型。有关该元素属性的信息，请参见本文档中的相关部分。

### **PropertyLists**

每个数据项都会返回此类型的元素，元素中包含数据项的请求属性。

*ItemPath*、*ItemName* 和 *ResultID* 属性可用于 PropertyLists 和 Properties 元素。如果在 Properties 元素中指定这些属性，则为该元素覆盖在 PropertyLists 中具有相同名称的属性的值。

### **ItemPath**

元素的路径。SIMATIC NET 不支持该属性。

### **ItemName**

此元素可以通过 ItemName 属性在服务器的命名空间中进行唯一标识。

### **ResultID**

如果发生错误或非关键例外，该属性将包含 OPC 错误的名称。

### **Properties**

该元素属于 ItemProperty 类型。有关该元素属性的信息，请参见本文档中的相关部分。

### **Errors**

与此响应一起出现的错误的数组。

该元素属于 OPCError 类型。有关该元素属性的信息，请参见本文档中的相关部分。

## 4.3.5.5 **GetStatus**

### 变量结构

#### **GetStatus**

*LocaleID*

*ClientRequestHandle*

## 元素和属性

### GetStatus

**GetStatus** 请求将检查 Web 服务并获取供应商特定的信息（其它 OPC 方法无法访问该信息）。

### LocaleID

一个可选属性，客户端通过该属性为某些返回数据指定语言。

### ClientRequestHandle

如果该属性包含在客户端的请求中，将由服务器返回并带有响应。

在庞大而复杂的系统中，该信息有助于客户端为请求分配响应。

指定该属性为可选操作。

## 示例

```
<soap:Body>
    <GetStatus
        LocaleID="de-AT"
        xmlns="http://opcfoundation.org/webservices/XMLDA/1.0/"
    />
</soap:Body>
```

## GetStatus web 方法的 C# 接口

使用 C# 用户程序时，可以使用以下方法检查 Web 服务：

```
ServerStatus GetStatus(string LocaleID,
    string ClientRequestHandle,
    out ServerStatus);
```

#### 4.3.5.6 GetStatusResponse

##### 变量结构

```
GetStatusResponse
  GetStatusResult
    RcvTime
    ReplyTime
    ClientRequestHandle
    RevisedLocaleID
    ServerState
  Status
    StartTime
    ProductVersion
  StatusInfo
  VendorInfo
  SupportedLocaleIDs
  SupportedInterfaceVersions
```

##### 元素和属性

###### BrowseResponse

包含 GetStatus 请求响应的所有信息

###### GetStatusResult

<GetStatusResult> 元素包含有关服务器响应的基本信息。

该元素为 ReplyBase 类型。有关该元素属性的信息，请参见本文档中的相关部分。

###### Status

<Status> 元素包含有关服务器状态的信息。

###### StartTime

服务器启动时的时间（UTC 格式）。

###### ProductVersion

由主版本号、次版本号和内部版本号组成的服务器版本号。

###### StatusInfo

包含有关服务器状态的附加信息，与语言相关。

#### **VendorInfo**

带有服务器附加信息的供应商特定字符串。

该属性的值应包括公司名称和支持设备的信息。

#### **SupportedLocaleIDs**

服务器支持的区域设置。 服务器不需要支持所有数据项的全部区域设置。

#### **SupportedInterfaceVersions**

包含服务器支持的 XMLDA 规范版本的字符串数组。

## 4.4 对 OPC UA 接口进行编程

### 本部分包含哪些内容？

本部分将介绍 OPC UA 服务器的编程接口、扩展组态以及证书管理。

#### 接口

既有用于从 C 程序发起简便、快速访问的 C 接口，也有便于在 .NET 应用程序中使用的 .NET 接口。安装 SIMATIC NET PC 软件后，可使用相应的库和程序集。

#### OPC UA 接口在 SIMATIC NET 下的可用性

从 2008 (V7.1) 版本开始，SIMATIC NET CD 提供 OPC UA 接口。

#### 对 OPC UA 服务器进行定位

OPC UA 服务器使用 OPC UA 发现服务进行定位。只需要输入计算机名称或 IP 地址以及发现端口 4840。现有已注册的 OPC UA 服务器随后将返回其组态的 OPC UA 端点。

#### 安全和验证

为了在 OPC UA 客户端与服务器之间实现安全通信 (OPC UA CreateSecureChannel)，X509 证书必须在两端都可用。  
证书在通信开始时就会进行交换。OPC UA 服务器和客户端此时将决定是否信任提供的证书。

## 通信

接受证书之后，可以建立和激活 OPC UA 会话。

可以对 OPC UA 节点进行监视（OPC UA MonitoredItems 和 Subscriptions），或者仅仅执行一次读取或写入操作。

如果节点需要多次读取或写入，则为了提高速度，强烈建议先注册这些节点 (OPC UA RegisterNode)。读取或写入时，推荐进行这一操作。

## 透明冗余

UA 客户端自动化数据的高可用性（无需使用附加软件）通过透明、冗余的 S7 UA 服务器实现。对于冗余和负载平衡，只需要使用“工业以太网”标准。

## 使用 IndexRange 访问 OPC UA 中的数组元素

使用 IndexRange 进行访问时，视目标变量而定，可能会受到限制。

由于一致性的原因，使用 IndexRange 将位数组的某些部分写入 S7 块的操作会被拒绝。

仅支持一维 IndexRange。使用 IndexRange

进行读取操作时，即使通信伙伴请求更多的变量，也总是会读取完整的变量。

### 4.4.1 对 OPC UA 服务器进行组态

#### 4.4.1.1 验证

## 用户验证

安装之后，为了进行测试和简单的调试，已将验证取消激活。

不过，我们建议激活用户验证。

在“通信设置”组态程序中，应该使用“OPC 设置 > OPC 协议选择 > 单击相关协议的箭头符号 > 允许匿名登录 OPC UA 服务器”(OPC settings > OPC protocol selection > click the arrow symbol of the relevant protocol > allow anonymous logon with OPC UA server)，取消激活匿名登录 OPC UA 服务器。  
也可以在这里再次激活。

#### 4.4 对 OPC UA 接口进行编程



完成此项设置之后，在安全连接建立期间将查询现有的 Windows 用户及其密码。  
用户名和密码根据客户端特定的步骤指定。

##### 4.4.1.2 端点安全

###### OPC UA 服务器的端点安全

应在“通信设置”组态程序的“OPC 协议选择”(OPC protocol selection) 中取消激活端点安全“无”(None) (无安全检查或加密) 和“允许到 OPC UA 服务器的非安全连接”(allow non-secure connections to OPC UA server)。随后，就只有安全端点可以使用。也可以在这里再次激活。

#### 4.4.2 OPC UA 服务器的证书管理

##### 证书管理文件夹

OPC UA 服务器的证书管理基于 OpenSSL  
(<http://www.openssl.org/>)，位于下列文件夹中：

- Windows 7/Windows Server 2008 R2 和 Windows 8/Windows Server 2012  
文件系统中的隐藏路径：  
"`<systemdrive>:\ProgramData\Siemens\OPC\PKI\CA\`"

##### OPC UA 服务器证书

上述文件夹中的子文件 "`\certs`" 是存放计算机特定的 OPC UA 服务器证书（以 X509 格式，具有扩展名“`der`”）的位置

- S7 UA: `OPC.SimaticNET.S7.der` `OPC.SimaticNET.S7.der`  
`OPC.SimaticNET.S7.der`, 包含公钥,
- S7OPT UA: `OPC.SimaticNET.S7OPT.der`, 包含公钥,
- PROFINET IO UA: `OPC.SimaticNET.PNIO.der`, 包括公钥,
- DP UA: `OPC.SimaticNET.DP.der`, 包括公钥,
- SEND/RECEIVE UA: `OPC.SimaticNET.SR.der`, 包括公钥。

这些证书将在通信开始时传送到客户端。取决于 OPC UA  
客户端，它们可以或者必须导入 OPC UA 客户端的证书管理。

---

##### 说明

还应该注意，计算机名称中只能使用标准字符，因此证书中也是如此。

标准名称由字母（A-Z、a-z）、数字（0-9）和（-）组成。  
如果在计算机名称中使用其它字符，将无法生成证书。

---

---

##### 说明

如果安装“SIMATIC NET PC 软件”后更改了计算机名称，将导致为 OPC UA  
安装的证书失效，OPC UA 将无法再运行。解决方法：在“通信设置”程序“OPC UA  
证书”(OPC UA certificates) 中，通过快捷菜单“重新创建 OPC UA 组态”(Recreate OPC  
UA configuration) 创建新的 UA 组态。

---

## OPC UA 服务器的 UA 客户端证书

上述文件夹包含存放可区分的计算机特定 OPC UA

客户端证书的位置，也是在“\certs”子文件夹中。例如，对于本地安装的 OPC Scout V10，则是包括公钥的证书“OPC.ScoutV10.der”。

此证书将在通信开始时传送到 OPC UA 服务器。此时，该证书必须存在于 OPC UA 服务器的证书管理中，并与通信开始时 OPC UA 客户端传送的证书进行比较。

如果没有此证书，将拒绝安全通信。

## OPC UA 服务器的 UA 撤销证书

可以在可组态的“\crl”子文件夹中创建不信任证书列表。

与此列表中的证书建立安全连接的请求会被拒绝。

“examplecrl.crl”列表已经预安装，并且可以进行调整。

可使用 OpenSSL 工具创建这样的列表

（请参见 <http://www.openssl.org/docs/apps/x509.html>）。

## SIMATIC NET S7 UA 服务器名称

服务器名称以及服务器 URI 如下所示：

```
<服务器 URI>urn:Siemens.Automation.SimaticNET.S7:(%guid%)</服务器 URI>
```

```
<ServerName>OPC.SimaticNET.S7</ServerName>
```

“%guid%”是服务器的唯一标识符。它由 28 个十六进制数组成，格式为 ABCDEF01-2345-6789-0123456789AB。

请勿更改此值。

## SIMATIC NET S7OPT UA 服务器名称

服务器名称以及服务器 URI 如下所示：

```
<服务器 URI>urn:Siemens.Automation.SimaticNET.S7OPT:(%guid%)</服务器 URI>
```

```
<服务器名称>OPC.SimaticNET.S7OPT</服务器名称>
```

“%guid%”是服务器的唯一标识符。它由 28 个十六进制数组成，格式为 ABCDEF01-2345-6789-0123456789AB。

请勿更改此值。

## SIMATIC NET PROFINET IO UA 服务器名称

服务器名称以及服务器 URI 如下所示:

```
<服务器 URI>urn:Siemens.Automation.SimaticNET.PNIO:(%guid%)</服务器 URI>
```

```
<ServerName>OPC.SimaticNET.PNIO</ServerName>
```

“%guid%”是服务器的唯一标识符。它由 28 个十六进制数组成，格式为 ABCDEF01-2345-6789-0123456789AB。

请勿更改此值。

## SIMATIC NET DP-UA 服务器名称

服务器名称以及服务器 URI 如下所示:

```
<服务器 URI>urn:Siemens.Automation.SimaticNET.DP:(%guid%)</服务器 URI>
```

```
<ServerName>OPC.SimaticNET.DP</ServerName>
```

“%guid%”是服务器的唯一标识符。它由 28 个十六进制数组成，格式为 ABCDEF01-2345-6789-0123456789AB。

请勿更改此值。

## SIMATIC NET SEND/RECEIVE UA 服务器名称

服务器名称以及服务器 URI 如下所示:

```
<服务器 URI>urn:Siemens.Automation.SimaticNET.SR:(%guid%)</服务器 URI>
```

```
<ServerName>OPC.SimaticNET.SR</ServerName>
```

“%guid%”是服务器的唯一标识符。它由 28 个十六进制数组成，格式为 ABCDEF01-2345-6789-0123456789AB。

请勿更改此值。

## 参见

使用图形用户界面进行证书管理 (页 597)

#### 4.4.3 OPC UA 客户端“OPC Scout V10”中的证书管理

##### 对 OPC UA 客户端“OPC Scout V10”的证书管理进行组态

提供的 OPC UA 客户端“OPC Scout V10”使用 Windows 证书管理。

##### 如何管理计算机的证书:

1. 以管理员身份登录系统。
2. 单击“开始 > 运行”(Start > Run)。  
输入框随即打开。
3. 键入“mmc”，然后单击“确定”(OK)。  
控制台打开。
4. 在“文件”(File) 菜单中，选择“添加/移除管理单元...”(Add/ Remove Snap-in...), 然后选择“添加”(Add)。  
选择管理单元的对话框随即打开。
5. 在“管理单元”(Snap-in) 列表中双击“证书”(Certificates)。  
在“证书管理单元”(Certificates snap-in) 对话框中选择“计算机帐户”(Computer account) 选项，然后单击“下一步”(Next)。
6. 在“选择计算机”(Select computer) 对话框中执行以下操作之一：
  - 要管理本地计算机的证书，请选择“本地计算机”(Local computer) 选项，然后单击“完成”(Finish)。
  - 要管理远程计算机的证书，请选择“其它计算机”(Other computer) 选项，输入计算机名称（或通过单击“浏览”(Browse) 选择），然后单击“完成”(Finish)。
7. 单击“关闭”(Close)。  
“证书 (计算机名称)”(Certificates (computer name))  
条目将显示在新控制台选定管理单元的列表中。
8. 如果希望向控制台添加更多管理单元，请单击“确定”(OK)。

9. 要保存此控制台，请在控制台中选择“文件”(File)菜单和“保存”(Save)，并且如果有必要，输入一个名称。
10. 关闭控制台。

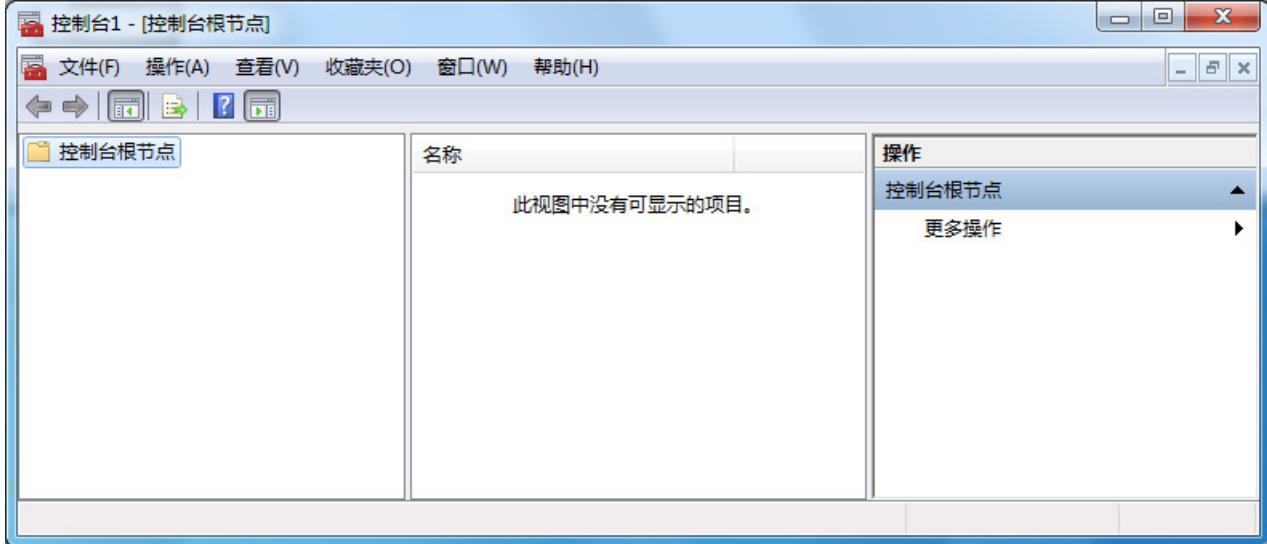


图 4-10 控制台窗口：证书管理

可以在“OpcScoutV10”下“UA 应用程序”(UA Applications) 中的“本地计算机”(Local computer) 区域列出和浏览 OPC Scout V10。

如何从客户端计算机的证书管理中导出证书：

要与远程 SIMATIC NET OPC UA

服务器进行必要的安全通信，必须先从客户端计算机的证书管理中导出该证书。

1. 打开控制台。

“操作 > 所有任务 > 导出...”(Action > All tasks > Export...) 菜单

2. 在没有私钥的情况下导出证书。



图 4-11 在没有私钥的情况下导出证书

3. 以“DER 二进制编码 X.509”格式导出证书。

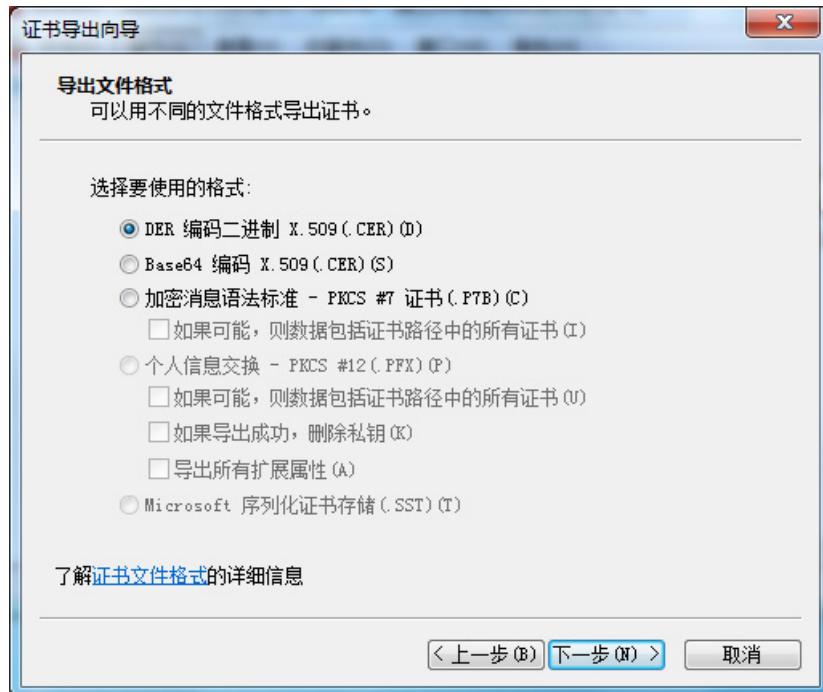


图 4-12 以“DER”格式导出证书

4. 选择所需的格式后，单击“下一步”(Next)。

请记住，下一对话框中的文件扩展名必须为“DER”。

将文件扩展名从“\*.cer”更改为“\*.der”。

5. 单击“下一步”(Next) 和“完成”(Finish)，完成导出。

该导出文件所处的文件夹与服务器计算机证书所处的文件夹名称相同。

#### 说明

##### “\*.der”格式的证书文件

如果创建 DER

编码的证书文件，但意外留下了“cer”扩展名，则需要将客户端计算机的文件夹扩展名从“\*.cer”更改为“\*.der”。

## 如何将证书复制到远程服务器:

要与远程 **SIMATIC NET OPC UA**

服务器进行必要的安全通信，现在必须将该证书导入远程 **OPC UA** 服务器的证书管理：

1. 为此，请将刚导出的客户端证书文件复制到合适的数据介质中。

2. 从该数据介质将客户端的证书文件复制到 **OPC UA**

服务器计算机上的相关文件夹中（请参见“**OPC UA** 服务器的证书管理  
(页 589)”部分）。

现在，可以建立客户端计算机到 **OPC UA** 服务器的安全连接。

---

### 说明

在证书中对计算机名称（主机名称）进行编码。

请注意，如果在安装证书之后更改计算机名称，这些证书将无效。之后将无法进行 **OPC UA** 通信。不过，可以重新创建 **OPC UA** 组态，请参见“使用图形用户界面进行证书管理  
(页 597)”部分。

---

---

### 说明

还应该注意，计算机名称中只能使用标准字符，因此证书中也是如此。

标准名称由字母（A-Z、a-z）、数字（0-9）和（-）组成。

如果在计算机名称中使用其它字符，将无法生成证书。

---

从 V8.0 版本的 **SIMATIC NET PC** 软件开始，**OPC UA** 服务器的证书管理也可用于进入 **UA** 客户端证书。在“通信设置”组态程序的图形用户界面中可以接受或拒绝这些证书。

远程服务器的客户端证书不再需要提前通过导出和复制进行手动传输。

只有在第一个连接建立期间需要接受客户端时，才需要这一操作。

#### 4.4.4 使用图形用户界面进行证书管理

从 V8.0 版本的 SIMATIC NET PC

软件开始，在“通信设置”组态程序中可以使用图形用户界面对 OPC UA 服务器进行证书管理。

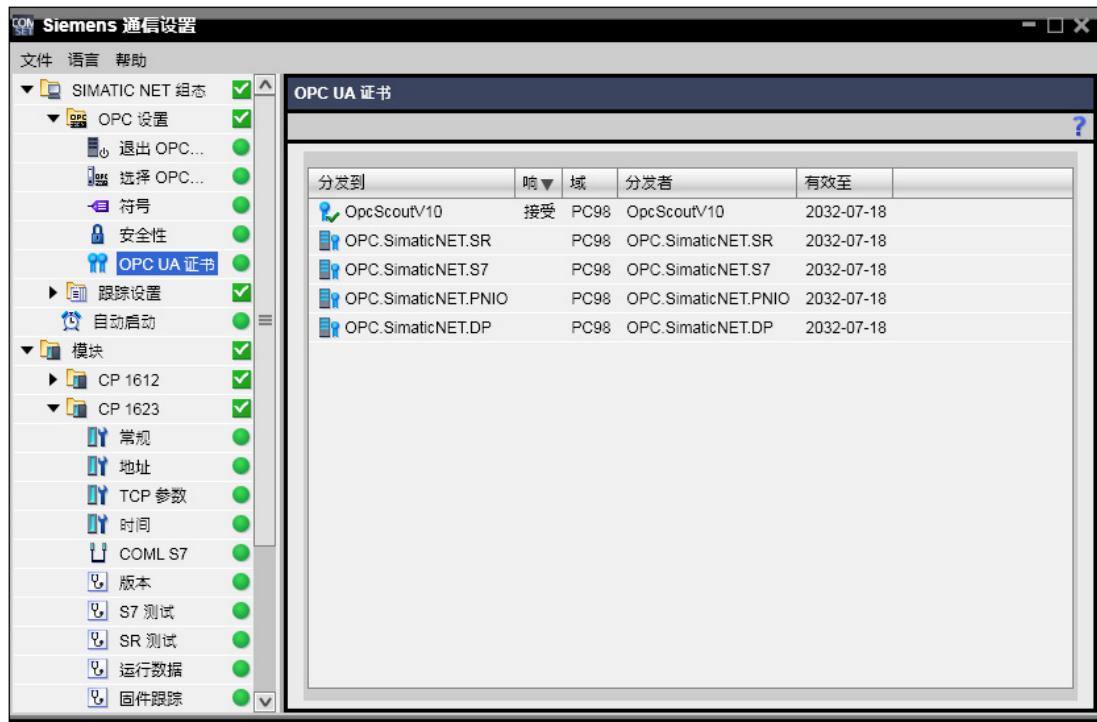


图 4-13 “通信设置”组态程序中的证书管理

此处可以显示和管理本地 OPC UA 服务器的证书以及 OPC UA 客户端用于向服务器标识自身的证书。

也显示带有软件组件（用于签发证书）的计算机的名称。

还可显示证书自身的有效期和内容。能够以 DER 二进制编码文件 (X.509) 形式导入客户端证书和导出服务器证书。

#### 4.4 对 OPC UA 接口进行编程

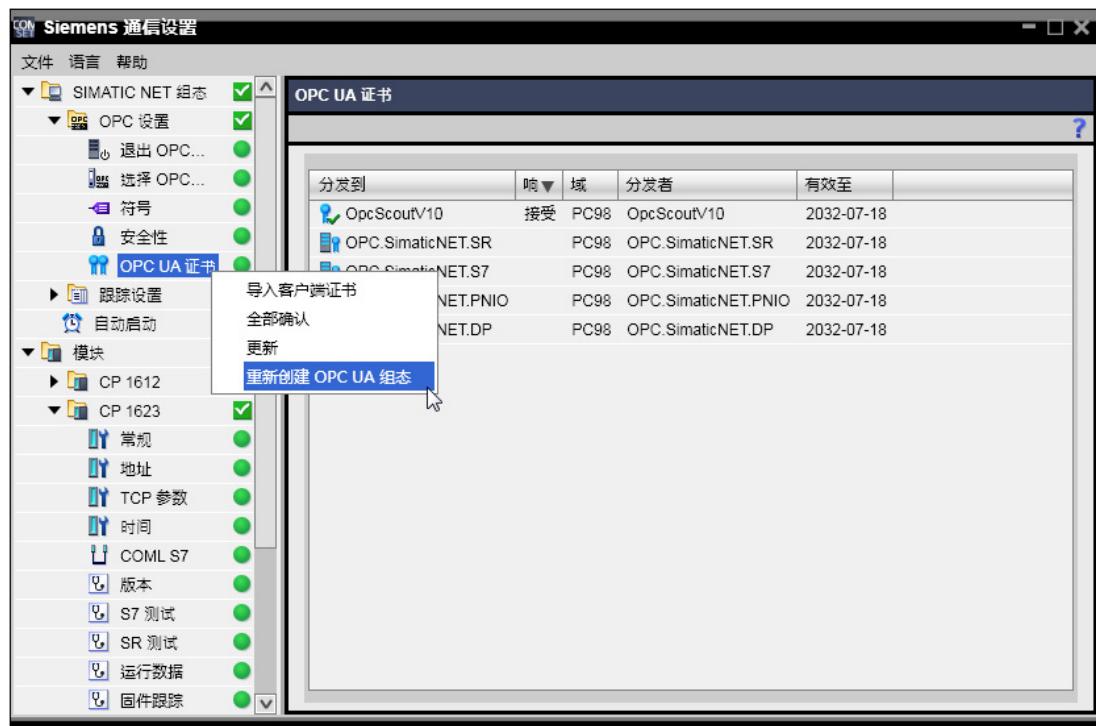


图 4-14 在“通信设置”组态程序中对证书进行组态

例如，如果计算机名称更改，则可以重新创建本地 OPC UA 组态。这将包括以下方面：

- 重新创建 S7、优化 S7、PROFINET IO、DP 和 SEND/RECEIVE 本地 OPC UA 服务器的组态文件（根据计算机名称进行调整）。
- 删除并重新创建 OPC UA 服务器证书。
- 删除并重新创建本地 OPC Scout V10 的 OPC UA 客户端证书。
- 从 Windows 证书存储器中重新导出 OPC UA 发现服务的证书。
- 退出并重新启动 OPC 服务器。

#### 说明

因此，使用该功能之前应关闭任何活动的 SIMATIC NET 通信。

#### 4.4.5 OPC UA 服务

##### UA 服务

OPC UA 服务分为多个逻辑功能组。本部分将介绍最重要的服务和方法。

有关详细信息，请参见 "<http://www.opcfoundation.org/UA>" 上的 OPC UA 规范。

##### 发现服务

使用发现服务，UA 客户端可以获取有关 UA 服务器的组态端点和安全要求的信息：

- **FindServers**

此服务将返回发现服务器已知的 OPC UA 服务器。

- **GetEndpoints**

此服务将返回服务器的端点以及连接到安全通道和会话的安全要求。

##### 安全服务

安全服务允许建立安全通信通道，这样可确保服务器与客户端之间交换信息的完整性和保密性。

- **OpenSecureChannel**

此服务用于打开和更新安全通道。客户端将与服务器交换证书。

有关详细信息，请参见 "<http://www.opcfoundation.org/UA>" 下的 "Downloads" > "Specifications" > "Part 6"。

- **CloseSecureChannel**

此服务用于关闭安全通道。

## 会话服务

会话服务允许客户端对想要使用该会话的用户进行验证。也可以对会话进行管理。

- **CreateSession**

OPC UA 客户端使用该服务创建会话。

服务器将提供两个用于对会话进行唯一标识的值：

- 第一个值是“**SessionId**”，用于在协议文件和命名空间中对会话进行唯一标识。
- 第二个值是“**AuthenticationToken**”，用于向会话分配进入查询。

在“**CreateSession**”之前，应调用“**OpenSecureChannel**”来创建一个安全通道。

该安全通道应与“**AuthenticationToken**”关联，以便只有该会话处理的查询才可与安全通道生成的会话关联。

- **ActivateSession**

客户端使用该服务向服务器传送证书和会话用户的标识。

安全通道的证书也应与会话中使用的证书相同。

客户端和服务器应对此进行检查并确保无误。

- **CloseSession**

此服务用于关闭会话

- **Cancel**

此服务用于中止未决的查询。

## 查看服务

可以使用这些服务浏览命名空间。可以注册待写入或重复读取的节点。

- **Browse**

此服务用于读出对节点的引用。

- **BrowseNext**

如果回复与简单回复帧不相符，则此服务用于请求下一组“**Browse**”或“**BrowseNext**”回复。

- **TranslateBrowsePathsNodeid**

此服务用于转换 **NodeId** 的浏览路径。

每一个浏览路径都以起始节点和相对路径开始。

相对路径包括一系列引用类型和浏览名称。

- RegisterNodes

如果多次或重复访问节点（例如写入或读取值），则应使用“**RegisterNodes**”服务。

此服务允许服务器准备进行进一步访问以及使后续访问更为有效。这将显著提升 OPC UA 服务器的性能。有关 **RegisterNodes**

使用的详细信息，请参见“<http://www.opcfoundation.org/UA>”的 OPC UA 规范。

- UnregisterNodes

此服务用于取消注册已注册的节点。

## 属性（变量）服务

属性服务用于读取和写入节点的属性。

由于变量值建模为属性，所以这些服务都可用于读取和写入变量值。

- Read

利用此服务，可以从一个或多个节点中读取一个或多个属性。

也可以从数组中分别读取数组元素或范围。

可以按区域将结构化属性元素编入索引或执行读取操作。

“maxAge”参数命令服务器从设备或本地存储的副本中读取不早于 **maxAge** 的数据。

- Write

此服务用于写入一个或多个节点的一个或多个属性。

也可以在数组中分别写入数组元素或范围。

可以按区域将结构化属性元素编入索引或执行写入操作。

## 方法服务

此服务用于调用方法。可以传送输入参数和接收输出参数。

- 调用

此服务用于调用一组方法。

## 注册服务

注册服务允许客户端生成、修改和删除订阅。订阅会将通知（由 **MonitoredItems** 生成）发送到客户端。

- **CreateSubscription**

此服务用于创建订阅。这些订阅可监视通知的多个 **MonitoredItem**，并将它们传递至客户端，作为对发布作业的响应。

- **ModifySubscription**

此服务用于修改订阅。

- **SetPublishingMode**

此服务可激活一个或多个订阅的通知发送。

- **Publish**

此服务具有两种用途：

- 第一，可用于确认收到一个或多个订阅的通知。
- 第二，可用于从服务器请求通知或“保持激活状态”。

- **Republish**

此服务可从订阅请求重复通知。

- **TransferSubscription**

利用此服务，可以将订阅及其 **MonitoredItem** 从一个会话传递到另一个会话。  
在客户端内和客户端之间都可以进行这一操作。

- **DeleteSubscription**

此服务用于删除订阅。

## 监视服务

监视服务与注册服务一起用于注册命名空间内节点的监视。

监视服务定义用于生成、修改和删除数据项列表（该列表用于监视数据更改属性或事件对象）的服务。

- **CreateMonitoredItems**

使用此服务，可以向订阅分配一个或多个 **MonitoredItem**。

- **ModifyMonitoredItems**

使用此服务，可以修改订阅的一个或多个 **MonitoredItem**。

- SetMonitoringMode

此服务可用于设置订阅一个或多个 **MonitoredItem** 的监视模式。

- DeleteMonitoredItems

使用此服务，可以删除订阅的一个或多个 **MonitoredItem**。

## 4.4.6 创建 OPC UA 客户端

### 4.4.6.1 OPC UA 下的接口

#### OPC UA 中的编程接口

要创建 OPC UA 客户端，应当根据应用程序和要求的证书管理选择编程接口：

- C 接口可以对 OpenSSL 证书管理进行简单、高速的访问
- .NET 接口方便使用，可访问 Windows 证书管理

---

#### 说明

有关相关示例程序，请参见“以 C 编写的 OPC UA 接口 (页 849)”部分。

---

### 4.4.6.2 OPC UA 的 C 接口

#### C 接口

OPC UA C 接口适用于 OPC UA 客户端应用程序，必要时可以轻松地移植到其它系统。

安装完成后，可以在以下文件夹中找到需要的头文件：

“<Installationprogrampath>\SIMATIC.NET\opc2\inc”

要求的导入库“uastack.lib”以及可装载库“uastack.dll”都位于以下文件夹中：

“<Installationprogrampath>\SIMATIC.NET\opc2\bins7\”

“<Installationprogrampath>\SIMATIC.NET\opc2\binpnio\”

“<Installationprogrampath>\SIMATIC.NET\opc2\bindp\”

“<Installationprogrampath>\SIMATIC.NET\opc2\binsr\”

“<安装程序路径>\SIMATIC.NET\opc2\bins7opt\”

## 4.4 对 OPC UA 接口进行编程

处理证书也需要 OpenSSL 导入库“libeay32.lib”和可装载库“libeay32.dll”。  
也可参见上述目录部分。

这样便可创建一个安全的 OPC UA C 客户端应用程序。

### 说明

有关实施的详细信息，请使用 OPC 基金会的实施示例。

### 4.4.6.3 OPC UA 的 .NET 接口

#### .NET 接口

OPC UA .NET 接口适用于简单、方便、可编程的 OPC UA 客户端应用程序。

例如，OPC Scout V10 可使用 UA .NET 接口。

安装完成后，可以在以下文件夹中找到需要的程序集文件：

“<Installationprogrammpath>\SIMATIC.NET\opc2\bin\”

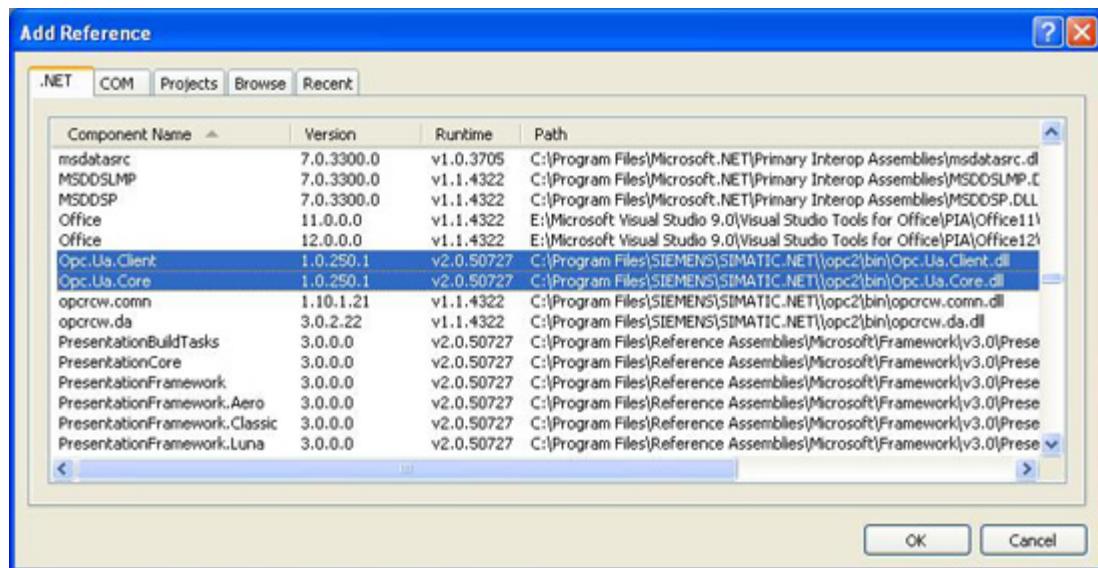


图 4-15 .NET 接口的程序集文件

需要以下组件：

- UA 客户端组件“Opc.Ua.Client.dll”
- UA 核心组件“Opc.Ua.Core.dll”

还可以使用 OPC 基金会文件夹中相应的 OPC UA 组件，这些组件通过 SIMATIC NET CD 安装。您可在以下目录中找到这些组件：

“<安装程序路径>\OPC Foundation\UA\V1.0\Bin”

#### 4.4.7 OPC UA 服务器的消息

##### 系统消息

SIMATIC NET OPC UA 服务器可以输出系统消息，这些输出包括接口上的错误代码和 OPC Scout V10 中的符号 ID。

下表列出了最重要的系统信息。

可以在 OPC UA 的头文件或 OPC 基金会（“<http://opcfoundation.org/UA>”）规范的第 4 部分和第 6 部分中找到更多消息。

表格 4- 1 SIMATIC NET OPC UA 服务器的系统消息

符号 ID	错误代码（十六进制）	含义
Good	0x00000000	良好 - 一切都在预期之中
Good_Overload	0x002F0000	由于出现资源瓶颈，当前的数据值记录减速。
Good_Clamped	0x00300000	已接受并写入该值。 传感器（数据目标）超出允许的设置范围。
Uncertain	0x40000000	不确定 - 无更多详细信息
Bad	0x80000000	无效 - 无更多详细信息
Bad_NoCommunication	0x80310000	已定义通信端点。 通信本身未建立且没有上一个已知值。 状态/子状态与接收第一个值之前的缓存值相关。 。
Bad_NodeIdInvalid	0x80330000	NodeId 的语法无效。
Bad_NodeIdUnknown	0x80340000	该节点的 ID 引用了服务器地址空间中不存在的节点。
Bad_NodeClassInvalid	0x805F0000	节点的值无效。
Bad_SourceNodeIdInvalid	0x80640000	源节点的 ID 未引用一个有效节点。
Bad_TargetNodeIdInvalid	0x80650000	目标节点未指定一个有效节点。

符号 ID	错误代码（十六进制）	含义
Bad_NoDeleteRights	0x80690000	服务器不允许删除节点。
Bad_HistoryOperationInvalid	0x80710000	“历史记录详细信息”参数（历史记录的详细信息）无效。
Bad_HistoryOperationUnsupported	0x80720000	服务器不支持请求的步骤。
Bad_IndexRangeInvalid	0x80360000	索引的区域参数的语法无效。
Bad_IndexRangeNoData	0x80370000	指定索引的范围内无数据。
Bad_DataEncodingInvalid	0x80380000	数据编码无效。
Bad_DataEncodingUnsupported	0x80390000	服务器不支持节点请求的数据编码。
Bad_NotReadable	0x803A0000	安全等级不允许读取节点或登录节点。
Bad_NotWritable	0x803B0000	服务器不允许写入节点。
Bad_OutOfRange	0x803C0000	值超出允许的范围。
Bad_NotSupported	0x803D0000	不支持请求的活动。
Bad_NotFound	0x803E0000	未找到请求的元素或在没有成功的情况下终止了浏览。
Bad_ObjectDeleted	0x803F0000	对象由于已删除而无法使用。
Bad_NotImplemented	0x80400000	未实施请求的活动。
Bad_MonitoringModelError	0x80410000	监视模式无效。
Bad_MonitoredItemIdInvalid	0x80420000	监视数据项的 ID 未引用一个有效的监视数据项。
Bad_MonitoredItemFilterInvalid	0x80430000	监视数据项的过滤条件参数无效。
Bad_MonitoredItemFilterUnsupported	0x80440000	服务器不支持监视数据项的过滤条件。
Bad_StructureMissing	0x80460000	所需的结构化参数丢失或等于零。
Bad_EventFilterInvalid	0x80470000	事件过滤条件无效。
Bad_ContentFilterInvalid	0x80480000	内容过滤条件无效。
Bad_FilterOperandInvalid	0x80490000	内容过滤条件中使用的操作数无效。
Bad_ContinuationPointInvalid	0x804A0000	连续点的返回值仍然无效。
Bad_NoContinuationPoints	0x804B0000	由于删除了所有连续点，所以未执行请求的作业。
Bad_ReferenceTypeIdInvalid	0x804C0000	该引用类型未引用一个有效的引用类型节点。

符号 ID	错误代码（十六进制）	含义
Bad_BrowseDirectionInvalid	0x804D0000	浏览方向不再有效。
Bad_NodeNotInView	0x804E0000	该节点不属于视图。

#### 4.4.8 从 OPC 数据访问/报警和事件移植到 OPC UA

哪些服务在 OPC 数据访问/报警和事件与 OPC 统一构架之间有所不同？

如果想要将现有的 OPC DA / AE 应用程序移植到 OPC UA，在此，可以找到两个系统中相应服务的概述。

OPC 数据访问/OPC 报警和事件	OPC 统一架构
CoCreateInstanceEx()	OpenSecureChannel() CreateSession() ActivateSession()
ChangeBrowsePosition() BrowseOPCItemIDs() GetItemID() QueryAvailableProperties() GetItemProperties()	Browse() Read()
通过项名称访问： Read() (DA 3.0) Write() (DA 3.0)	通过节点 ID 访问： Read() Write()
使用句柄访问组的数据项： AddItems() Read(...,handle,...) Write (... ,handle,...) RemoveItems()	使用句柄访问已注册节点： RegisterNodes() Read(...,handle,...) Write (... ,handle,...) UnregisterNodes()
AddGroup() SetState() RemoveGroup()	CreateSubscription() ModifySubscription() DeleteSubscription()

OPC 数据访问/OPC 报警和事件	OPC 统一架构
AddItems() RemoveItems()	CreateMonitoredItems() DeleteMonitoredItems()
DataChange()	Publish()

SIMATIC NET OPC 服务器将继续支持先前规范（例如“OPC 数据访问”或“OPC 报警和事件”）的语法。这样就简化了到 OPC UA 的移植。

## 4.5 对 OPC UA 冗余接口进行编程

### 透明的冗余 OPC UA 服务器

冗余 OPC UA 服务器用于在正常运行期间分配负载。如果其中一个 OPC UA 服务器出现故障，则由冗余 OPC UA 服务器接管其通信功能。

透明表示从 OPC UA 客户端的角度来看，簇中组态的 OPC UA 服务器显示为单个 OPC UA 服务器，这种服务器可防止出现故障。OPC UA 客户端只需使用一个簇 IP 地址就可对簇的 OPC UA 服务器进行寻址。OPC UA 服务器之间会进行数据同步，确保能够自动处理两个 OPC UA 服务器之间的故障切换，并且在很大程度上，OPC UA 客户端方面无需执行任何操作。

### 透明冗余 OPC UA 服务器的工作方式

借助 OPC UA 冗余规范，可以在标准自动化技术中使用透明的服务器冗余。标准 UA 客户端无需使用其它专用冗余软件。

使用透明冗余服务器时，假定没有服务器发生故障，OPC UA 客户端的任务将在 OPC UA 服务器上平均分配。这样可确保负载平衡。两个 OPC UA 服务器（见下图）客户端上统一的任务分配有助于更好地利用资源。

OPC UA 客户端会话和订阅在 OPC UA 服务器之间定期进行同步。服务器通过工业以太网以 10/100/1000 Mbps 的速率同步。这意味着同步过程不需要额外的电缆。如果客户端与 OPC UA 服务器之间的连接中断，则第 2 个 OPC UA 服务器将接管其任务。会话和订阅只是临时中断，这样可以快速恢复作业。

可以使用 STEP 7 对 OPC UA 冗余服务器进行集中组态。这样，只需使用几个组态规范就可创建一个组态。必要时会创建并分配证书。

服务器之间的 **S7** 协议连接不进行冗余组态。换言之，如果一个服务器发生故障，则第 2 个服务器会再次读取数据。

下图是使用两个 **OPC UA** 服务器时负载平衡的图形表示。

## 4.5 对 OPC UA 冗余接口进行编程

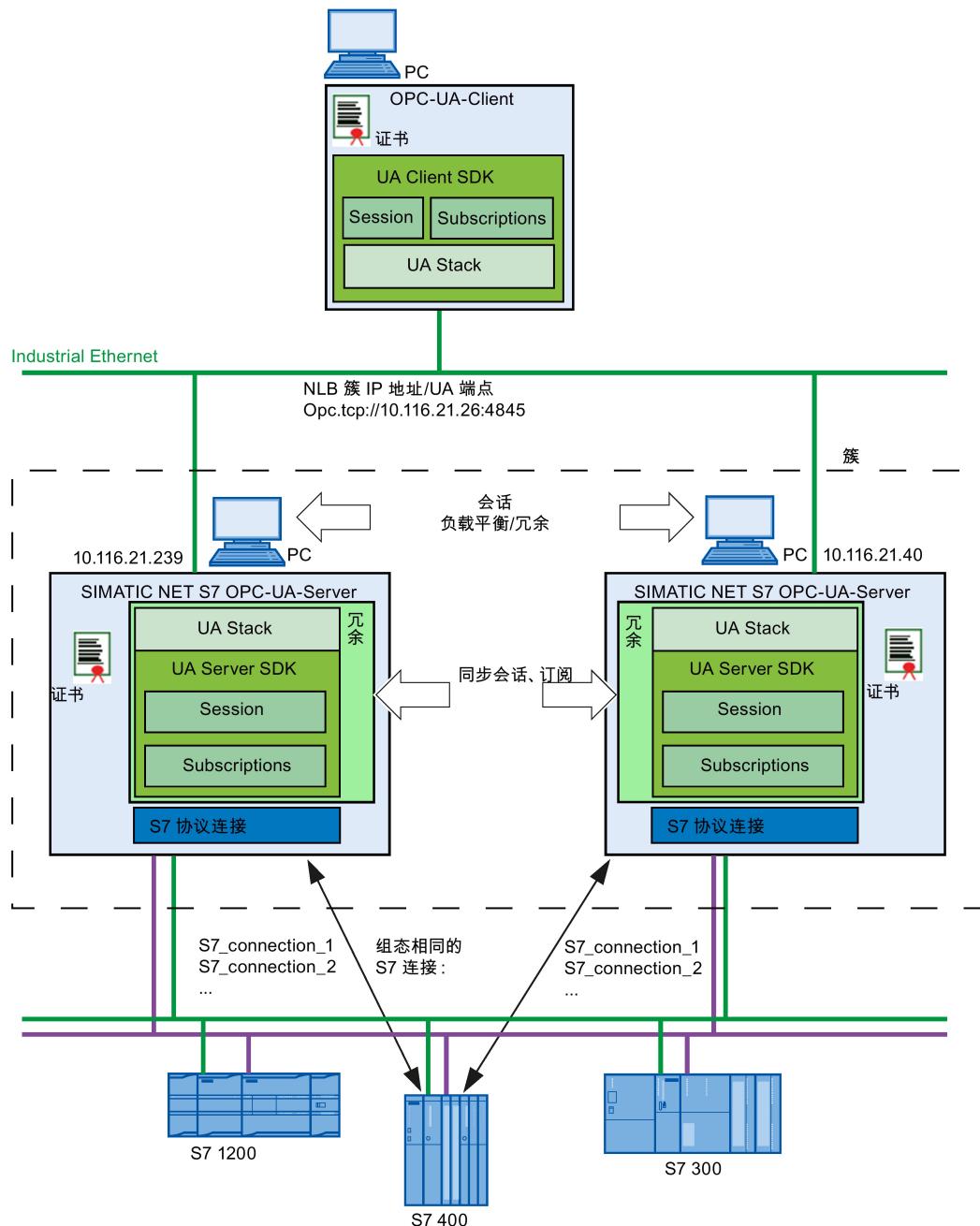


图 4-16 冗余网络的示意图结构

透明的 OPC UA 冗余需要 Windows 功能“网络负载平衡”(NLB)。由于只有 Windows 服务器操作系统具有 NLB 功能，所以最低要求是 Windows 服务器操作系统。从 SIMATIC NET PC 软件 V8.1 开始，需要使用“Windows Server 2008 R2 64 位”。

### “网络负载平衡”(NLB) 功能

Windows“网络负载平衡”(NLB) 功能可以保证负载分配和多个服务器 PC 的冗余。

利用 NLB，可以在网络中的一个网络地址下注册多个 OPC UA 服务器 PC。为此，NLB 为一个客户端提供一个簇 IP 地址，在该地址下最多可组合 32 个物理服务器 PC。建立一个新的 TCP/IP 连接时，连接会在可用的服务器上平均分配。这样可确保负载分配和高可用性。不需要冗余的客户端特殊组态。

#### 如果存在连接的 OPC UA

服务器发生故障，则客户端会根据组态的连接监视时间检测中断连接。在 OPC UA 规范定义的重新连接方案中，通过 NLB 可以自动建立到冗余服务器的新 TCP/IP 连接。

仅在消息返回至 OPC UA 客户端后，才会执行冗余组中 OPC UA

服务器间的会话和订阅信息同步，这也需要一定的时间（参见上图）。

同步通过工业以太网使用带有单独端口的 OPC UA 连接进行。这就意味着在 TCP/IP 重新连接后，客户端无需创建新的订阅就能立即再次接收数据。在故障切换期间，HMI 操作员将接收不到任何数据更新。在大多数 OPC UA

客户端上，连接错误检测与重新连接之间的数据质量也会暂时设定为“BAD”。对于 OPC UA 客户端，OPC UA 服务器故障看起来只是 TCP/IP 连接中止。如果 OPC UA 服务器之间的连接中断，在 OPC UA 客户端与冗余 OPC UA

服务器之间的连接故障切换期间，可能无法获得有关新建会话和订阅的所有信息。

### 条件

为能够使用此功能，必须对所有服务器进行相同的组态。

有关相同组态的详细信息，请参见“簇组态 (页 618)”部分中的“确保 OPC UA 服务器的一致性”。

### 使用“网络负载平衡”(NLB) 的限制和相关注意事项

负载平衡期间，NLB 端口规则决定了服务器的响应

默认情况下，协议设置为“兼有”(Both)，过滤模式设置为“多个主机”(Multiple host)，相关设置为“单个”(Single)。如果更改这些设置，则会更改 OPC S7 冗余服务器对负载平衡的响应。

例如，如果将协议组态为“TCP”，同时将过滤模式组态为“多个主机”(Multiple host)，相关设置为“无”(None)。同时，应检查对其它使用 NLB 的服务器应用程序所产生的影响。

#### 4.5.1 安装和组态“网络负载平衡”功能

要允许冗余服务器操作，从 SIMATIC NET PC 软件 V8.1 开始，需要使用支持“网络负载平衡”(NLB) 功能的“Windows Server 2008 R2 64 位”操作系统。

##### 安装网络负载平衡

要安装此功能，请按以下步骤操作：

1. 启动安装。

单击“开始 > 控制面板 > 管理工具 > 服务器管理器 > 功能”(Start > Control Panel > Administrative Tools > Server Manager > Features)，“添加功能”(Add Features)。

响应：将打开“添加功能向导”(Add Features Wizard) 对话框。

2. 选择“网络负载平衡”(Network Load Balancing) 功能并单击“安装”(Install)。
3. 按照安装说明操作。

##### 组态网络负载平衡

要组态“网络负载平衡”(NLB) 功能，请启动“网络负载平衡管理器”(Network Load Balancing Manager)。请按照下面列出的步骤进行操作：

1. 启动“网络负载平衡管理器”(Network Load Balancing Manager)。

单击“开始 > 控制面板 > 管理工具 > 服务器管理器 > 网络负载平衡管理器”(Start > Control Panel > Administrative Tools > Server Manager > Network Load Balancing Manager)。

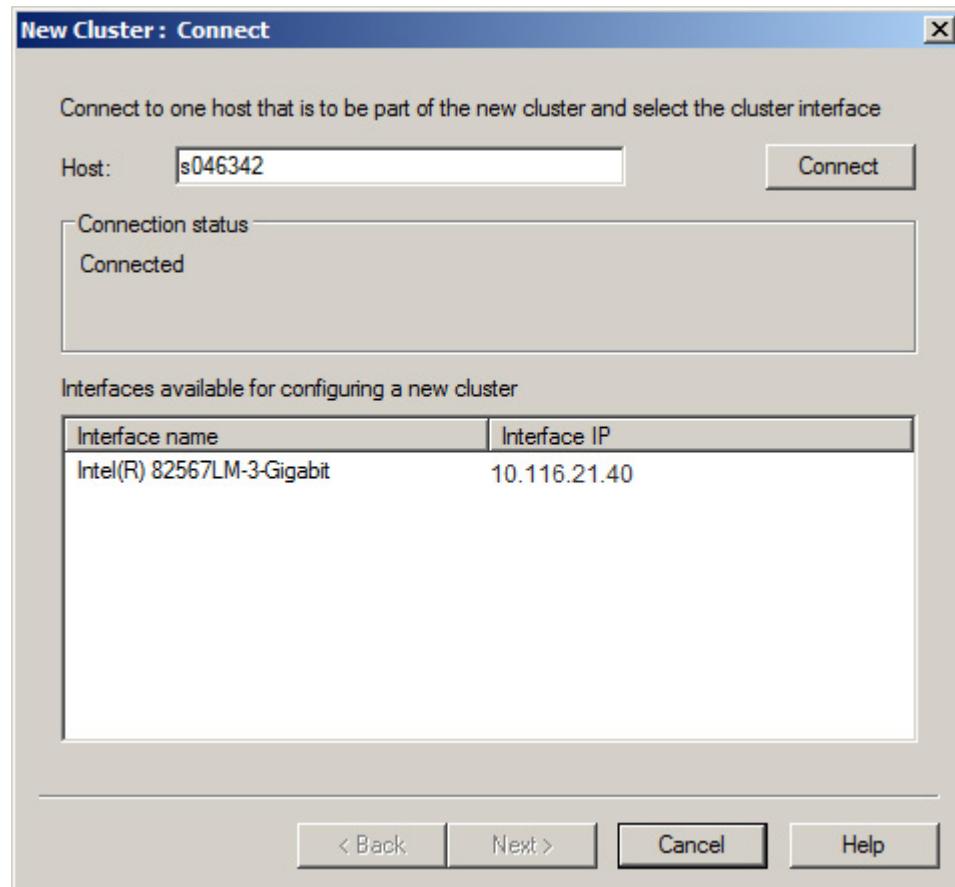
2. 创建新簇。

在菜单中单击“新建 > 簇”(New > Cluster)

响应：将打开“创建新簇：连接”(New Cluster: Connect) 对话框。

3. 将簇首先连接至本地主机和本地以太网接口（见下图）。

在“主机：”(Host: ) 输入框中，输入主机的名称并单击“连接”(Connect)。

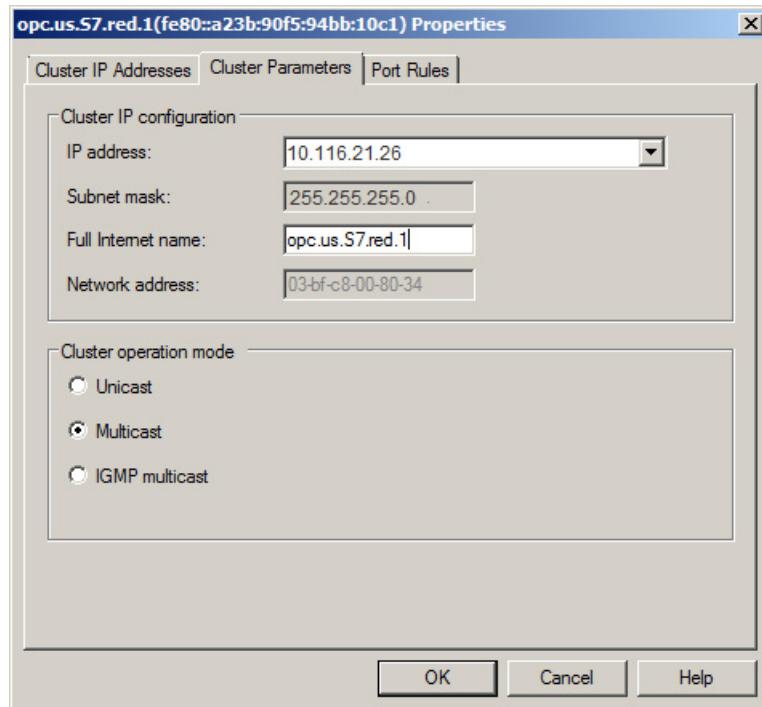


#### 4.5 对 OPC UA 冗余接口进行编程

4. 打开“属性”(Properties) 对话框。

右键单击接口名称并从快捷菜单中选择“属性”(Properties)。

响应： 将打开“属性...”(Properties of...) 对话框。



5. 然后，组态簇的 IP 地址和 Internet 名称（见下图）。

建议使用簇操作模式“组播”(Multicast) 或“IGMP 组播”(IGMP multicast)，以允许多次使用网络适配器。

---

#### 说明

##### 选择簇操作模式

- 利用簇操作模式的参数，可以指定簇操作是否使用组播 MAC 地址（介质访问控制）。如果激活组播，簇适配器的簇 MAC 地址将通过网络负载平衡转换为组播地址。这样还可确保在此组播地址中作为 ARP（地址解析协议）的一部分解析簇的主 IP 地址。  
现在可以使用在单播模式下取消激活的适配器原始集成 MAC 地址。
  - 在激活 Internet 组管理协议 (IGMP) 的支持之前，需要激活组播支持。  
也可以在网络适配器上激活 IGMP 支持。
  - IGMP 组播复选框可激活用于限制切换流的 IGMP 支持。  
通过限制到“网络负载平衡端口”的数据流量可实现这一限制。激活 IGMP 支持，以确保用于 NLB 簇的数据通信只通过用于簇主机的端口进行。
- 

6. 单击“确定”(OK) 进行确认。

---

#### 将其它主机添加到簇

---

#### 说明

如果想要将其它主机（服务器）添加到簇，则需要远程主机的管理员权限。

---

## 4.5 对 OPC UA 冗余接口进行编程

要获取远程主机的管理员权限，需要使用管理员权限登录“网络负载平衡管理器”(Network Load Balancing Manager)。请按照下面列出的步骤进行操作：

1. 打开“网络负载平衡管理器”(Network Load Balancing Manager)

中的“凭据”(Credentials)。

单击“选项 > 凭据”(Options > Credentials)。

响应： 将打开对话框窗口“NLB 管理器默认凭据”(NLB Manager Default Credentials)。

2. 输入用户名和相应密码。

服务器处于工作组模式时，请使用默认 Windows 用户“管理员”(Administrator)（内置）及相应密码。



3. 单击“确定”(OK) 进行确认。

### 说明

由于用户权限由 Windows 域进行管理，因此服务器的域模式将提供更高的安全性。推荐使用该模式。

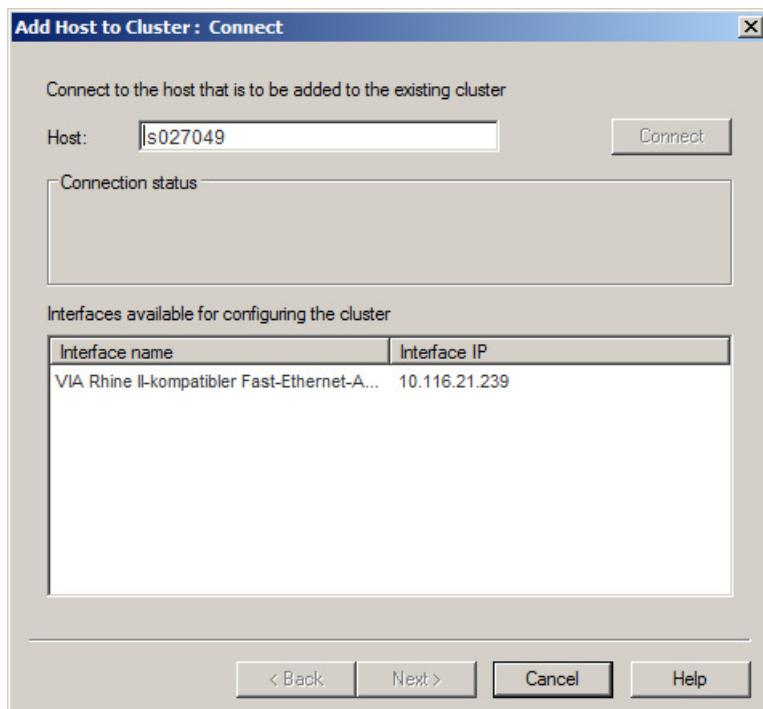
要向簇中添加其它远程主机，请按以下步骤操作：

1. 单击“簇 > 添加主机”(Cluster > Add Host) 菜单命令。

响应： 将打开“添加主机至簇：连接”(Add Host to Cluster: Connect) 对话框。

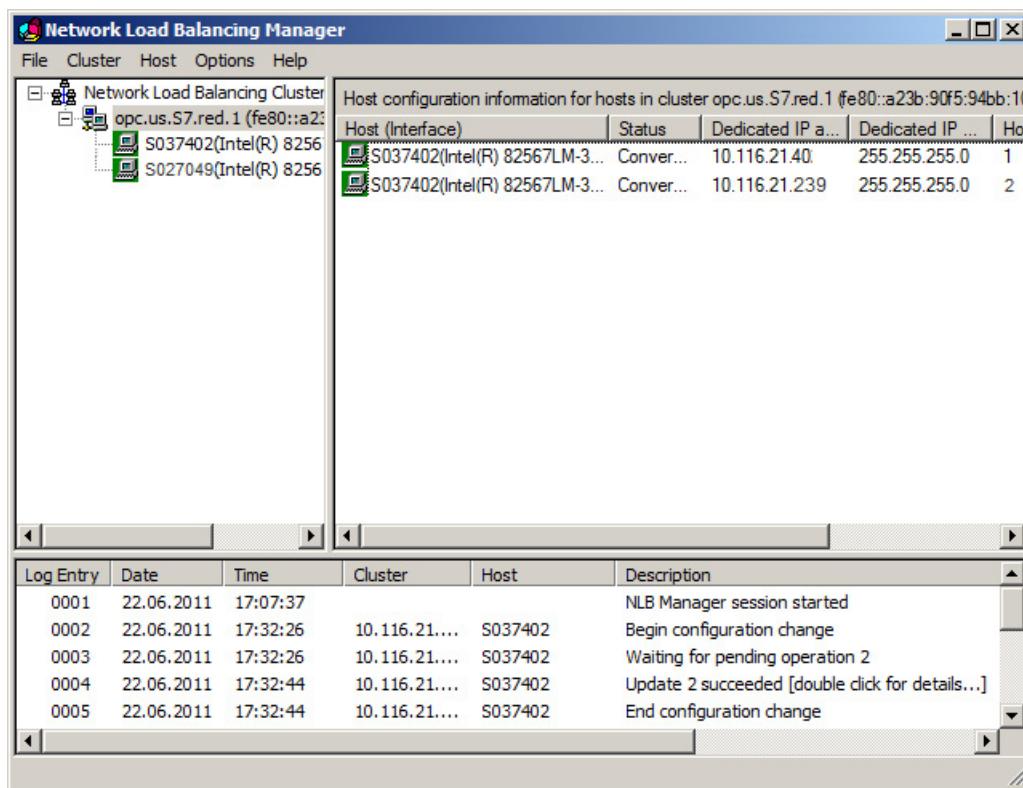
2. 在“主机：”(Host: )

输入框中，输入想要添加到簇中的主机的名称并单击“连接”(Connect)。



成功建立主机连接后，将创建一个新簇。

## 4.5 对 OPC UA 冗余接口进行编程



### 说明

透明的 OPC UA 冗余需要激活并且适当组态的“网络负载平衡”(NLB)。

可以在操作系统中使用 PING 测试 NLB 是否激活。

## 4.5.2 簇组态

簇的初始组态在 STEP 7 中创建。此外，使用 STEP 7 还可组态会话和订阅的同步。

使用组态接口可执行下列各项操作：

- 组态簇的虚拟网络地址（例如，冗余 IP 地址和冗余端口）。
- 组态用于冗余 OPC UA 服务器之间服务器到服务器通信的端口。
- 指定其它簇成员的 IP 地址。
- 从列表中选择本地服务器。

必要时，可以在组态中创建并显示冗余集的证书。

## 说明

### 组态接口

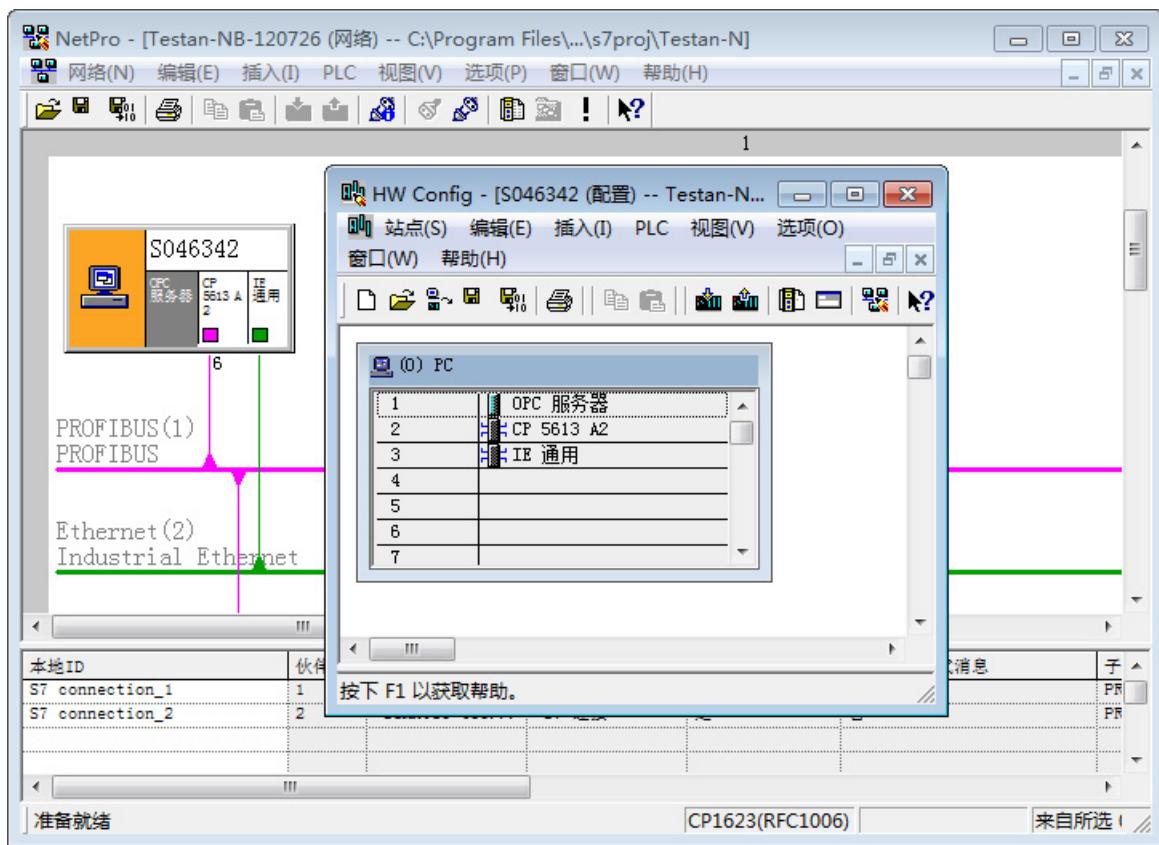
- 簇的所有服务器上的列表和序列必须相同。
- 只能使用 IPv4 地址。
- 不需要组态“网关”(gateway) 或“子网掩码”(subnet mask)。
- 无需组态 MAC 地址。

无法使用 STEP 7/NCM 组态“网络负载平衡”(NLB)。同时，也无法组态 NLB 的端口阻止列表。NLB 的组态需要集成到冗余服务器的操作系统中，并且必须在此处执行。

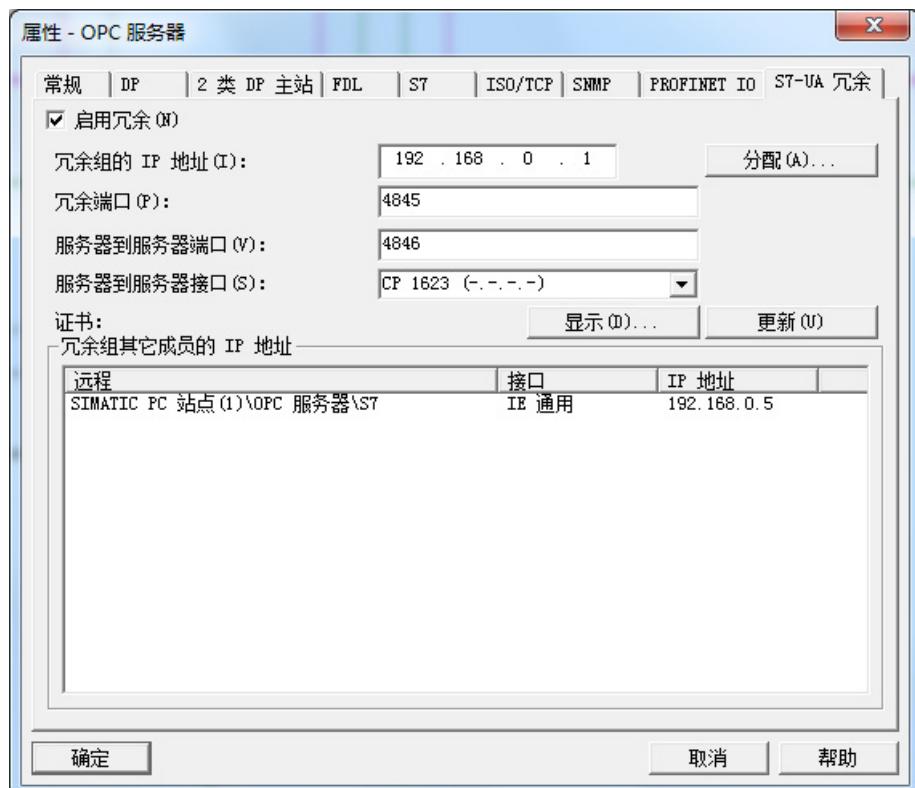
## 组态示例

以下示例说明了一个带工厂总线（例如用于 OPC 客户端/服务器通信的“PROFIBUS(1)”和“工业以太网(2)”）的 STEP 7 项目。在示例“CP 5613 A2”和“IE General”中，每一个冗余服务器都有两个 CP。作为目标站，可以选择 S7 站。应当通过工厂总线和适当 CP 将它们连接起来。

## 4.5 对 OPC UA 冗余接口进行编程



下图显示了带有必需组态选项的示例。该组态集成到 STEP 7 中。可以在 OPC 服务器（版本 8.1 及以上版本）的“属性”(Properties) 窗口中设置 S7 UA 冗余的参数。



## 确保 OPC UA 服务器的一致性

在一致性检查中，例如保存和编译项目时，需要检查以下实际信息：

- 簇中的服务器到服务器地址必须是唯一的。簇的公共 IP 地址也只能在项目中使用一次。
- 簇的每个成员必须使用相同的 S7 连接，连接的名称和通信伙伴都应相同。
- 簇中各 PC 站的硬件组态必须相同。

- S7 连接的参数必须相同。其中包括：连接建立、报警、优化、PDU 大小、并行作业数、监视时间。
- OPC UA 服务器的 S7 设置必须相同。其中包括：循环时间、访问权限和 S7 符号。

#### 说明

但是，可以组态所有服务器，使其也能访问同一网络中使用单个 IP 地址的其它簇。

#### 说明

在 S7 协议等级，也可使用 S7 Redconnect 产品。不过，这并非冗余 OPC 所必需。

#### 说明

强烈建议您在 OPC UA 服务器中组态永久建立的主动 S7 连接，以实现以下几点：

- 服务器出现故障时缩短故障切换的时间。
- 能够检查被动冗余服务器的连接状态（S7 连接诊断）。

要使两个服务器具有相同的命名空间，必须对两个 PC 站进行相同的组态。

为此，建议首先创建一个 SIMATIC PC 站并在 NetPro 中对其进行组态（例如创建要求的 S7 连接）。

在 SIMATIC 管理器中，现在可以复制创建的 SIMATIC PC 站（图“S046342”）。

使用“复制”(Copy) 和“粘贴”(Paste) 菜单命令，可以复制 PC 站，其中包括具有相同名称的 S7 连接。现在，调整站名称（图“S027049”）、S7 连接的本地地址和目标。

如果添加其它 S7 连接，则必须在两个站中对其进行相同组态。

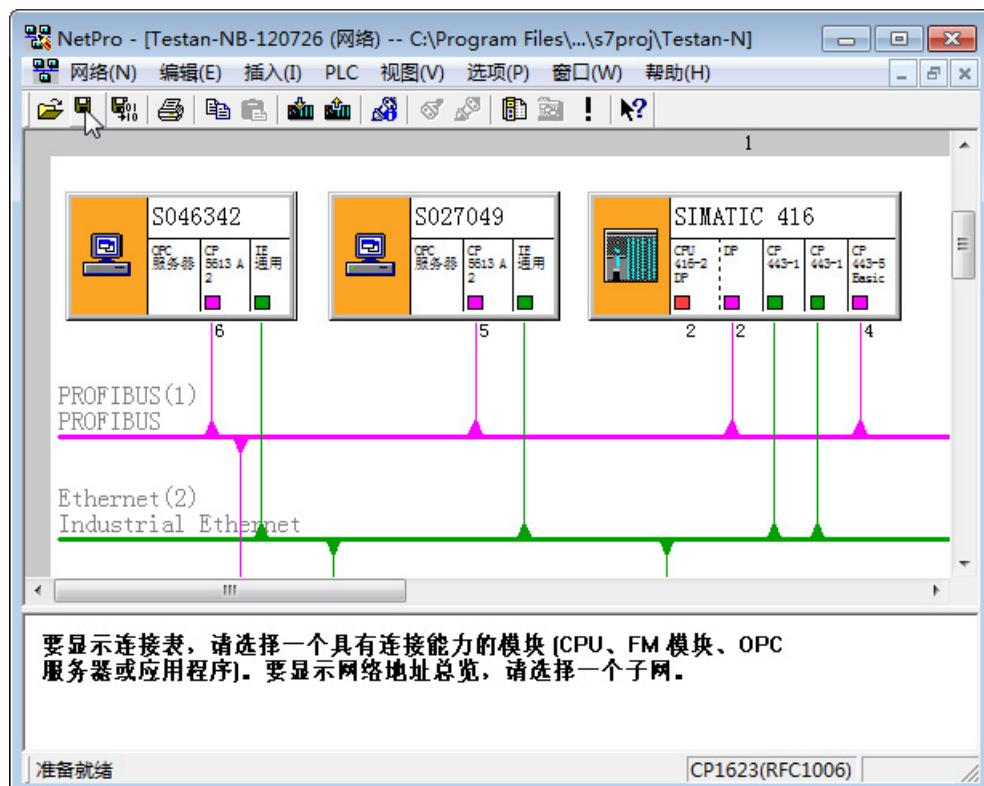


图 4-17 使用 S7 连接在 NetPro 中对 PC 站进行相同组态

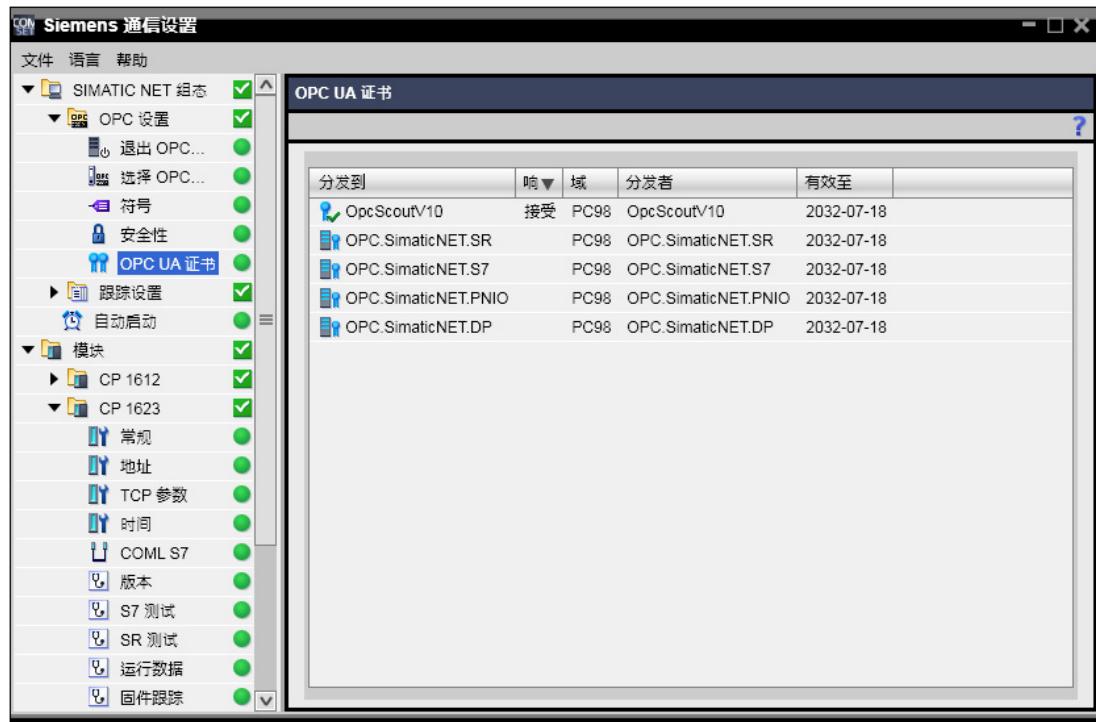
在冗余组态中，也需要将此站分配给第一个站的簇。这是利用同一 SIMATIC S7 站的相同 S7 连接创建冗余服务器集的最简单方式。

请记住，组态期间需要将 SIMATIC S7 站的 S7 连接数加倍。

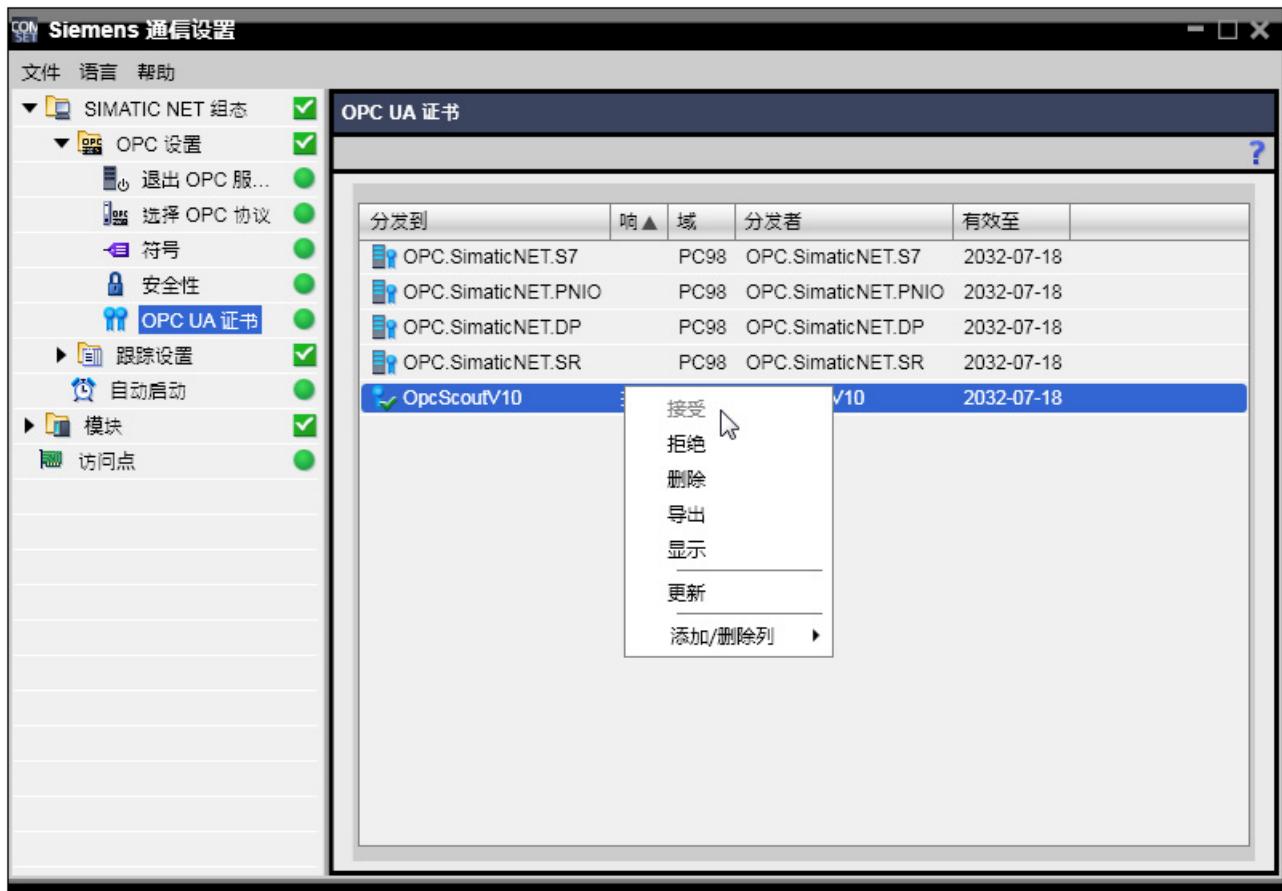
## 4.5 对 OPC UA 冗余接口进行编程

### 下载组态和更新 UA 证书

完成组态并成功检查一致性之后，需要下载所有节点的组态。还要在两个 SIMATIC PC 站上下载并激活相同的 OPC UA 证书。使用组态程序“通信设置”（“应用程序 > OPC 设置 > OPC UA 证书”(Applications > OPC Settings > OPC UA Certificates)）对其进行检查。



在证书管理中，还必须接受 OPC UA 客户端的证书。为此，使用冗余 IP 地址（例如“10.116.21.26”）将 OPC UA 客户端与冗余服务器连接起来，并与“组态网络负载平衡”部分中的图进行比较。



当 NLB 激活时，将连接两个现有服务器之一，接收安全连接上的客户端证书。

检查组态的 OPC UA 命名空间和数据项的可用性。

要同时接收第二个服务器上的客户端证书，请从第一个服务器中拔出网络电缆。

然后，使用冗余 IP 地址再次连接到 OPC UA 客户端。现在，NLB

将连接到第二个服务器。输入 PC 的计算机名称或本地 IP 地址。

第二个冗余服务器的动作应当与第一个的完全相同并提供相同值。

如果未建立连接，请检查系统结构和组态。

## 4.5 对 OPC UA 冗余接口进行编程

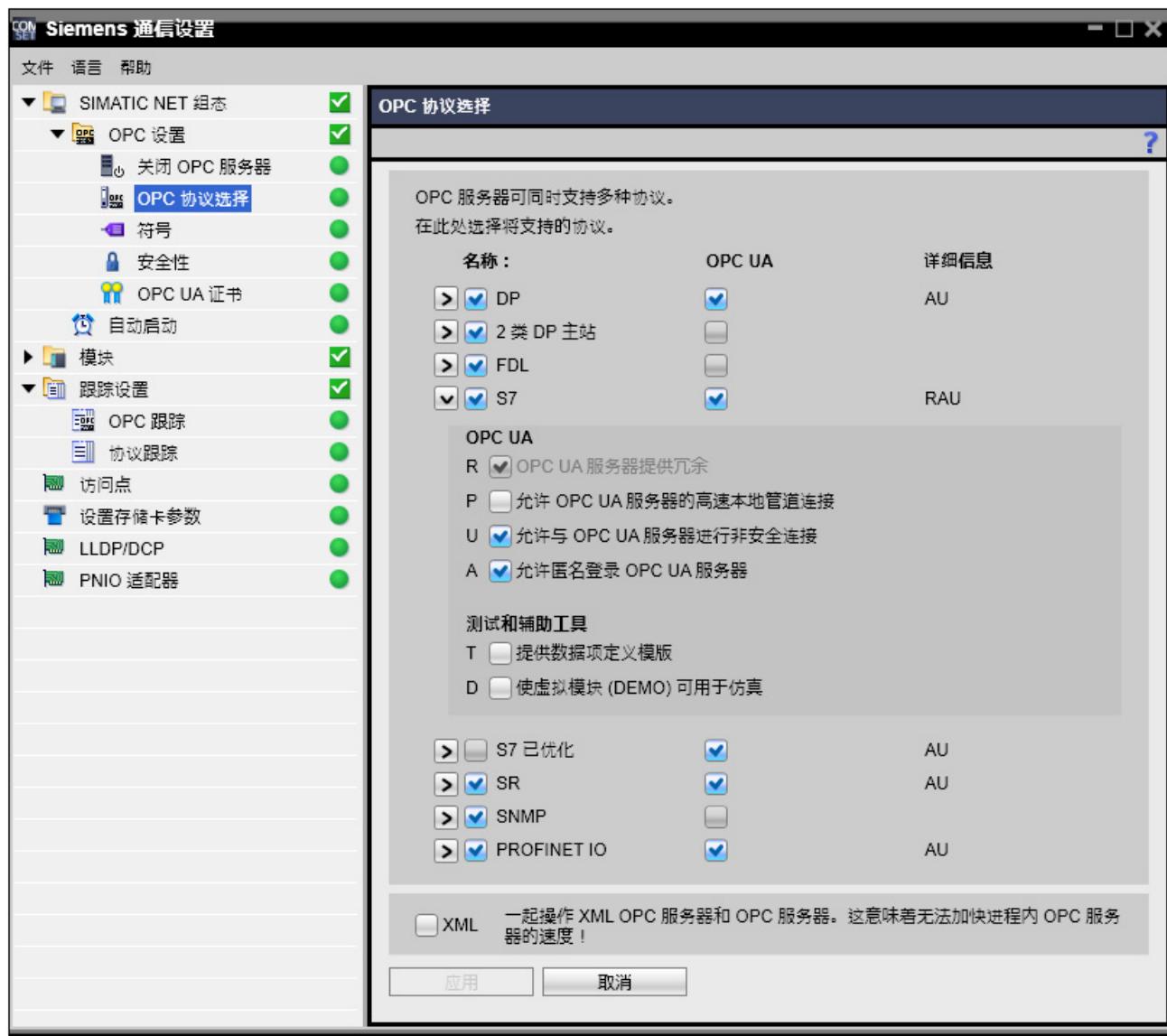


图 4-18 OPC UA 服务器冗余运行时协议详细信息的视图

## 测试冗余故障切换

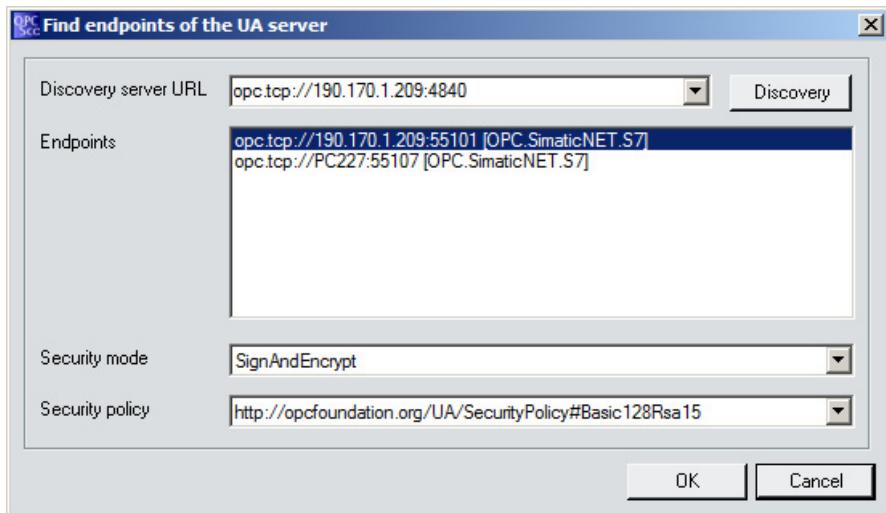
可以通过拔出活动服务器的网络电缆来测试冗余故障切换。

当客户端上检测到网络中断时，必须重新连接到现有的其它服务器。

可使用诊断项监视故障切换响应。有关此主题的详细信息，请参见“由 OPC UA 客户端读出簇的状态 (页 636)”部分。

### 4.5.3 使用发现服务器定位服务器端点

使用 OPC UA 发现服务，可以定位冗余 S7 OPC UA 服务器的端点。通过输入公共簇 IP 地址和端口（“4840”），使用 OPC UA TCP 协议可以找到端点。



由于通过 NLB 的 TCP 连接透明且固有负载平衡，因此无法预测将连接哪些冗余服务器。此列表框显示了服务器在相关安全模式下的所有端点。

对于冗余、透明的 OPC UA 连接，请选择具有已组态公共 IP 地址和已组态服务器端口的端点。示例为“`opc.tcp://190.170.1.209:55101`”。

对于一个有意非冗余 OPC UA 连接，也可以选择特定 S7 OPC UA 服务器的端点。示例为“`opc.tcp://PC227:55107`”。

此连接可用于与特定服务器交换证书或检查各个服务器的数据质量。

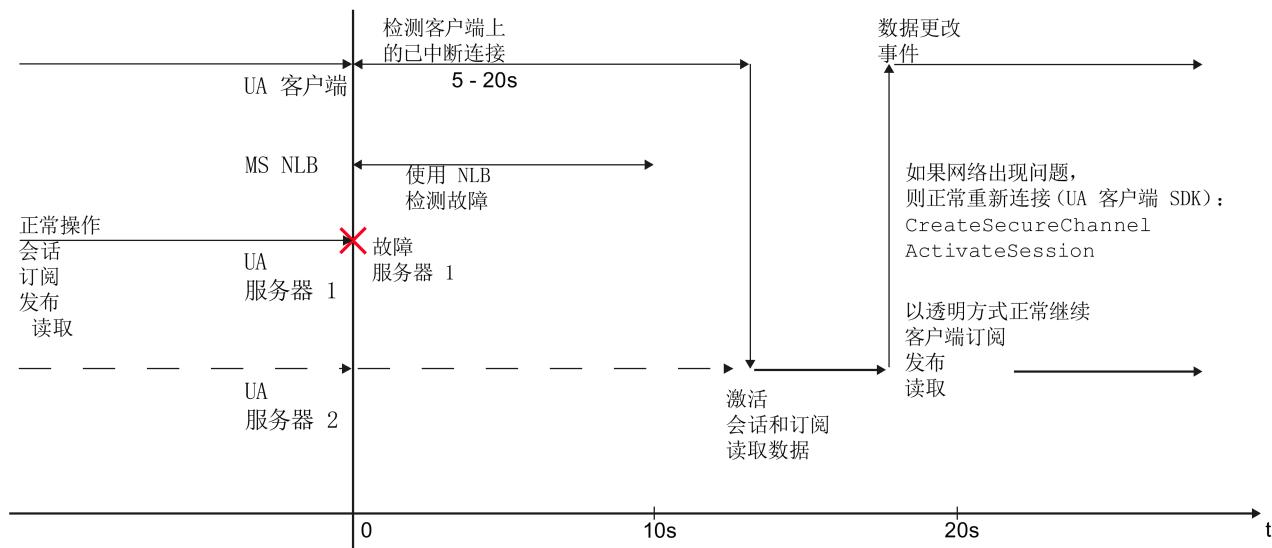
请记住，用于其它冗余服务器的 OPC UA 会话和订阅将实现同步，但不是针对此端点。

### 4.5.4 OPC UA 重新连接步骤

如果冗余 OPC UA 服务器发生故障，必须由服务器操作系统的 Windows“网络负载平衡”功能和 OPC UA 客户端检测该故障。根据网络错误的发生或超时可以检测连接中断。

“网络负载平衡”(NLB) 需要大约 10 秒来检测 OPC UA 服务器 1 的故障。如果存在故障，NLB 将把客户端的新调用导向可用的冗余 OPC UA 服务器 2。

## 4.5 对 OPC UA 冗余接口进行编程



## OPC UA

客户端上的检测时间取决于连接监视的类型和设置的监视周期，因为它们能够对检测时间产生直接影响。OPC UA 客户端可通过两种方式监视连接：

- 使用较短超时循环读取服务器状态。

检测时间取决于以下因素：

- 读取作业的循环时间（例如 5 秒）。
- 读取作业的超时（例如 2 秒）。
- 发生错误时，读取作业在连接显示为中断之前的重复次数（例如 1x）。

- 监视订阅的保持连接值。

检测时间取决于以下因素：

- 保持连接时间（例如 5 秒）。
- 出现错误消息之前的额外等待时间（例如 5 秒）。

检测到连接错误（网络错误、超时）之后，将开始客户端正常的 OPC UA 重新连接步骤。此重新连接功能通常集成在客户端 UA SDK 中，并且对 OPC UA 客户端透明。在重新连接步骤中，将出现以下情况：

- 使用“`CreateSecureChannel`”创建新的“`SecureChannel`”。

然后，TCP/IP 会与 NLB 的冗余 IP 地址连接。NLB 将平均分配 TCP 连接。由于 OPC UA 服务器 1 发生故障且无法访问，将建立到 OPC UA 服务器 2 的 TCP 连接。

- 现有会话将由 `ActivateSession` 通过分配 `SecureChannel` 激活。

OPC UA 客户端未识别其已连接到不同的服务器，因为会话和订阅 ID 仍保持有效。因此，需要再次激活会话和现有订阅。

现在，OPC UA 客户端能够以透明方式继续其正常的功能。

例如，使用再次可用的订阅进行监视，或者进行简单的读取和写入操作。

OPC UA 客户端将向服务器发送一个 **PublishRequests**，以获取新的数据和事件。OPC UA 服务器 2 了解会话和所有订阅，并且会激活它的监视数据项。数据使用 SIMATIC NET 通信协议读取。如果创建多个事件监视数据项，还会触发 **RefreshRequiredEvent**。只要存在数据或事件，就会立即使用 **PublishResponse** 将其发送至客户端。

## 4.5.5 OPC UA 客户端支持的冗余

### 简介

OPC UA 客户端要求的“优化冗余故障切换”属性遵从 OPC UA 标准方面的特性进行评估。

基本特性如下所示：

- 连接中止或连接监视检测
- 重新连接功能
- 刷新必需事件

### 连接中止检测

要检测连接中止，OPC UA 客户端必须具备连接监视的能力。

例如，可通过下列方式监视连接：

- 监视“Read”方法的状态
- 监视订阅的保持连接值

一旦检测到连接中止，OPC UA 就应执行重新连接逻辑。

### 重新连接功能

对于重新连接，OPC UA 将使用继续使用现有会话和订阅的 OPC UA 选项。

为此，必须按照下列顺序操作。该顺序与初始化连接步骤不同。

1. CreateSecureChannel

2. ActivateSession

如果 AcitvateSession 中存在错误

- CreateSession
- ActivateSession
- 创建一个新订阅

3. 等待数据更改/事件或检查订阅是否仍然存在。

如果存在订阅错误

- 创建一个新订阅

如果 OPC UA

客户端按照和连接步骤相同的方式执行重新连接步骤，唯一的好处就是客户端可以再次连接到相同的服务器端点。

### RefreshRequiredEvent

OPC UA 客户端必须对服务器的 RefreshRequiredEvent 做出响应。为此，适当地刷新 OPC UA 客户端。此顺序将在“OPC UA 第 9 部分 - 报警和条件”规范中进行介绍。

#### 说明

检查正在使用的 OPC UA 客户端，确保其行为遵从标准。通常，基于 SDK 的客户端已经包括了此类机制。

### 4.5.6 什么时候会发生服务器故障切换？

如果服务器发生故障，OPC UA

客户端检测到通道、会话或订阅的故障时会发生故障切换。例如：

- 主机 PC 发生故障时
- OPC UA 服务器崩溃时
- 操作系统崩溃时

下列情况下不会发生故障切换：

- S7 连接中断或出现的其它错误状态阻止 OPC UA 服务器从控制器接收 S7 数据。
- OPC UA 服务器上出现的错误状态未引起通道、会话或订阅的故障，或者 OPC UA 客户端未检测到该故障。

#### 4.5.7 服务器故障切换的详细信息

##### OPC UA 客户端可能的错误消息

如果服务器 1 发生故障时写入或读取作业处于活动状态，OPC UA 客户端的堆栈将发送信号指示这些作业的错误。

然后，客户端应用程序必须重复这些作业。

##### OPC UA 客户端用户可能的错误消息

正常情况下，如果与服务器的连接中止，OPC UA 客户端会在内部将变量的质量设置为“BAD”。重新连接之后，新数据将再次复位质量。

如果出现连接中止，数据质量（换言之“BAD”）会显示给用户。

该信息只会显示几秒或者根本不会出现，具体取决于客户端。

出现这种情况的原因是，只有在检测到连接中止且随后立即自动开始重新连接的情况下，数据质量才会设置为“BAD”。

##### 数据和事件不同步

使用 S7 通信协议实现客户端和所连 PLC 之间的数据和事件交换，该过程在冗余 OPC UA 服务器之间不同步。

冗余集中的每个服务器都只能通过其自身的 S7 连接处理此信息。

冗余故障切换之后，新服务器将从 PLC 再次接收该信息。

如果故障切换之后激活订阅，将发生以下情况：

- 对于数据类型“数据更改监视项”，当前值可使用 SIMATIC NET 通信协议获取，并在读取后返回。
- 对于数据类型“事件监视项”，将向客户端发送 RefreshRequiredEvent，以便由客户端触发刷新。  
这会将激活报警的当前状态发送至客户端，该客户端具有一个新服务器已知的新“EventId”。UA 报警客户端可以依据相同的“ConditionId”将事件分配给已知条件。

不过，为了能够执行操作（例如确认报警），客户端需要一个服务器已知的“EventId”。

如果只有短暂中断并且重新连接到相同服务器，则不会向客户端发送“RefreshRequiredEvent”。

## 服务器故障切换限制

服务器上的以下本地数据和状态不能实现同步：

- @ 由客户端或使用 PUT 和 GET 函数块通过远程访问（S7 站到 PC 站、S7 连接）写入或读取的本地服务器数据。
- 本地 DEMO 变量。
- 块服务“BSEND/BRCV”的本地缓冲区（包括它们的段），因为块服务在 PLC 和其中一个冗余 OPC 服务器之间进行显式交换。
- 来源于不同组态的本地特殊组态连接参数（波特率、PDU 大小、超时间）。

不同的组态能够产生如下影响：

- 不同的项目或工作状态下载到两个服务器中。
- 运行期间，不同的参数在通信伙伴之间进行协商。
- 来源于不同组态的本地特殊组态符号（S7 连接的 S7 符号名称、死区...）。
- 本地报警状态、历史和确认。例如，这适用于本地生成的 Statepath 报警。
- 本地接收或拒绝冗余集中本地 OPC UA 服务器的 OPC UA 客户端证书。

使用“通信设置”中的证书管理来接受涉及到的所有 OPC UA 服务器的 OPC UA 客户端证书。这样可确保出现故障切换时，服务器能够接受这些证书。自“SIMATIC NET PC 软件 V8.2”起，冗余服务器的 OPC UA 客户端证书会自动同步。

- 子组件的本地故障和 SIMATIC NET 通信协议的本地错误状态。

该数据必须由客户端管理，在检测到连接中断时必须对其进行重新初始化或请求。只有在簇的组态期间对服务器命名空间进行完全相同的组态时，此操作才会生效。

## 关于 S7 报警的注意事项

- 如果存在连接中断，不会在服务器与客户端之间转发诊断报警（简单事件）。这也适用于发生故障切换的情况。相应事件将丢失。
- S7 报警。如果 PLC 上发生 S7 报警更改，此更改会分别发送至两个 OPC UA 服务器。S7 报警可以在不同时间到达 OPC UA 服务器。

例如，时差可以由多个报警或组态过大引起，因此可以遵照故障切换时间的顺序。  
如果是这种情况，则会出现：

- 服务器发生故障之前，OPC UA 客户端上的报警状态将从未激活切换至激活。  
冗余故障切换之后，第二个 OPC UA 服务器将触发刷新。  
报警状态先后指示为“未激活”和“激活”。这也适用于相反的情形。
- 服务器发生故障之前，OPC UA 客户端会确认报警。冗余故障切换之后，第二个 OPC UA 服务器将触发刷新。报警将先后指示为“未确认”状态和“已确认”状态。

#### 读取/监视 BSEND/BRECEIVE 变量时的特点

如果 OPC UA 服务器在缓冲区中已存在 BSEND/BRECEIVE

监视项的旧值（变量），则激活订阅后会首先将缓冲区中的值（包括质量和时间戳）返回到客户端，然后再通过 SIMATIC NET 通信协议获取更多新值并返回新的质量和时间戳。  
缓冲区的值可能会比故障切换之前由冗余组另一服务器返回至客户端的值还要旧。

一般情况下，会发送信号指示状态为“良好”(Good)。

因此，读取/监视操作时客户端必须记录相应时间戳。

#### 4.5.8 SIMATIC NET OPC UA S7 A&C 对冗余故障切换的响应

在冗余故障切换之前，如果某种情况下出现更改，OPC UA S7 服务器 A 会将事件发送到客户端。在故障切换期间，不会向客户端发送事件。  
故障切换之后，OPC UA S7 服务器 B 会将事件发送到客户端。

如果故障切换之后，OPC UA S7 服务器 B 处于激活状态，OPC UA S7 服务器 B 会向客户端发送请求，以使用 RefreshRequired 事件 (Nodeld i=2789)  
执行每个事件监视数据项的刷新。此刷新请求可确保将条件的当前状态告知客户端。

S7 中断块将映射到具有相应报警状态的 OPC UA 报警条件（或简单报警）。

两个 OPC UA S7 服务器（服务器 A 和服务器 B）都连接到 S7 PLC  
连接，因而能够接收相同的消息。消息指示 S7 中断块的状态更改。尽管两个 OPC UA S7 服务器都接收消息，但不能认为这两台计算机是同步的。S7 PLC  
发送信号，以不确定的顺序和不确定的执行时间逐个指示所有连接的 OPC UA S7 服务器、其它操作员控制和监视站。如果有 2000 个消息，S7 PLC  
可能会将相同的消息先发送给 OPC UA S7 服务器 A，然后在 10 到 20 秒再发送给 OPC UA S7 服务器 B。

OPC UA 报警的更改事件具有 4 个相关时间戳：ReceiveTimeClient（客户端接收事件的 RTC 时间）、ReceiveTime（服务器接收消息的 RT  
时间）、Time（源的修改的时间标记时间戳 - S7 中断块上触发沿的修改时间）以及

## 4.5 对 OPC UA 冗余接口进行编程

S7Time (S7 中断块上触发沿源时间的时间戳)。以下信息仅与 ReceiveTime (RT) 和 S7Time 有关。

通常，由于不确定的执行时间以及计算机 A 和 B 上的时间设置稍有不同，OPC UA S7 服务器 A 和 B 相同消息的 ReceiveTime RT 总是稍有不同。

S7Time 由 S7 PLC 设置，并且相同沿更改的 S7Time 在两个 OPC UA S7 服务器上始终相同。

将事件订阅从 OPC UA S7 服务器 A 切换至 B 时，可能出现以下 4 种情况：

- S7 PLC 在故障切换期间会向两个 OPC UA S7 服务器发送消息，但此期间不会在客户端上触发事件。  
故障切换之后，会将新报警状态告知客户端。RT 将显示 OPC UA S7 服务器 B 的时间戳。
- 在故障切换之前的很短时间内，S7 PLC 会将消息发送至 OPC UA S7 服务器 A，在服务器 RT 时间戳带有新报警状态的情况下，将在客户端上触发事件。OPC UA S7 服务器 B 会从 S7 PLC 的其它点及时接收相同消息（甚至可能在 A 之前），不过这一操作是在完成故障切换之前，因而 OPC UA S7 服务器 B 不会在客户端上触发事件。刷新可以指示具有相同 S7Time 和其它相同属性但 ReceiveTime RT 不同的相同报警状态。  
如果记录事件（相关系统事件）以备日后检查，必须放弃计算机 A 或计算机 B 上后接收的报警状态。假定两台计算机的时间同步，这就是具有较新（更高）RT 的计算机。
- 在故障切换之前的很短时间内，S7 PLC 会将消息发送至 OPC UA S7 服务器 A，在服务器 RT 时间戳带有新报警状态的情况下，将在客户端上触发事件。  
在故障切换之后开始刷新之前，OPC UA S7 服务器 B 将从 S7 PLC 接收相同的消息。刷新步骤不可中断，换言之，OPC UA S7 服务器 B 可以在刷新之前或之后立即发送相应事件。这种情况下，将总共指示具有相同 S7Time 的相同报警状态三次。具有较晚（较高）RT 的两个事件来源于 OPC UA S7 服务器 B，并且由于记录的原因，需要放弃这两个事件。
- 在故障切换之前的很短时间内，S7 PLC 会将消息发送至 OPC UA S7 服务器 A，在具有此服务器 RT 时间戳的客户端上，仍然触发带有新报警状态的事件，OPC UA S7 服务器 B 将在刷新开始后从 S7 PLC 接收相同消息。  
因此，刷新将指示具有过时 S7Time 的过时报警状态。客户端将检测并放弃此状态。之后，OPC UA S7 服务器 B 将为与 OPC UA S7 服务器 A 相同的报警状态触发其它事件，该事件可因记录原因而丢弃。

如果向其中一台计算机提供消息的 S7 连接短暂丢失，OPC UA S7 服务器将再次建立该连接。成功建立连接之后，将在 S7 PLC 中查询有关 S7 中断块的状态。S7 PLC 提供的信息不完整，换言之，其中不包括附加值和 S7Time。

如果报警不完整，S7Time 将采用值 01.01.1990 00:00:01。由于该日期只能与这些不完整的报警一起出现，建议在相同报警状态具有多个事件时，排除这些不完整报警。

#### 4.5.9 有关 OPC UA S7 通信的常规建议和注意事项

PC 站是冗余的；S7 伙伴站通常是单独的。

这将产生高可用性，但却需要更多的组态工作。请注意以下建议和要点：

- 面向块的服务“BSEND/BRCV”和 S7 连接本身包括特定 PC 站和 OPC UA 服务器的地址参数。

对于 SIMATIC S7 站，为每个 S7 连接组态一个单独的函数块调用“BSEND/BRCV”。

建议使用 S7 永久连接。PLC 程序必须为两个 S7 连接同时执行函数块调用。

因此，PLC 程序会变得更复杂。

通常情况下，被动冗余端的函数块调用会被否定确认或需要被中止。

还应该注意 PC 站上的数据仅存在于本地。有故障切换时，PC 站上的冗余 OPC UA 服务器之间发送的数据不同步。

即使一端的函数块调用成功，也会因冗余故障切换而导致 OPC UA 客户端上的数据丢失。

- 对于 SIMATIC S7 站的每个 S7 连接，都需要对 S7 通信服务“PUT/GET”进行编程。

每个函数块调用都将查看相应 OPC UA 服务器的数据区。还应该注意 PC 站上的数据仅存在于本地。PC 站上的冗余 OPC UA 服务器之间发送的数据不同步。

- 对于 OPC UA 客户端，与冗余集的 OPC UA 服务器的连接是透明的。  
根据负载平衡标准，NLB 将连接分布到冗余服务器中。  
如果服务器故障，将重新连接冗余服务器。建议提前通过从激活的 OPC UA 服务器中移除网络电缆来特意试用此故障测试。还应该测试两个冗余服务器 S7 通信（变量）的质量。

要在服务器故障时实现与冗余服务器的 S7 变量之间的快速重新连接，建议组态已激活的且已永久建立的 S7 连接。

- OPC UA S7 冗余服务器不具有同步时钟的自动功能。  
为确保故障切换时不会出现不必要的时间跳变，需手动同步各冗余服务器的时钟。  
需要为此组态相关概念。
- 如果冗余服务器组中所有服务器的 ServerState 均为“正在运行”(Running)，则无需假定整个冗余组无错运行。这仅表示连接的 OPC UA 服务器处于“正在运行”(Running) 状态，且可与伙伴服务器进行通信。

冗余组中的另一台 OPC 服务器将指示其伙伴的 ServerState 为“未知的”(Unknown)，因为它无法与之通信。

如果可以与其伙伴服务器相连，则 OPC UA 服务器的 ServerState 为“正在运行”(Running)，ServiceLevel 为 255，否则为 127。值“255”和“正在运行”(Running)

表示冗余服务器组内的两台服务器均处于无误状态。

- 枚举冗余 OPC UA

服务器的端点时，也将枚举其自身地址的端点和冗余对公用地址的端点。

根据网络的组态，也可枚举其冗余伙伴的端点。

- 装载冗余组态时，如果先前的端口与组态冗余端点的端口相同，则会为当前的本地端点指定端口 14845。删除冗余组态时，会将端口 14845 的端点重新指定为其在冗余组态之前所具有的端口。

- 请注意，块服务的方法调用（例如 password()、blockread()）的结果对冗余故障切换之后的当前服务器不可用。必须为当前服务器组态方法调用并重新调用。

- 读取/监视服务器状态变量时的特点：

状态变量是服务器特定的，因此各服务器在同一个时间点的值会有所不同。

与此相关的典型示例为每台服务器“CurrentServerId”明显不一样。

即使在服务器运行期间，它们的时间戳也不会发生变化。

#### 4.5.10 由 OPC UA 客户端读出簇的状态

使用诊断变量，可以查明冗余支持的类型或 OPC UA 客户端与 OPC UA 服务器之间的连接状态。簇的 OPC UA 服务器将提供以下诊断变量。

可以使用标准化服务器对象将诊断变量显示在命名空间索引“0”中，而该命名空间索引位于“网络负载平衡管理器 (NLB) ”中的服务器节点之下。

在服务器节点下，您将看到对象“ServerRedundancy”。

这是一个具有属性“RedundancySupport”且类型为 ServerRedundancyType 的对象。

利用该对象，可以读出冗余支持的类型。

使用数据类型“enum\_OpcUa\_RedundancySupport”可以读出以下值：

- OpcUa\_RedundancySupport\_None = 0,
- OpcUa\_RedundancySupport\_Cold = 1,
- OpcUa\_RedundancySupport\_Warm = 2,

- `OpcUa_RedundancySupport_Hot = 3,`
- `OpcUa_RedundancySupport_Transparent = 4`

如果支持透明冗余，该对象的类型将为 `TransparentRedundancyType`。

该对象类型定义了其它元素 `CurrentServerId` 和“String”。

尽管所有服务器在透明模式下都具有同样的用于连接的

`URI`，但是当前正在使用的簇的服务器也可使用“`CurrentServerId`”标识符进行标识。

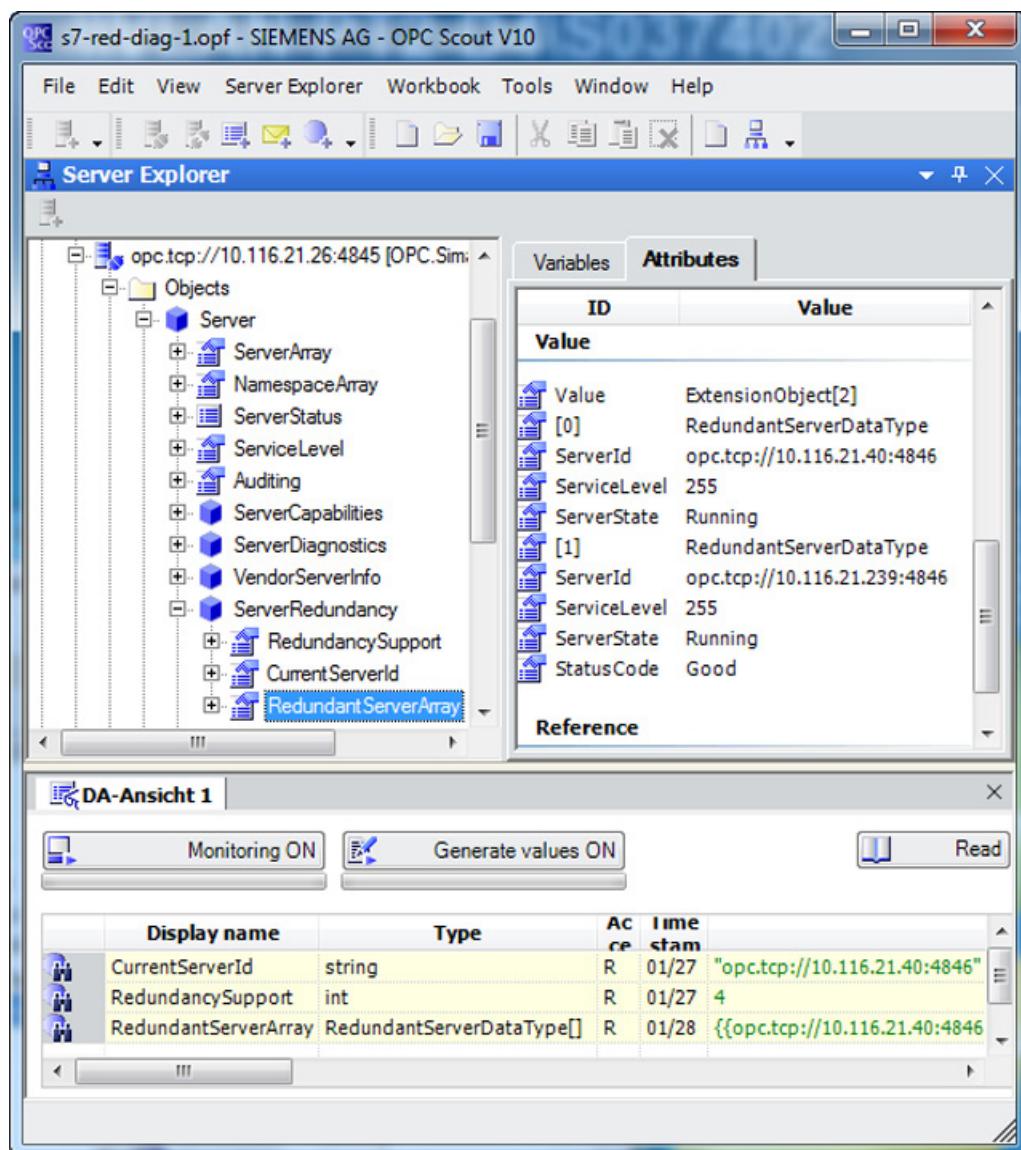
请注意，该值仅在会话中有效。

如果服务器故障或者多个会话故障，则簇的其它服务器可以接管其任务。

不过，客户端可以标识特定的数据源。

簇的所有服务器列表包含在类型“`RedundantServerDataType`”的“`RedundantServerArray`”中。该服务器列表包含各种服务器特定的信息或值 (`ServerId`, `ServiceLevel`, `ServerState`)。这些值可在会话中发生更改。

## 4.5 对 OPC UA 元余接口进行编程



此外，“RedundantServerDataType”结构包含以下元素：

- **ServerId**

服务器的 ID。包含簇中服务器的唯一标识符的字符串。它不能是服务器 URI，因为所有服务器的 ID 都必须相同。该字符串的结构与服务器 URI 的结构类似。然而，作为本地 PC 的一部分，唯一 IP 地址将被使用，而该地址也可用于服务器与服务器之间的通信。

- **ServiceLevel**

一个用于显示服务器与客户端之间的数据传送情况的字节。“0”（ $\triangleq$ “UNKNOWN”）表示最坏，“255”（ $\triangleq$ “RUNNING”）表示最好状态。此值将显示冗余服务器可用性的状态。

- **ServerState (enum OpcUa\_ServerState)**

可以采用以下值（其中的“UNKNOWN”状态表示该服务器无法访问）：

- RUNNING (OpcUa\_ServerState\_Running = 0)
- FAILED (OpcUa\_ServerState\_Failed = 1)
- NO\_CONFIGURATION (OpcUa\_ServerState\_NoConfiguration = 2)
- SUSPENDED (OpcUa\_ServerState\_Suspended = 3)
- SHUTDOWN (OpcUa\_ServerState\_Shutdown = 4)
- TEST (OpcUa\_ServerState\_Test = 5)
- COMMUNICATION\_FAULT (OpcUa\_ServerState\_CommunicationFault = 6)
- UNKNOWN (OpcUa\_ServerState\_Unknown = 7)

---

#### 说明

请注意，不能使用冗余的客户端诊断变量切换激活的服务器。

---



使用作为 SIMATIC NET PC 软件一部分的“SIMATIC 计算”软件，您可以创建简单的应用程序（使用该程序可以访问过程数据而不需要很多编程工作）。该过程数据可以通过任何 SIMATIC NET 通信系统进行读取或写入。

在“SIMATIC 计算”中有多种 .NET 程序集可用。可组态和互连这些控件以满足任务要求。

### .NET 的 OPC 数据控件

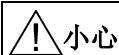
对于 .NET 接口，有一个 SIMATIC NET OPC 数据控件可用于简单的图形操作员控制。使用 Microsoft Visual Studio 可创建简单的应用程序。在此，可以找到 SIMATIC NET OPC 服务器命名空间中的过程变量并将其链接至用户界面元素，例如文本框。无需写入任何程序代码即可使用这些功能。有关此主题的详细信息，请参见“Auto-Hotspot”部分。

然而，对于更多的扩展应用，有一个可用的编程接口进一步简化了 OPC 接口。

### 与 OPC 的兼容性

SIMATIC NET 的 .NET 程序集和 ActiveX 控件直接使用 OPC 接口。

使用 SIMATIC NET 控件创建的应用程序只能与 SIMATIC NET PC 软件的 OPC 服务器搭配使用。



#### 小心

通过控件互连过程变量时，将建立对过程数据的访问。  
如果在控件中更改过程变量的值，则过程中的值可更改为直接结果。  
更改过程数据可能会在过程中触发意外响应，从而导致严重人身伤害或设备损坏。  
请记住这些，并且务必要多加注意。例如，应该限制过程变量的访问权限。  
为机器或过程安装物理紧急停止电路。

## 5.1 .NET OPC 客户端控件

### 5.1.1 概述

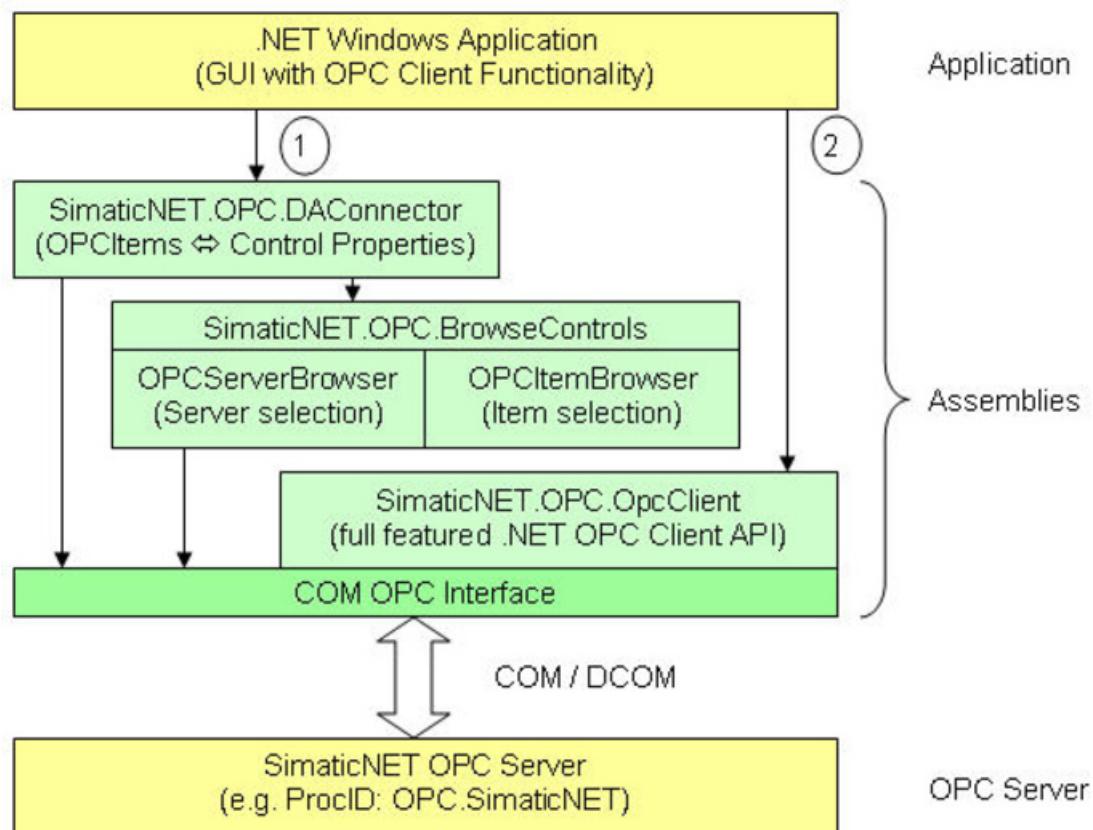
#### 后台

Microsoft Visual Studio 开发环境支持可在 .NET 编程语言下使用的附加控件的链接。

通过 SimaticNET .NET OPC 客户端控件, Siemens 提供了一个简单选项, 可对带有 MS Visual Studio 内标准控件 (如文本框、标签或按钮) 的 OPC 服务器中的数据点实施互连。这样, 无需编写一行代码便可创建简单的 OPC 客户端应用程序。概述中将此访问路径标记为“1”。

需要编写更为复杂的应用程序或处理来自 OPC 服务器的大量数据的用户应使用

SimaticNET .NET OPC Client API 接口。这一功能强大的数据接口基于 C++ 库, 但通过 .NET 简化了访问。概述中将此访问路径标记为“2”。



## .NET OPC 客户端控件的特性

.NET 控件和底层 C++ 库具有以下特性：

- 用户无需详细了解各种“OPC 数据访问”接口。
- 该组件将完全隐藏 OPC 的各种基本技术，例如 COM、DCOM、Web 服务、SOAP 和 XML。
- 如果出现错误，该组件将使用连接建立和重复的连接建立来完全隐藏对与 OPC 服务器之间的连接的处理。
- 使用 C-Sharp .NET 和 Visual Basic .NET 对 OPC 客户端应用程序进行的开发也适用于 SimaticNET .NET OPC 客户端控件，无需另外编程。
- 从各种“OPC 数据访问”接口到 .NET 数据类型的 OPC 数据转换。
- 在本地和网络上快速而简单地发现 OPC COM 服务器。
- 以图形方式支持在所选服务器上浏览 OPC 数据项。
- 通过实现 C++ 中的核心功能，进行客户端与服务器之间的高速而优化的通信。
- OPC UA 的支持及其证书管理。

### 说明

使用这些控件，只能建立与 SIMATIC NET 产品系列的 OPC 服务器之间的连接。  
无法与其它 OPC 服务器或其它供应商的 OPC 服务器进行连接。

## 5.1.2 步骤 1 - 安装

SimaticNET .NET OPC 客户端控件使用“SIMATIC NET PC 软件”的安装程序进行安装。

将组件安装在“%ProgramFiles%/Siemens/Simatic.Net/OPC2/OCX.NET”路径下。

SimaticNET .NET OPC 客户端控件包含以下文件：

- “SimaticNET.OPC.DAConnector.dll”
- “SimaticNET.OPC.BrowseControls.dll”
- “SimaticNET.OPC.OpcClient.dll”

### 5.1.2.1 引用控件

第一次使用 SimaticNET .NET OPC

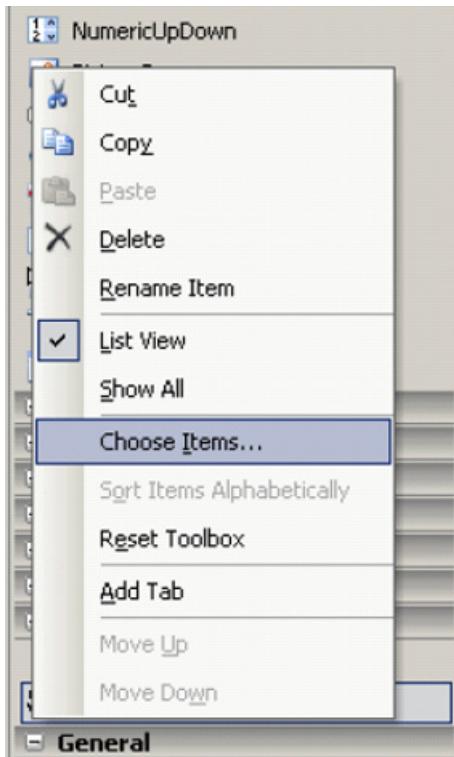
客户端控件之前，需要手动将这些控件添加到工具箱。

## 5.1 .NET OPC 客户端控件

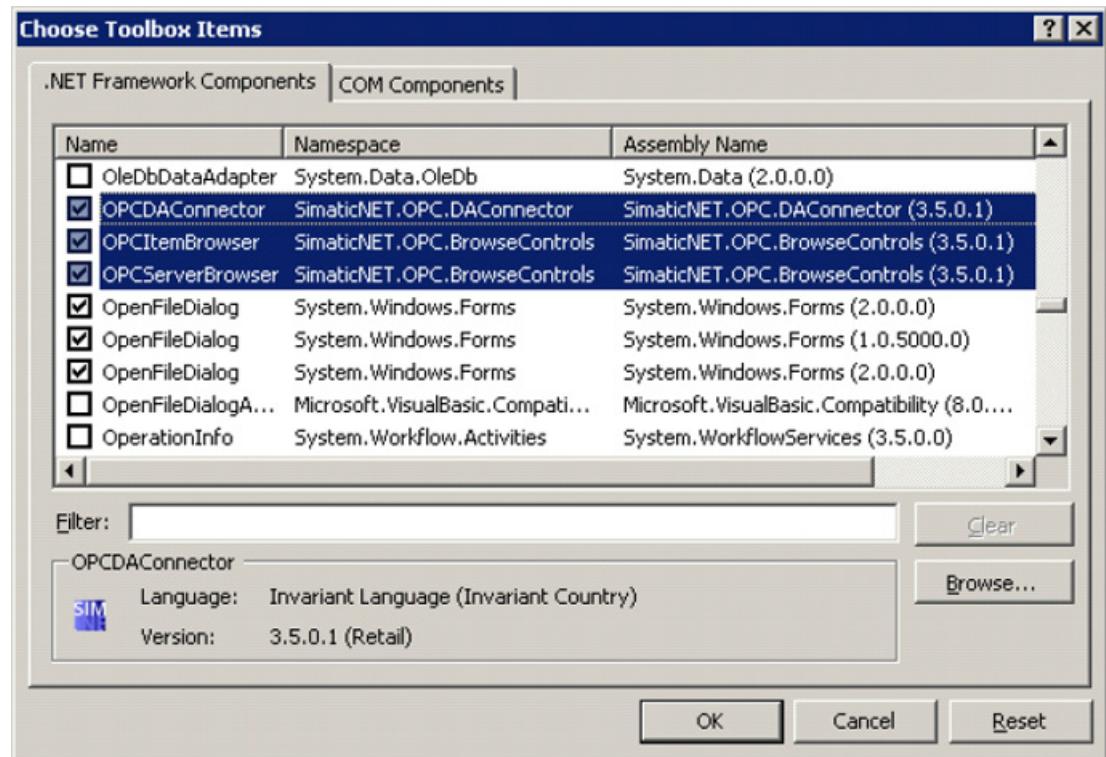
右键单击“工具箱”(Toolbox) 会看到一个子菜单，使用其中的“添加选项卡”(Add tab) 可创建新的选项卡，使用“选择元素”(Select elements) 可添加新的组件。

在下一个对话框中 (“选择工具箱项”(Choose Toolbox Items))，浏览到上方命名的安装路径。添加下列文件：

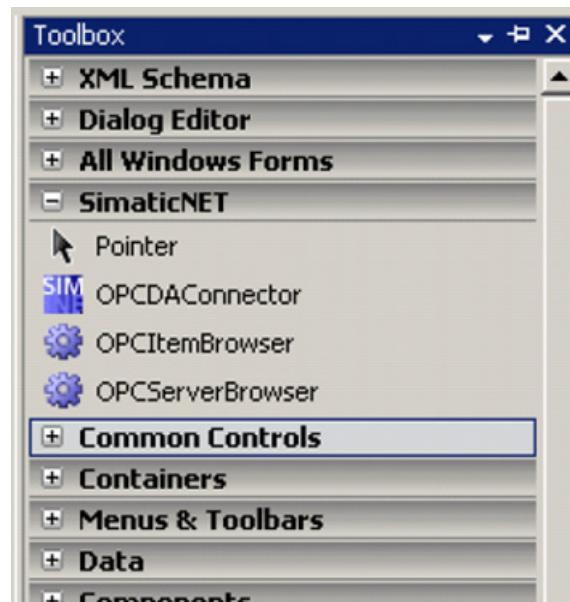
- “OPCDACConnector.dll”
- “OPCItemBrowser.dll”
- “OPCServerBrowser.dll”



⇒



在 Microsoft Visual Studio 的“工具箱”(Toolbox)  
中出现一个新选项卡，其中包含各个组件。



### 说明

MS Visual Studio 中引用的组件必须位于本地硬盘中，而不能位于网络驱动器上。这是 Microsoft .NET 开发环境的限制。

---

#### 5.1.2.2 设置示例程序

必须先启用 OPC 服务器的仿真连接，然后当前程序才能运行。示例程序使用了 S7 仿真连接的 **demo** 变量。

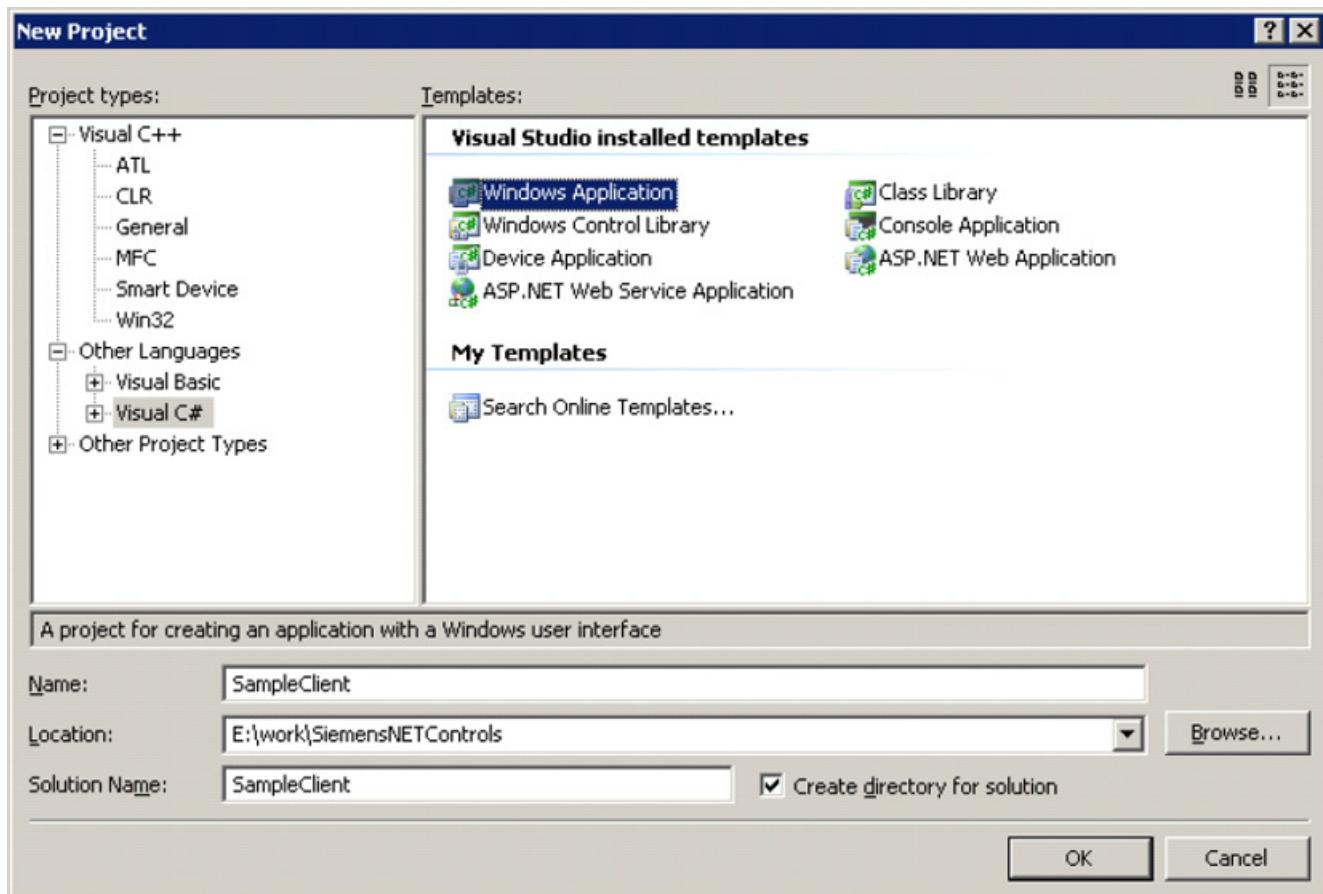
按照以下步骤为仿真选择虚拟模块 (DEMO):

1. 从“开始”(Start) 菜单中启动“通信设置”(Communication Setting) 程序：  
“开始 > SIMATIC > SIMATIC NET > 设置 > 连接设置”(Start > SIMATIC > SIMATIC NET > Settings > Communication Settings)
2. 在左侧的导航窗口中，打开 OPC 协议选择的属性页面。
3. 选择 S7 连接，然后单击旁边的箭头符号。
4. 在扩展的参数列表中，选择 D 复选框“为仿真提供虚拟模块 (DEMO)”(provide virtual module (DEMO) for the simulation)。
5. 单击“应用”(Apply) 按钮以确认所做的选择。
6. 关闭“通信设置”(Communication Settings) 程序。

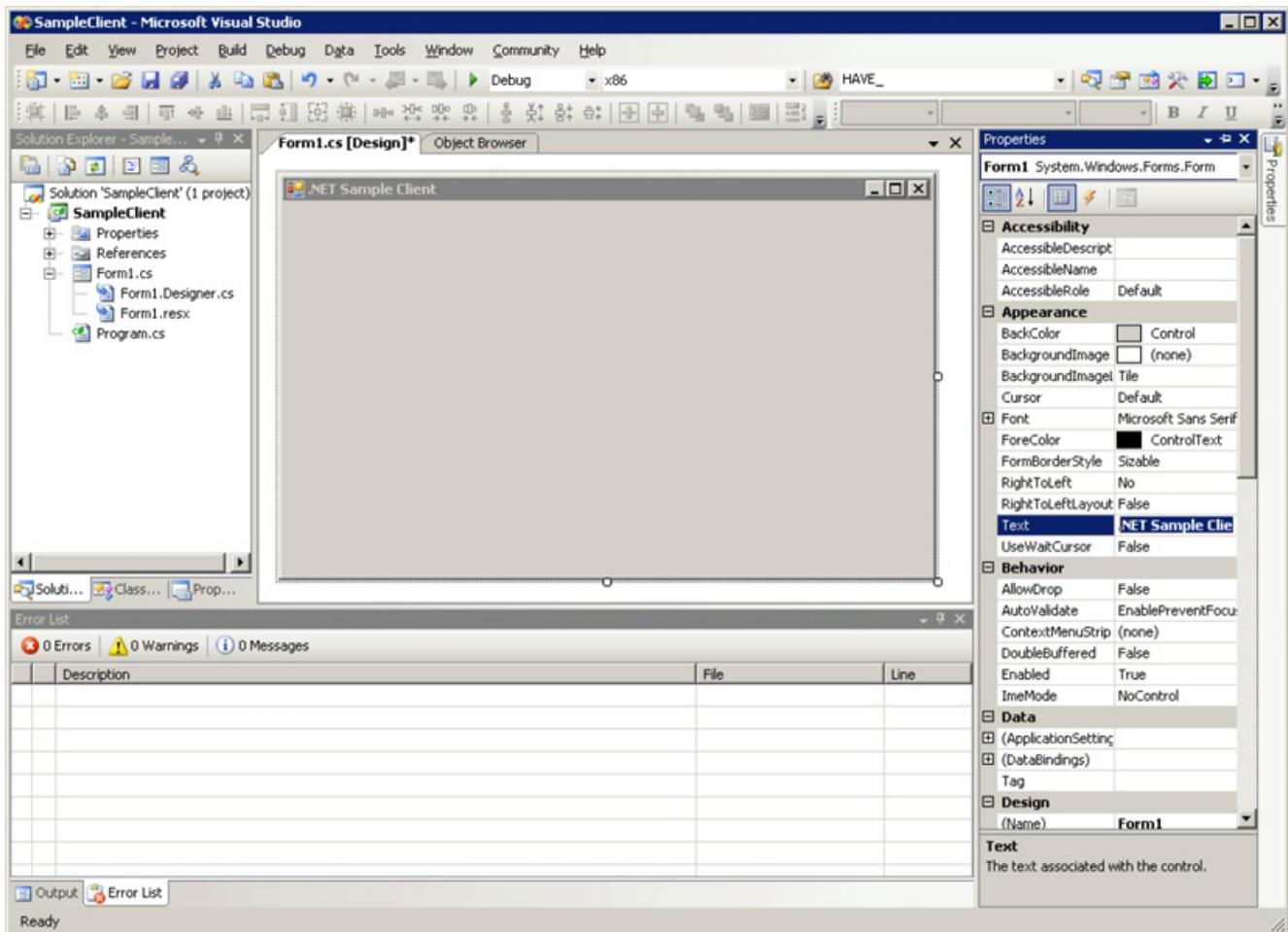


### 5.1.3 步骤 2 - WINDOWS FORM 中的控件

在 Microsoft Visual Studio 中，通过“Windows 应用程序”(Windows Application) 模板类型（此处也可采用 VB.NET）创建一个新的 C# 项目。

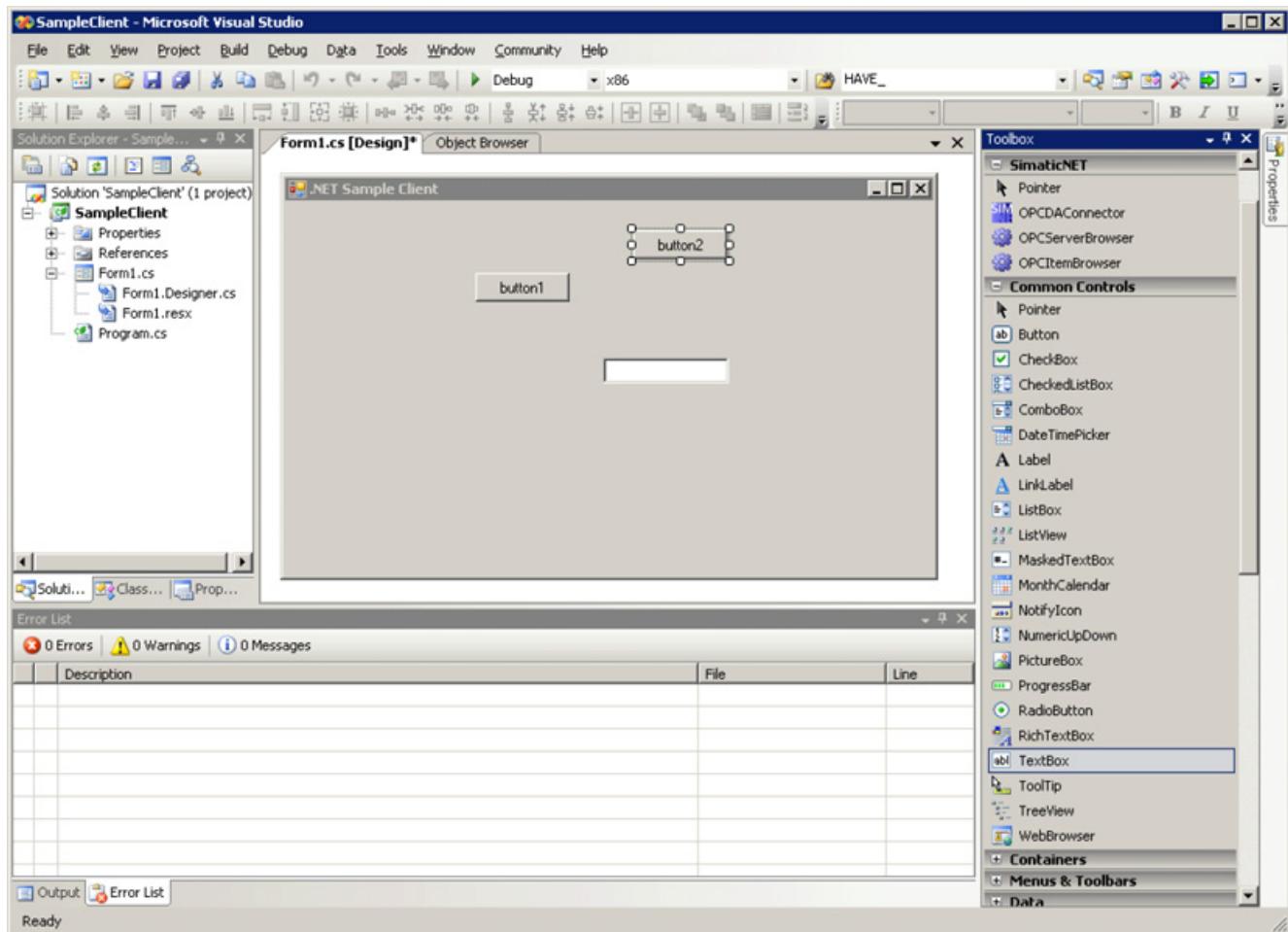


您会看到一个空的应用程序窗口（“窗体 1”(Form1)），可重命名其标题。



打开“工具箱”(Toolbox) 并将所需组件拖动到示例程序的“窗体设计器”(Forms Designer) 中。

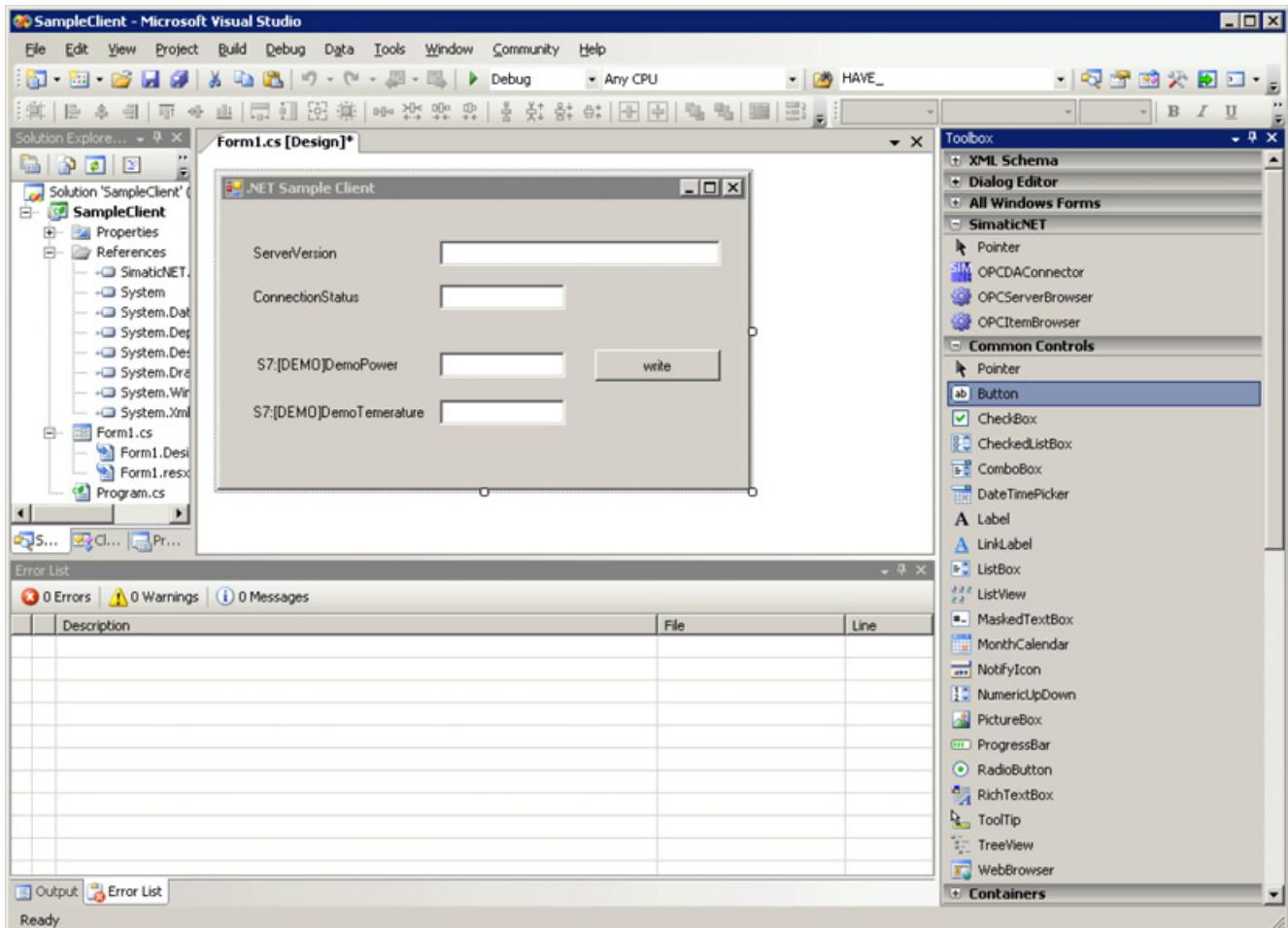
## 5.1 .NET OPC 客户端控件



### 5.1.3.1 添加 Windows 控件

如图所示，在空窗体中添加四个标签、四个文本框和一个按钮。

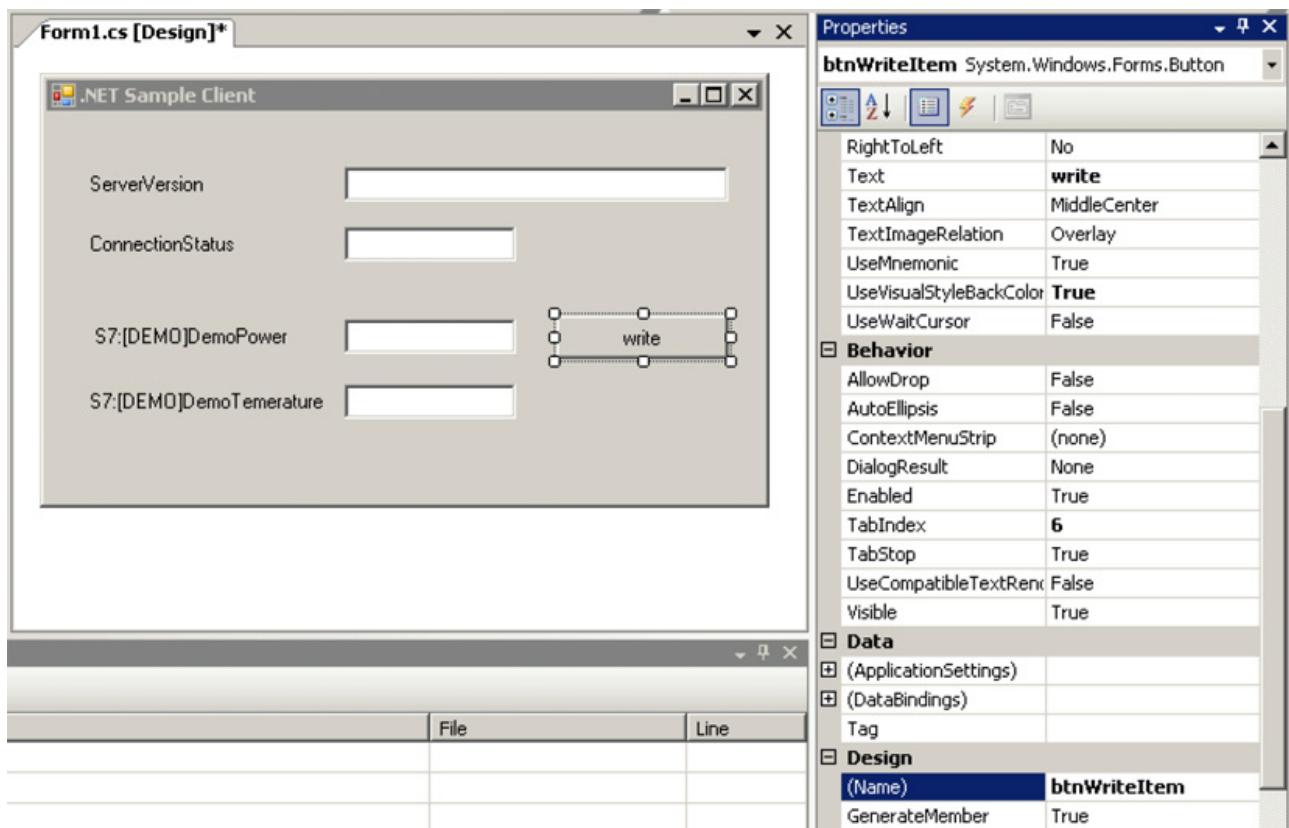
可在“公共控件”(Common Controls) 选项卡中找到这些组件。放置组件，如图所示。



对示例使用下表中的名称（如果可行），因为在后续解释时会引用这些名称。

打开属性窗口，然后选择要编辑其属性的组件。

## 5.1 .NET OPC 客户端控件

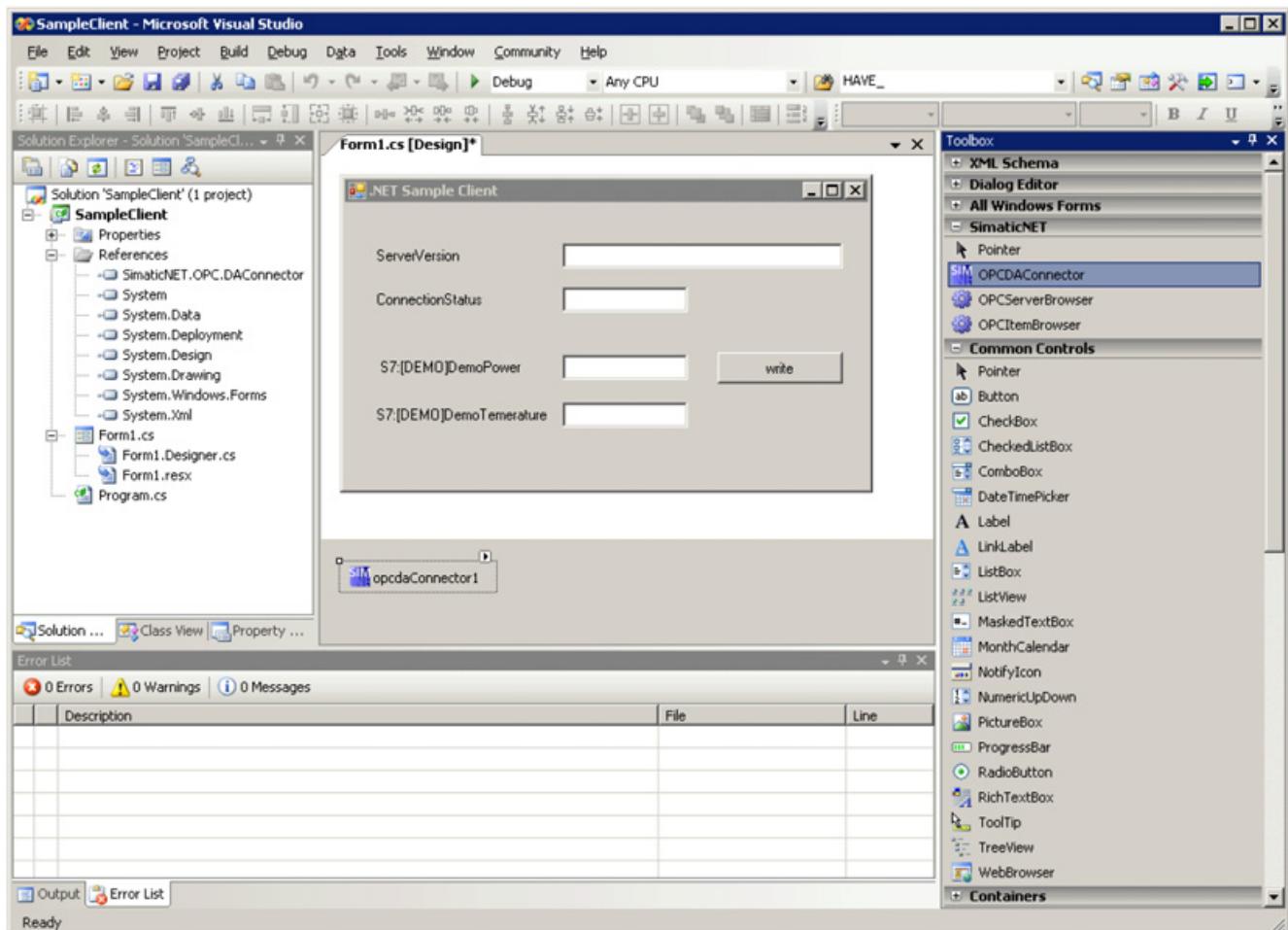


对要添加的所有组件重复此步骤。

源名称	新属性(名称)	新属性(文本)	新属性(启用)
Label1	lblServerVersion	ServerVersion	True
Label2	lblConnectionStatus	ConnectionStatus	True
Label3	lblItemID1	S7:[DEMO]DemoPower	True
Label4	lblItemID2	S7:[DEMO] DemoTemperature	True
TextBox1	txtServerVersion		False
TextBox2	txtConnectionStatus		False
TextBox3	txtDemoPower		True
TextBox4	txtDemoTemperature		True
Button1	btnWriteItem1	write	True

### 5.1.3.2 添加 .NET OPC 客户端控件

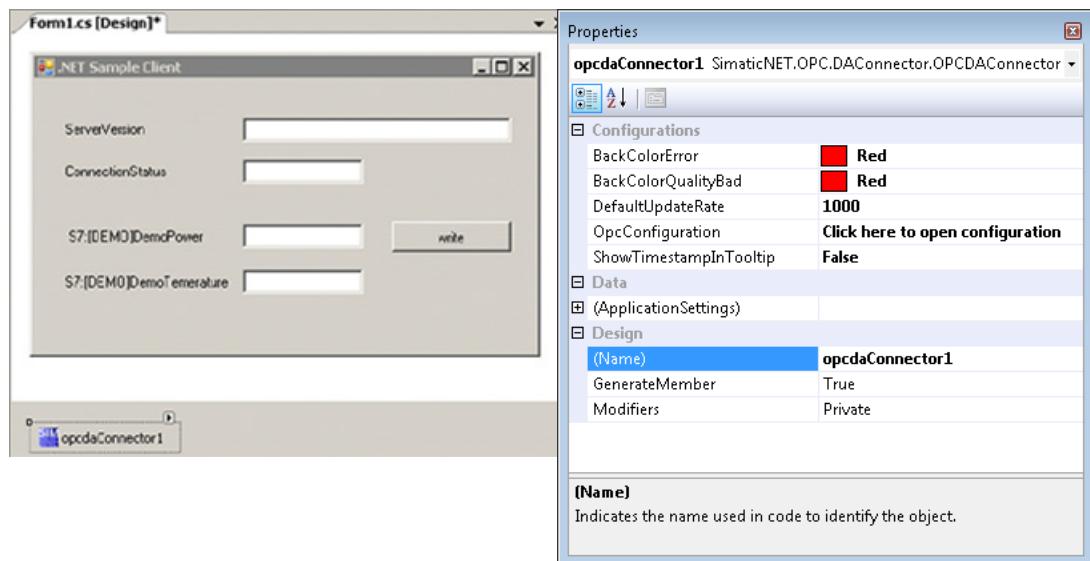
现在将“OPCDAConnector”控件从“工具箱”(Toolbox) 中取出并将其拖拽至应用程序窗口。此步骤可将整个 OPC 客户端功能链接至应用程序。



可在属性窗口中进行 OPC 客户端控件的常规设置。

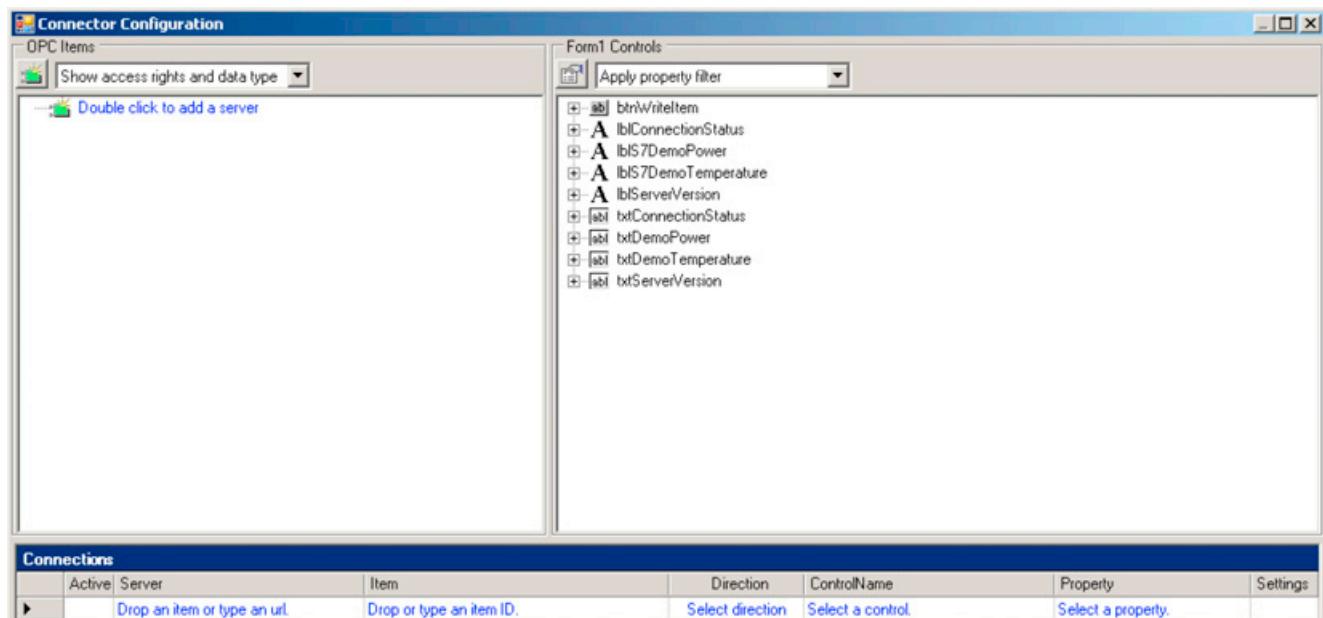
- 使用“组态”，打开用于链接 OPC 数据项的组态对话框。
- 以毫秒为单位指定更新时间 (DefaultUpdateRate) 并将其传递到 OPC 服务器以允许其将数据变化传送至客户端应用程序。
- 为与 OPC 数据项链接的全部控件设置背景颜色 (BackColorError)。如果出现了 OPC 错误，则互连控件的背景颜色将变为此处所设置的颜色。
- 为与 OPC 数据项链接的全部控件设置背景颜色 (BackColorQualityBad)。若返回的 OPC 值的质量变为“差”，则互连控件的背景颜色将变为此处所设置的颜色。
- OPC 时间标记的显示会在已链接控件的工具提示中出现，其中也显示了时间（UTC 时间）和值。

## 5.1 .NET OPC 客户端控件



### 5.1.4 步骤 3 - 组态数据链接

打开此对话框，单击 OPC 客户端控件的属性窗口中的“组态”(Configure) 框或双击“窗体设计器”(Forms Designer) 中的组件。

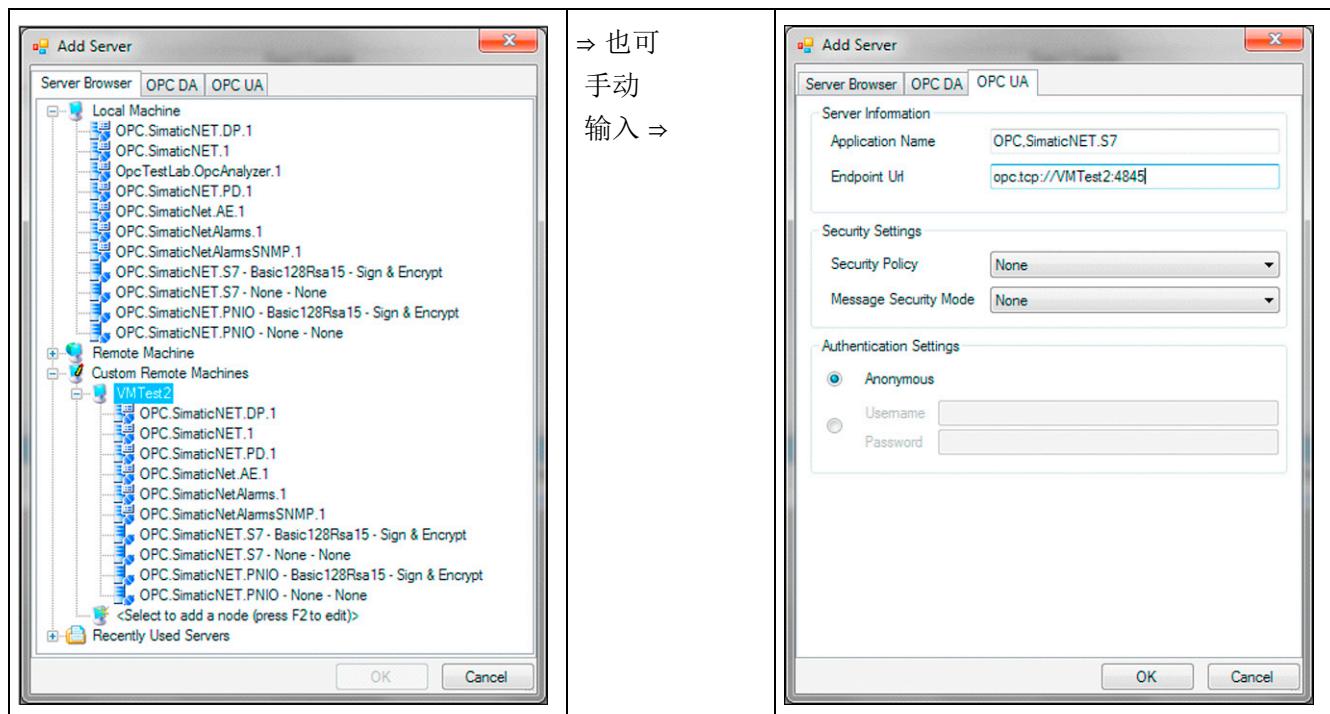


实际上，对话框由三部分组成：

- 左上方：OPC 浏览器窗口  
可在此处选择 OPC 服务器然后选择要进行互连的 OPC 数据项
- 右上方：Windows 组件浏览器  
可在此处选择要进行互连的 Windows 控件的属性
- 下方：全部链接的列表  
若要建立 OPC 数据项和控件属性的链接，可分别将它们拖至此列表

#### 5.1.4.1 浏览 OPC 数据项

在 OPC 客户端连接器对话框的左上方窗口中，可看到 OPC 服务器及其变量（OPC 数据项）。首先需要添加一个 OPC 服务器。打开（“添加服务器”(Add Server)）对话框后，可通过单击绿色图标或双击所显示的文本添加 OPC 服务器。



可以选择一个本地 OPC 服务器，稍后应用程序将尝试本地启动此服务器。

如果在网络（远程计算机）中选择 OPC

服务器，则计算机名称将储存到应用程序中且稍后还会用到该名称。确保有足够的 DCOM 权限启动这样的远程 OPC 服务器并与它们建立连接。在第二个选项卡 (OPC DA) 中，有手动指定 OPC 服务器的选项 (ProgID 和主机名称或 IP 地址)。

---

#### 说明

使用这些控件，只能建立与 SIMATIC NET 产品系列的 OPC 服务器之间的连接。

无法与其它 OPC 服务器或其它供应商的 OPC 服务器进行连接。

---

如果已添加了一个 OPC 服务器，则可浏览其地址空间并选择要互连的 OPC 数据项。

请记住，在可进行互连之前，OPC 数据项必须有所需的访问权限。

可更改浏览器视图以通过窗口左上方的组合框来显示访问权限和数据类型。

选择两个可能的视图之一：

- “显示访问权限和数据类型”(Show access rights and data type)

除了数据项级别的访问权限，还会显示 OPC 数据项的数据类型

- “仅显示数据项名称”(Show only item names)

仅显示 OPC 数据项的名称

由于有些 Windows

控件包含大量属性，因此可在窗口右上角使用组合框来为此视图设置筛选器。

选择两个可能的视图之一：

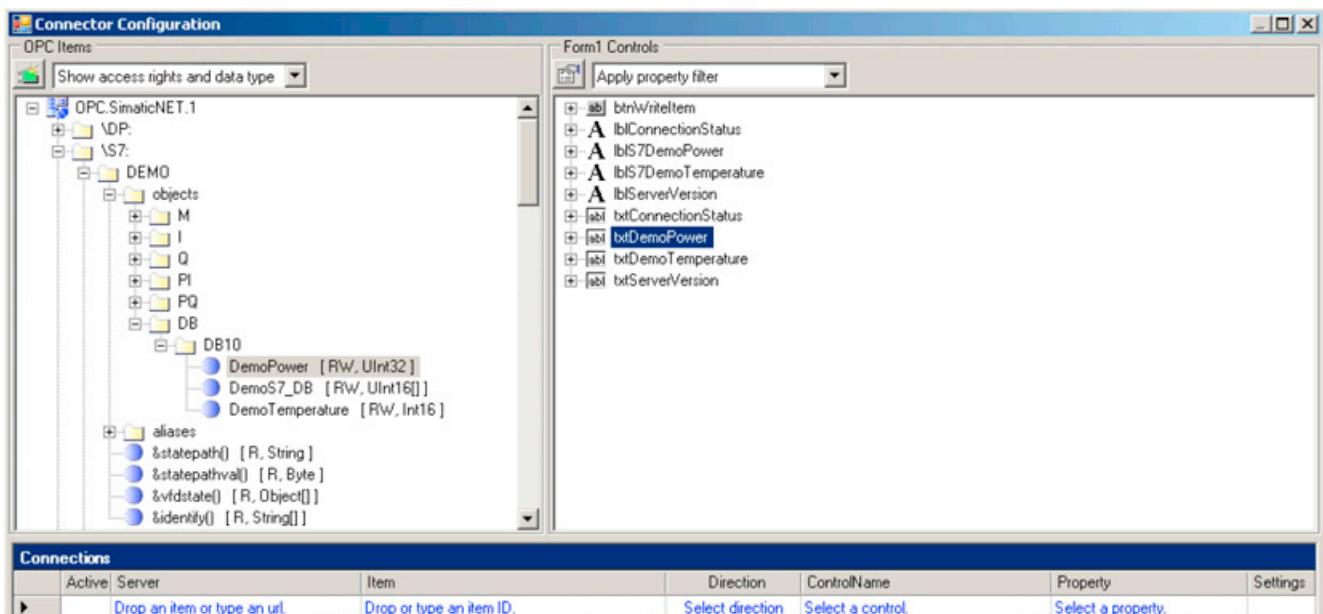
- “应用属性筛选器”(Apply property filter)

此处，显示内容取决于筛选器设置。

可以筛选数据类型、访问权限和属性级别。筛选器可在使用组合框 (“应用筛选器”(Apply Filter)) 左侧按钮打开的对话框中进行设置，请参阅“浏览 Windows 控件属性 (页 658)”部分。

- “显示所有属性”(Show all properties)

显示 Windows 控件的所有属性。



现在，将 OPC 数据项“&version()”、“&statepath()”及 demo

连接“DemoPower”和“DemoTemperature”的变量拖动到下方窗口的链接列表中。

### 说明

连接到“统一架构”服务器 (UA

服务器) 时，根据所选端点，可能必须先在客户端与服务器之间交换证书，然后才能进行连接 (请参见“为 NET OPC 数据控件创建 OPC UA 证书 (页 671)”部分)。

### 访问权限

要读写 OPC 数据项，该数据项必须具有访问权限 RW (读写)。

## 数据类型

必须还要考虑到 OPC 数据项的数据类型，因为它应匹配将与之互连的 Windows 控件的属性的数据类型。 OPC .NET

客户端控件会请求与互连属性相匹配的数据类型中的 OPC

数据项，然而不是所有的转换都可实现，OPC 服务器也并不支持所有的转换。

## 链接状态

如果无法实现转换或者会出现丢失数据的结果，则链接（连接）列表中的相关图标会指示该状态。将鼠标指针悬停在图标上可获取更多信息。

可能的符号包括：

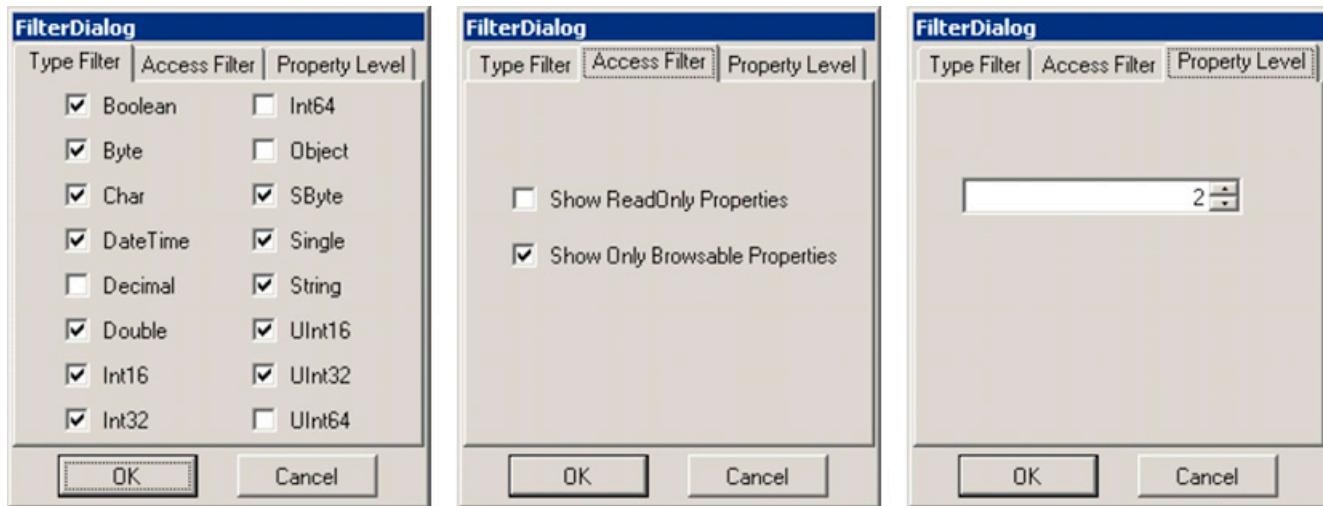
- 红色十字  
无法互连或互连不完全
- 橙色复选标记  
可互连，但数据类型仅在特定限制下才匹配
- 绿色复选标记  
互连正常

### 5.1.4.2 浏览 Windows 控件属性

在“连接器组态”(Connector Configuration)

对话框右上方的窗口中，显示了应用程序窗口（“窗体 1”(Form1)）中包含的所有 Windows 组件。可打开层级视图来选择要互连的属性。

要使视图更加清晰，可使用筛选器来隐藏因数据类型而不能考虑互连 OPC 数据项的属性。



## 数据类型

相关属性的数据类型通过图标显示。例如，文本框的“文本”属性具有“字符串”类型。  
由于所有 OPC 数据类型都能转换为字符串，因此 OPC  
数据项的每个值都能在文本框中显示。

## 访问权限

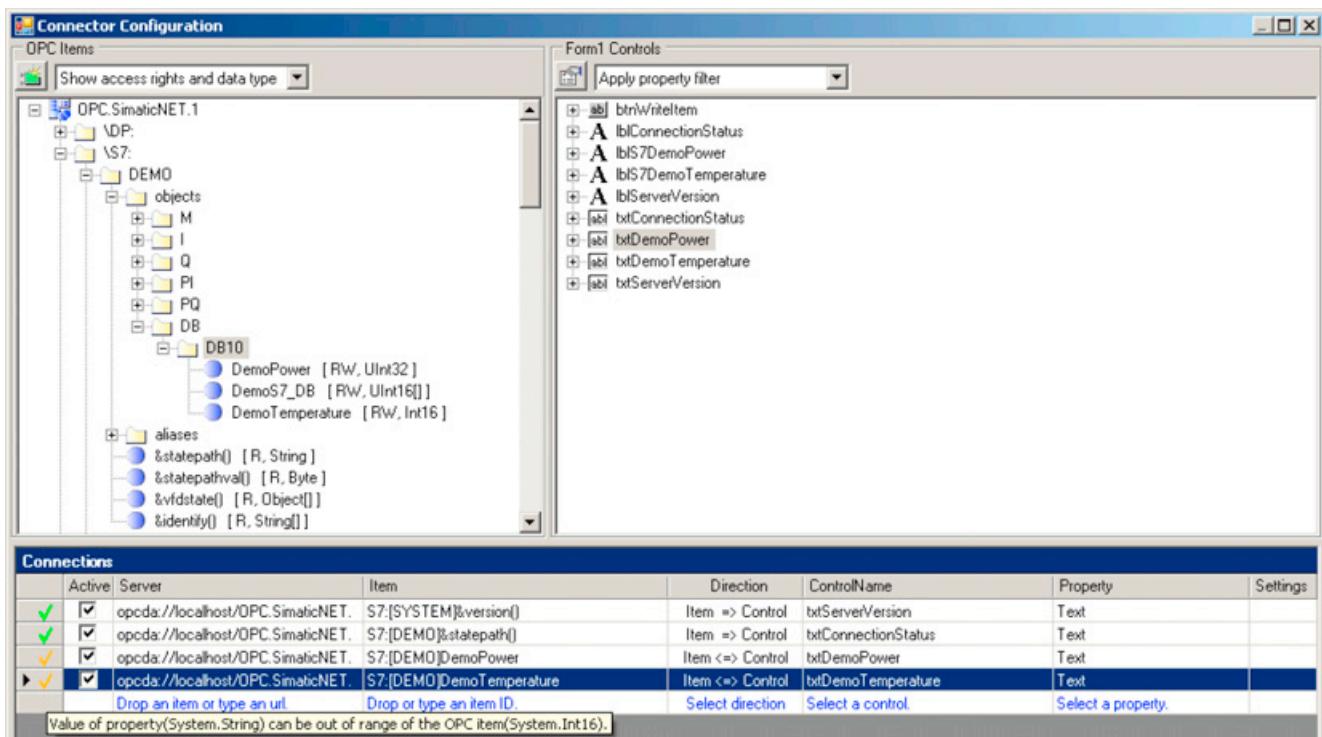
请确保所选属性具有所需访问权限，并且与 OPC 数据项的互连可实现。  
例如，某些属性只能读取不能写入。

现在，将 Windows 窗体相关文本框控件的文本属性拖拽到下方窗口“连接”(Connections)  
中。这会在此行创建与 OPC 数据项的链接。

### 5.1.4.3 创建链接列表

在“连接器组态”(Connector Configuration)

对话框下部的窗口中，可看到已创建链接（连接）的列表。使用拖放功能，将 OPC  
数据项连接到 Windows 控件属性并设置数据传送方向。此操作指定了数据的传送方向。



链接列表会自动检查创建的链接是否正常工作，并基于不同的图标（绿色复选标记、橙色复选标记和红色十字）显示出来。可将鼠标指针悬停在图标上以获取更多信息。  
必须解决显示的所有冲突，从而实现完整功能的连接。

可能的符号包括：

- 红色十字  
无法互连或互连不完全
- 橙色复选标记  
可互连，但数据类型仅在特定限制下才匹配
- 绿色复选标记  
互连正常

## 后续编辑和禁用

可通过单击相关框并编辑行，来编辑列表中已输入的链接。

使用复选标记（“激活”(Active)），可暂时禁用链接。

## 多个互连

OPC 数据项可与多种 Windows 控件或同一控件的不同属性进行多次互连。

## 传送方向

下表显示了数据的传送方向。

方向	功能	说明
数据项 ⇒ 控件	读取 OPC 数据项	每次 OPC 数据项的值发生更改时，该值都会写入控件的已链接属性。
数据项 ⇐ 控件	写入 OPC 数据项	触发事件触发时，属性值将写入已链接的 OPC 数据项
数据项 ⇄ 控件	双向传送	OPC 数据项的值的每次更改都将写入属性，并且当触发器触发时，属性的当前值将写入 OPC 数据项。 可通过设置选项“控件有焦点时禁用数据更改”(Disable Datachange when Control has focus) 来阻止写入数据项值的更改。

正确设置已链接的数据连接的传送方向。 OPC

数据项“&version”和“&statepath”为只读数据项，并且通过“数据项 ⇒ 控件”链接至 Windows 控件“txtServerVersion”和“txtConnectionStatus”的文本属性。对于 OPC 数据项“DemoPower”和“DemoTemperature”，选择双向传送（数据项 ⇄ 控件）。

变量的双向互连显示为“功能”，但带有一个橙色复选标记。尽管 OPC 数据项的值（整数）能够正确无误地写入控件的文本属性（字符串），但在相反方向可能会出现错误，因为此时可在文本框中输入的大量数字，甚至字母无法再写入 OPC 数据项，否则将导致错误的出现。

#### 5.1.4.4 用于写入的触发器

要写入到 OPC 数据项，需要触发器事件。

必须先对这些触发器事件进行组态然后才能开始写入操作。单击“连接器组态”(Connector Configuration)

对话框的连接列表中最右列（“设置”(Settings)）的相关单元格后，会出现一个按钮，此时必须为每个用于写入 OPC 数据项的链接组态触发器事件。

Connections						
Active	Server	Item	Direction	ControlName	Property	Settings
✓	<input checked="" type="checkbox"/> opcda://localhost/OPC.Sim	S7:[SYSTEM]&version()	Item => Control	txtServerVersion	Text	
✓	<input checked="" type="checkbox"/> opcda://localhost/OPC.Sim	S7:[DEMO]&statepath()	Item => Control	txtConnectionStatus	Text	
▶ ✓	<input checked="" type="checkbox"/> opcda://localhost/OPC.Sim	S7:[DEMO]DemoPower	Item <=> Control	txtDemoPower	Text	
✓	<input checked="" type="checkbox"/> opcda://localhost/OPC.Sim	S7:[DEMO]DemoTemperature	Item <=> Control	txtDemoTemperatur	Text	
	Drop an item or type an url.	Drop or type an item ID.	Select direction	Select a control.	Select a property.	

用于对链接的触发器事件进行组态的对话框有两个选项卡，分别用于设置常规行为和特定行为。

#### 常规设置：

- 更新速率

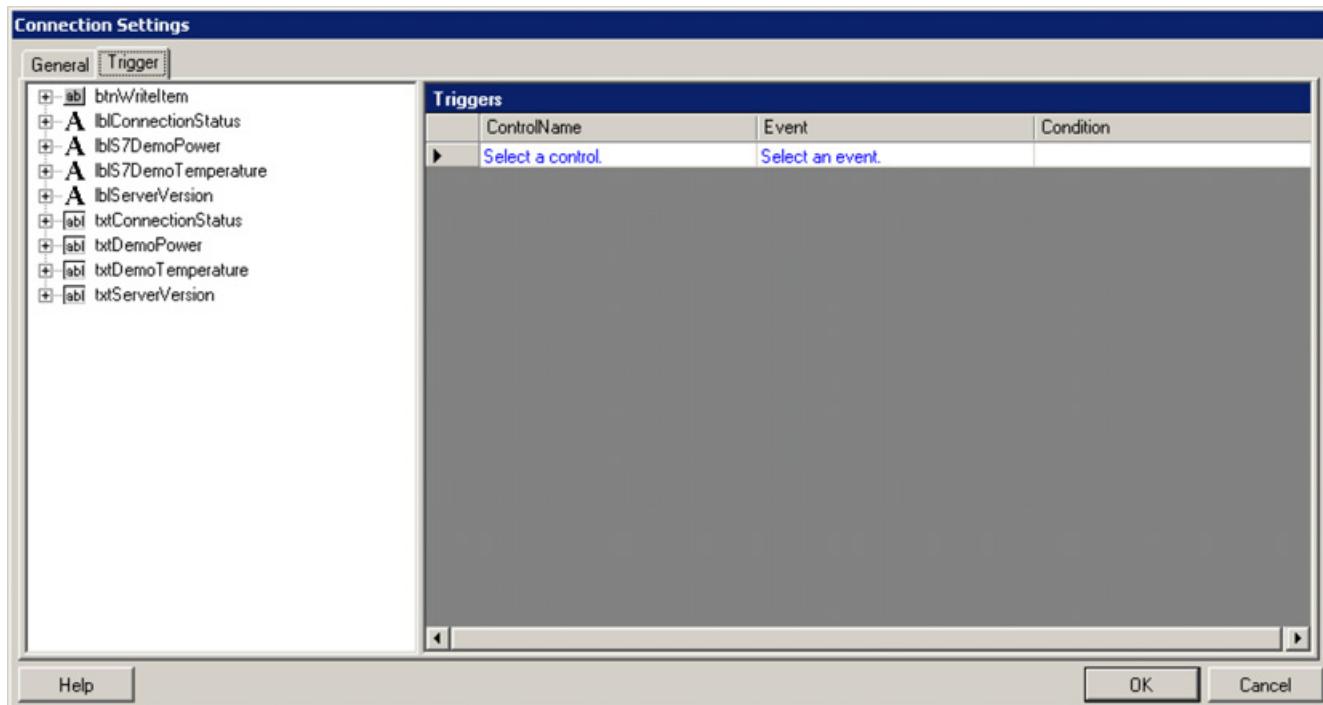
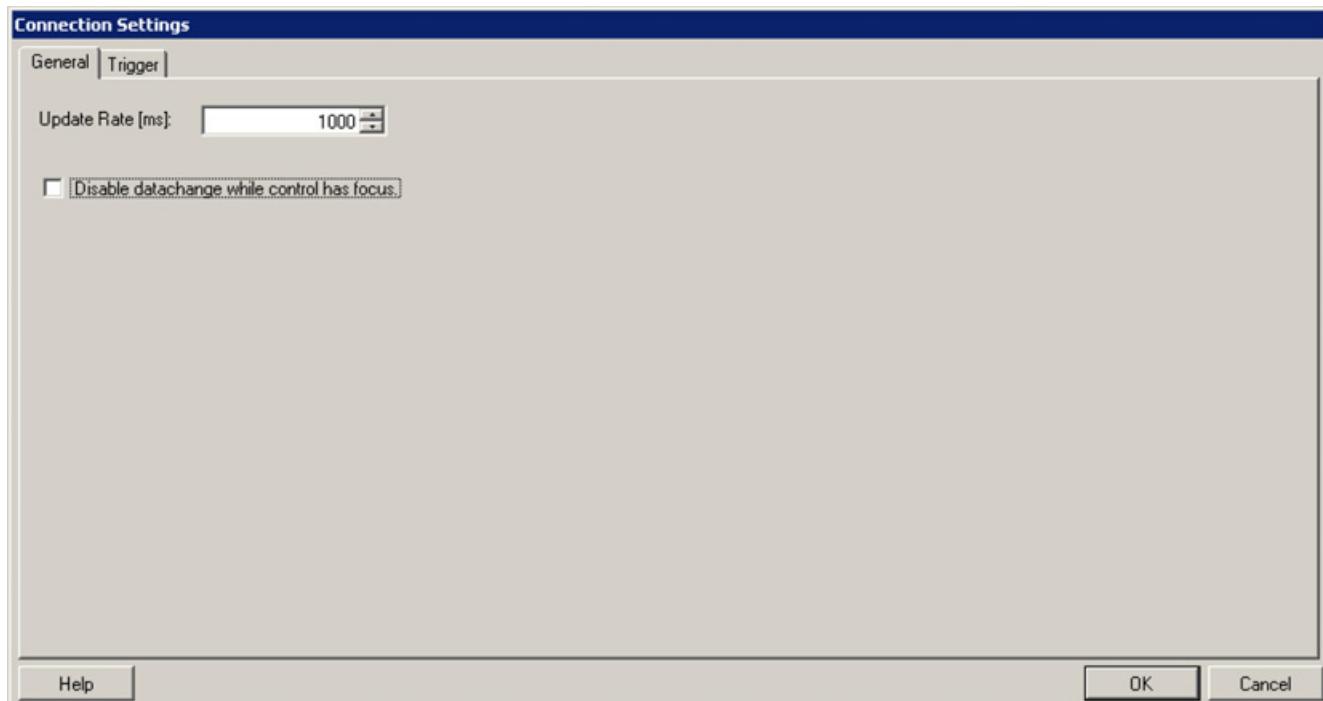
OPC 数据项为监视而向服务器注册时的以毫秒为单位的更新速率

- 控件有焦点时禁用数据更改

只要控件有焦点就暂时禁用更新；此项设置应与双向传输一起使用

## 触发器设置

- 选择事件
- 在此事件中设置特殊条件



## 多个触发器事件

也可组态多个触发器事件。

当这些触发器事件中的其中一个触发时，始终为此互连执行写入操作。

在触发器对话框的左侧，选择相应 Windows 控件的事件并将其拖至触发器列表。

通过同样也能返回参数的事件，可将它们作为条件进行查询。

当控件有文本框“txtDemoPower”和“txtDemoTemperature”的焦点时（触发器的常规设置）禁用框的更新。

为文本框“txtDemoPower”或“txtDemoTemperature”的数据链接组态下列特殊的触发器事件：

控件名称	事件	条件	说明
btnWriteItem1	Click	--	单击 btnWriteItem1 按钮后，将文本框 txtDemoPower 的文本属性的当前值写入 OPC 数据项 S7: [DEMO]DemoPower。
txtDemoTemperature	KeyDown	返回	当 KeyDown 事件有“返回”值时（即按下返回键），将文本框 txtDemoTemperature 的文本属性的当前值写入 OPC 数据项 S7: [DEMO]DemoTemperature。

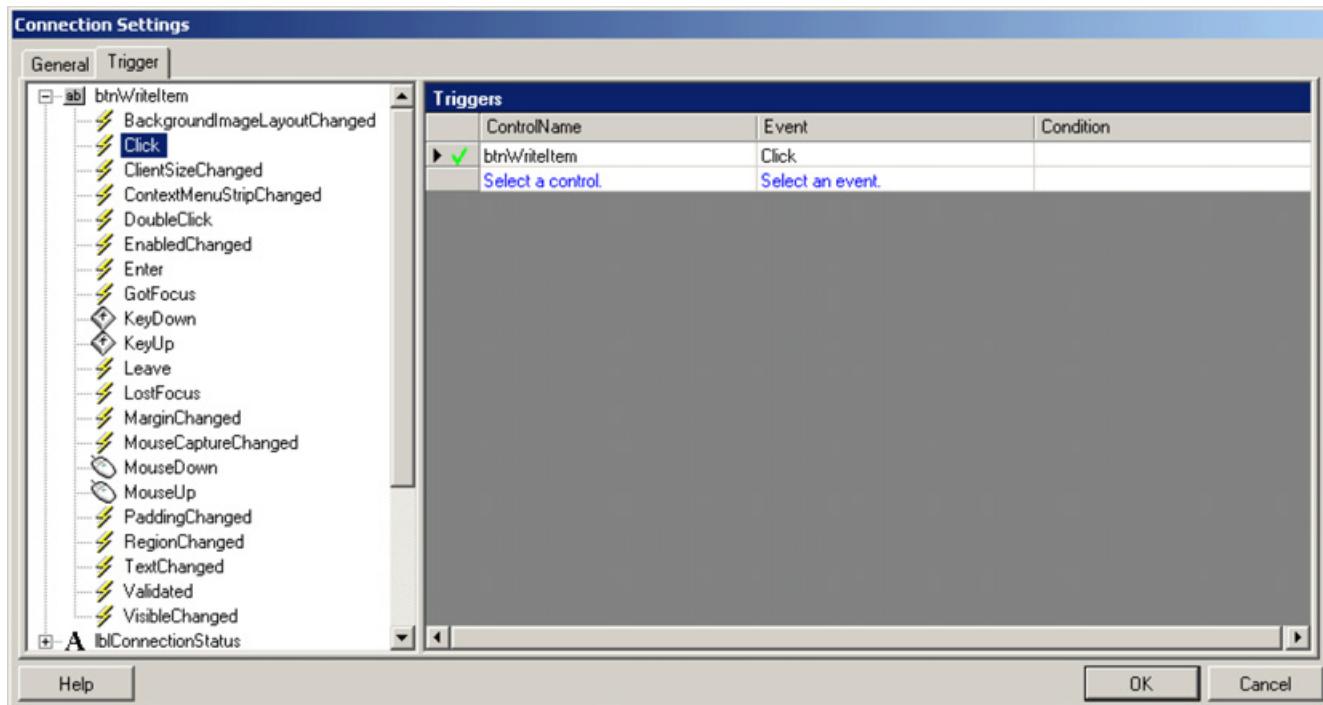
这两个示例说明了写入 OPC 变量时数据控件的行为。

### 示例 1:

如果文本框有焦点，即光标位于文本框内时，文本框不再随 OPC 值更新（控件有焦点时禁用数据更改）。

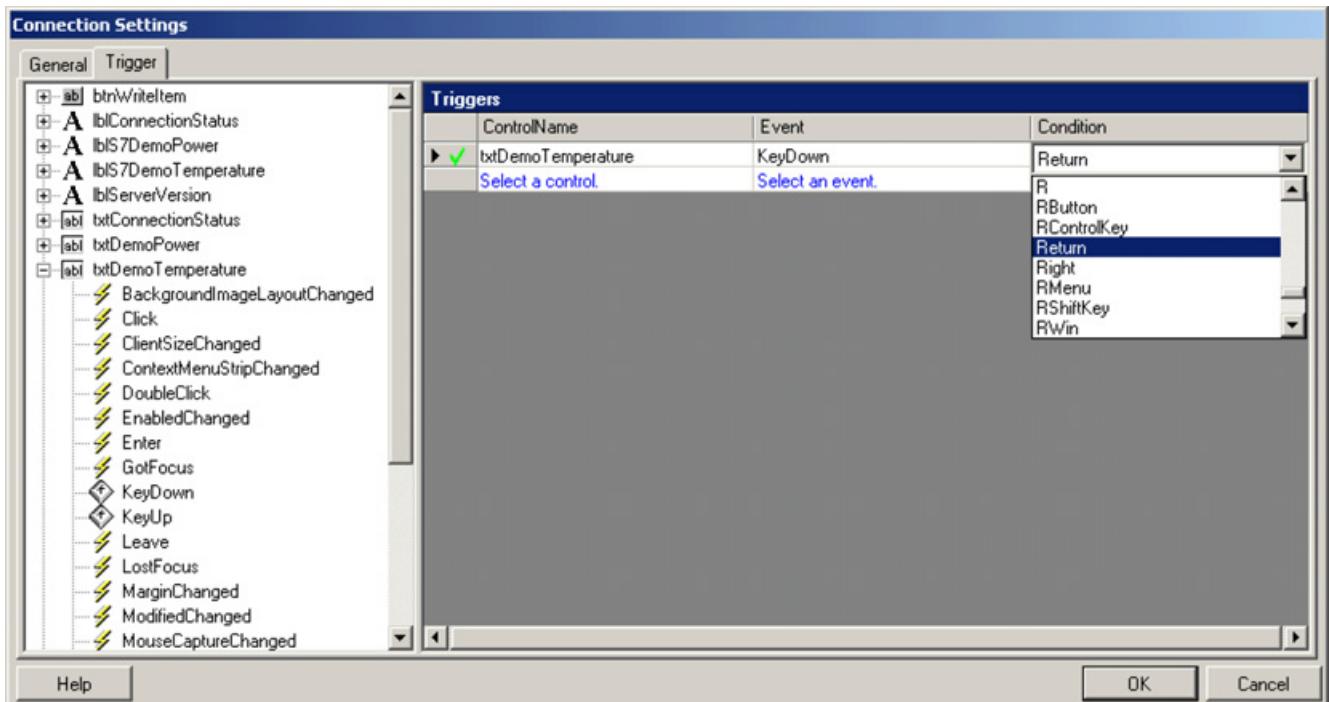
这可防止用户在进行编辑操作时文本框内的值突然发生更改。

这使得用户可以安静地编辑文本框并且只有在单击“btnWriteItem1”按钮后才会触发单击事件。之后会触发对 OPC 服务器的写入作业并将文本框的当前值写入 OPC 数据项。



### 示例 2:

如果在没有任何条件的情况下使用 **KeyDown** 事件，则当通过每个 **KeyDown** 事件写入值“152”时都会把所有的三个写入作业（**Write(1)**、**Write(15)** 和 **Write(152)**）发送到 OPC 服务器。然而，若还要组态“返回”条件，则要在没有值写入 OPC 服务器的情况下编辑文本框。只有检测到返回键的 **KeyDown** 事件才能将文本框中的值写入 OPC 服务器。



#### 5.1.4.5 OPC 质量和错误

OPC 属性“质量”和“错误”的链接适用每个 OPC 数据项，由 DataControl 自动进行处理。此 OPC 属性的单独互连无法通过“SimaticNET .NET OPC 客户端控件”来实现。

## 5.1 .NET OPC 客户端控件

### Quality

读取 OPC 数据项的 OPC 质量指定所读取的值是否可信。

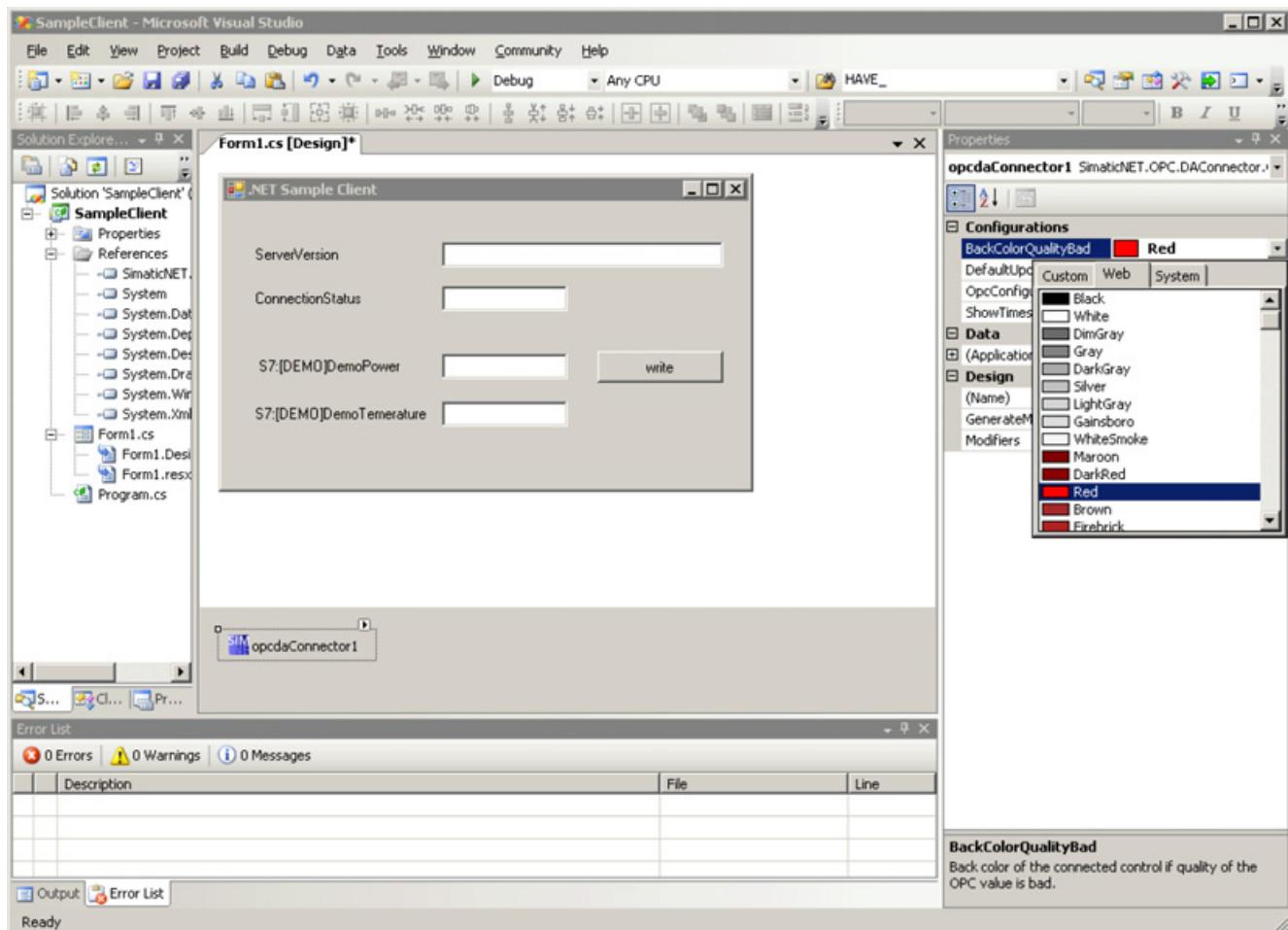
质量可以是“良好”、“不确定”或“不良”，将被映射到互连的 Windows

控件的背景颜色（BackColor 属性）中，同时也会显示在工具提示中。

相应地，如果返回的“OPC

值”不是“良好”，则某些项目（例如文本框）的背景颜色会更改为红色（默认）。

此互连是自动建立的，在“连接器组态”对话框中，OPC 值只是简单地链接到“Windows 属性”控件中。可以为整个“OPCDAConnector”对象修改颜色的一般设置。



## 错误

如果产生错误（例如在写入时），会返回一个错误代码。

描述该错误的相关文本会显示在互连的 Windows 控件的工具提示中。

如果将鼠标指针悬停在其上方，即可显示最后一条错误文本。

此互连同样是自动建立的，在“连接器组态”对话框中，OPC

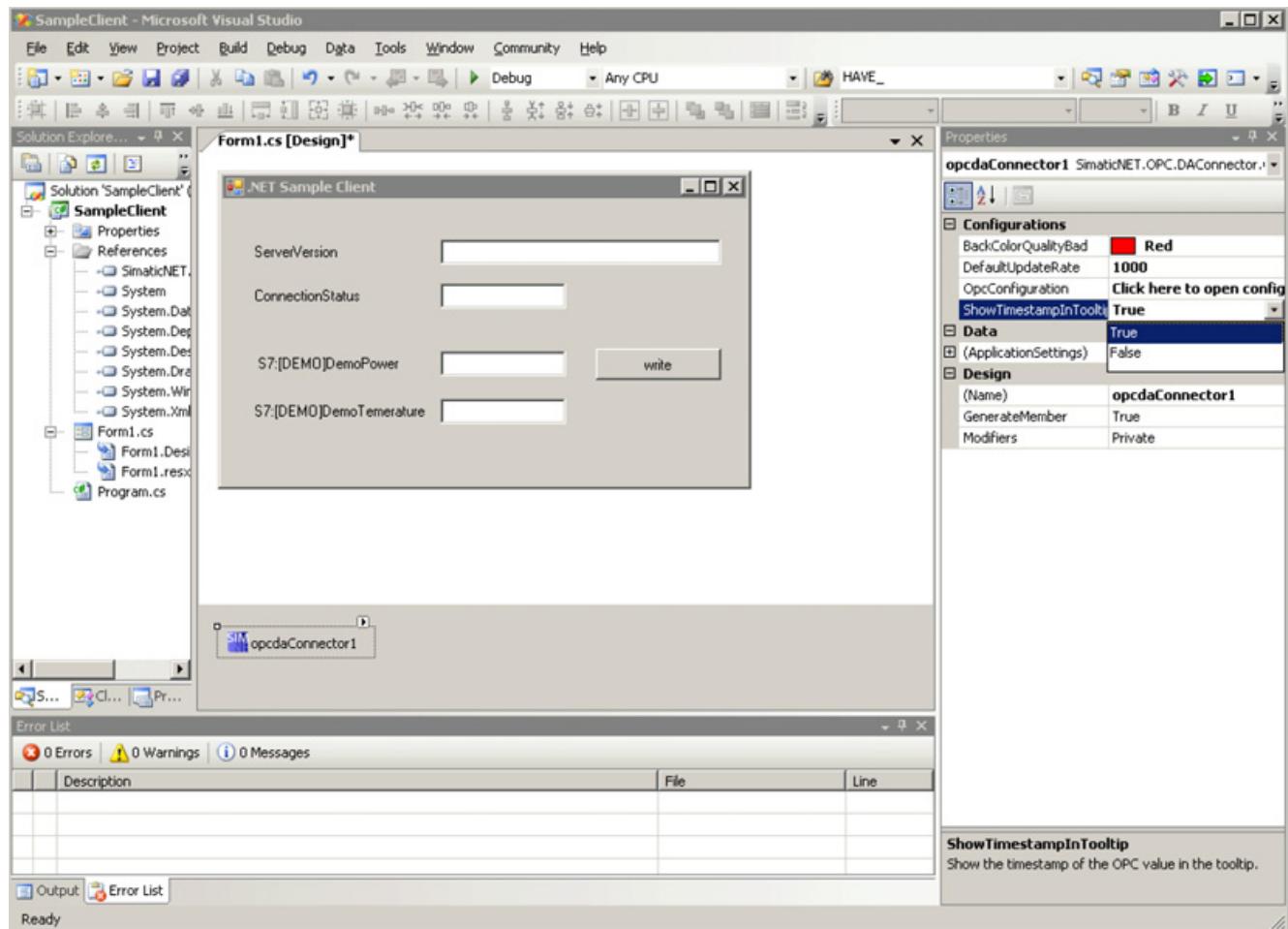
值只是简单地链接到“Windows

属性”控件中，与相应工具提示之间的互连是在后台建立的。

这些设置是固定的，无法对其进行修改。可以用单独的颜色显示每个错误。

## Timestamp

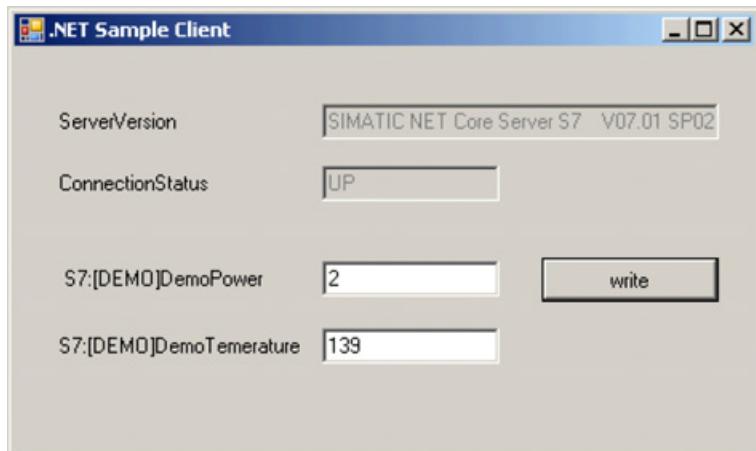
读取 OPC 数据项时，同时会返回 OPC 时间戳（UTC 时间）。OPC 属性不能与 Windows 控件属性单独互连。然而，如果启用了“ShowTimeStampInTooltip”，与相应的 Windows 对象的工具提示间的互连就会自动建立。



### 5.1.5 步骤 4 - 启动示例程序

示例应用程序通过双击“\*.exe”文件来启动。它位于 Visual Studio 示例项目下的“bin\Debug”或“bin\Release”子文件夹中，具体取决于编译时使用的组态。

要运行该示例，确保 S7 演示连接已激活。（请参见“设置示例程序 (页 646)”部分）。



#### 5.1.5.1 调试和发布

示例应用程序可编译为“调试”或“发布”。.NET 控件的汇编程序对这两种变型均适用。

#### 5.1.5.2 分发应用程序

新建的 OPC 客户端应用程序通常不会在开发计算机上运行，而是在安装了“SIMATIC NET PC 软件”的 SIMATIC PC（本地客户端）上运行，或是在另一台未安装 MS Visual Studio 和 SIMATIC NET 的计算机（远程 OPC 客户端）上运行。

在这种情况下，应该确保目标系统已安装了所有必要的文件。

##### 哪些文件需要随应用程序一起提供？

为了能够在任意计算机上运行该应用程序，除了可执行文件（“\*.exe”）和 .NET 汇编程序外，还需要 OPC proxies/stubs 和 VC 2010 SP1 Runtime。.NET 汇编程序需要 .NET Framework 4.0 或更高版本。OPC 核心组件 (proxy/stub) 必须与每个 OPC 产品（客户端或服务器）一起安装。这些文件已存在于安装了 SIMATIC NET 的任意计算机中。作为一种替代方法，合并模块或可再发行的安装程序也可以从 OPC 基金会 ([www.opcfoundation.org](http://www.opcfoundation.org)) 下载

### DLL 的位置以及它们需要安装的位置是什么？

编译示例应用程序时，“汇编程序”会自动复制到示例项目的 bin 目录中，它们必须随客户端应用程序一起安装（在与应用程序相同的目录中）。

- 对于 OPC 核心组件，还包括来自 OPC 基金会的合并模块。  
文件被安装到“%systemroot%\windows\system32\”中
- 对于 VC Runtime (2010 SP1)，还包括来自 Microsoft 的合并模块。

### 需要链接到安装程序的组件有哪些？

如果使用 Visual Studio 创建安装程序（部署项目），Studio 会自动识别其相互依赖性并在项目中插入汇编程序。除了相应的 C Runtime 2010 SP1，还需要链接 OPC 基金会的核心组件。

---

### 说明

或者，也可安装 OPC Scout V10 客户端安装程序。这已经包含了 OPC proxy/stub redistributable 和 .NET Framework。

---

## 5.1.6 故障排除

### Visual Studio LoaderLock 例外情况

如果从 Visual Studio 的调试程序启动示例应用程序，可能会看到来自 Studio 的警告（LoaderLock 例外情况）。导致此警告的原因是混合使用了管理和非管理代码。此警告可以忽略，因为 DataControl 是以正确的顺序启动的混合“汇编语言”，因此不会产生闭锁。尽管如此，我们还是建议您禁用该警告。要停止调试程序，请按以下步骤操作：

1. 选择 Studio 菜单中的“调试”(Debug) >“例外情况”(Exception)。
2. 打开“管理调试帮助”条目
3. 禁用“LoaderLock”复选框
4. 使用“确定”(OK) 退出对话框。

**无法建立至远程 OPC 服务器的连接。**

检查 DCOM 设置并遵循服务器用户文档中的说明。

记住，如果从开发环境启动应用程序，则应用程序将在 Visual Studio 环境 (DevEnv.exe) 中运行。这也意味着此过程所处的运行环境中 DCOM 权限必须设置正确。

**OPC 服务器不会报告任何数据更改。**

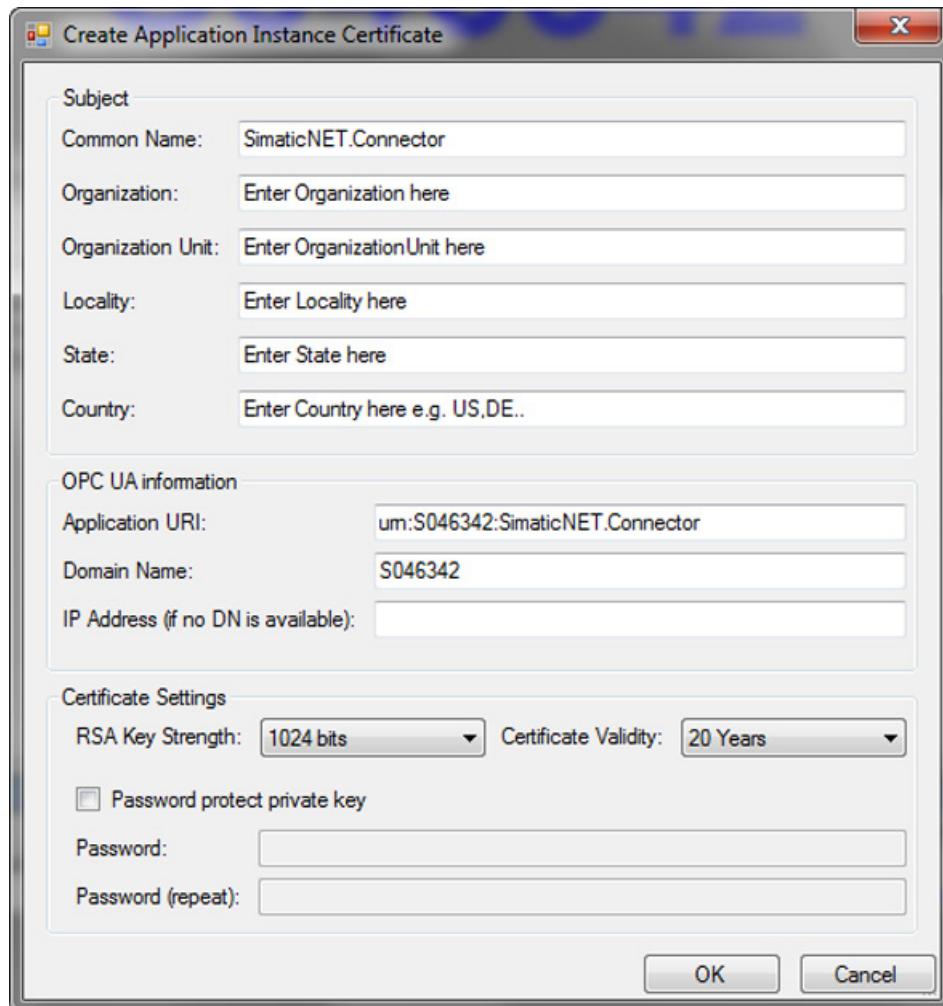
必须由应用程序进行这些安全设置，这样才能接收 OPC 服务器发送的消息（例如，“关闭事件”或“数据更改通知”）。

为了能够执行此操作，客户端应用程序需要设置正确的安全等级。这是在应用程序的初始化阶段完成的。

应用程序启动之前，必须使用合适的设置（AuthenticationLevel = RPC\_C\_AUTHN\_LEVEL\_CONNECT，模拟级别 = RPC\_C\_IMP\_LEVEL\_IDENTIFY）进行 CoInitializeSecurity 调用（有关详细信息，请参见 MSDN）。如果应用程序未设置 COM 安全性，则使用默认设置。随后必须对 DCOM 安全性进行相应的设置（“开始”>“运行”>“dcomcnfg.exe”）。

### 5.1.7 为.NET OPC 数据控件创建 OPC UA 证书

在对话框中，“.NET OPC 数据控件自签发 X.509 OPC UA 证书”在 Windows 证书存储库中生成。如果不存在合适的证书，则会创建它。



Windows 证书存储库中的证书管理与 UA S7 客户端 OPC Scout V10 中的证书管理相一致（请参见“OPC UA 客户端“OPC Scout V10”中的证书管理（页 592）”部分）。

### 5.1.8 连接至 OPC UA 服务器时的 OPC UA 证书

使用 .NET OPC 数据控件与“统一架构”(UA) 服务器建立连接通常可以使用不同的端点。可以组态 SIMATIC NET OPC UA 服务器，使一个端点允许未加密的连接，而另一个端点只接受加密的连接。

## 5.1 .NET OPC 客户端控件

连接到具有安全性的端点时，换句话说，端点的“SecurityPolicy”以及“MessageSecurityMode”不为“None”时，必须在客户端和服务器之间交换证书。

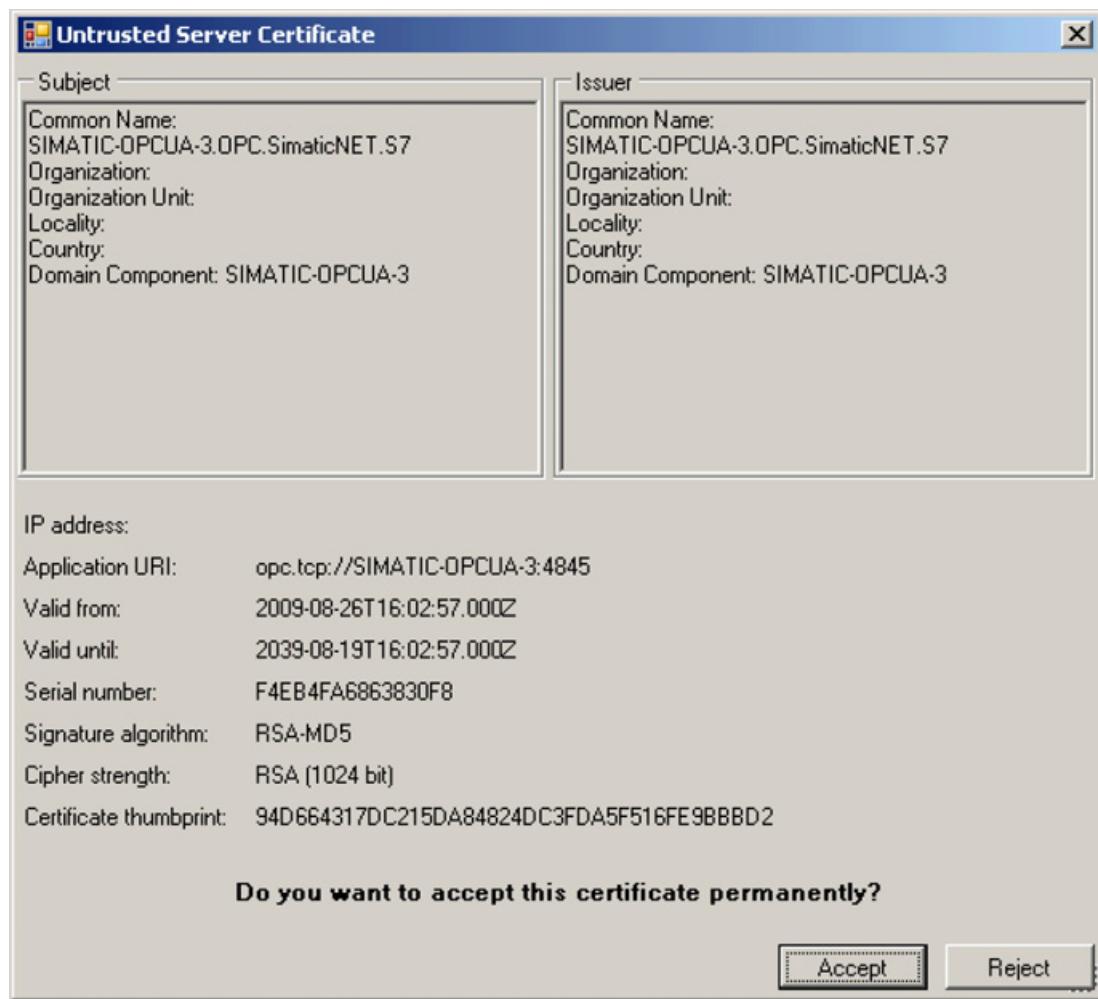
之后的连接建立需要遵循以下步骤：

- 客户端检查是否已经具有证书。如果没有，将提示用户创建一个。
- 客户端加载服务器的证书。

如果服务器证书尚不可知或尚未由 .NET OPC 数据控件 UA

客户端归类为可信证书，则系统会询问用户是否接受该证书。

这些证书位于“我的”存储库中，有效期限为 20 年。



当客户端尝试连接到服务器时，如果客户端证书未知或未由服务器归类为可信证书，服务器将拒绝该连接。

有关如何接受、拒绝和管理 SIMATIC NET OPC UA 服务器证书的方法，请参见“OPC UA 服务器的证书管理 (页 589)”和“使用图形用户界面进行证书管理 (页 597)”部分。

## 5.2 .NET OPC 客户端 API

### 目的

SIMATIC NET .NET OPC 客户端 API 的目的是，为 C-Sharp 和 Visual Basic .NET 语言的用户提供简单的优化类库，而该库不仅可直观使用，也可在快速创建用于访问 SIMATIC NET 产品系统的 OPC 客户端应用程序时进行使用。

### .NET OPC 客户端组件的特性

.NET 类库和基础 C++ 库具有以下特性：

- 简单而直观的 .NET 接口（API – 应用程序编程接口）。
- 减少基本功能的“OPC 数据访问”接口。
- 用户无需详细了解各种“OPC 数据访问”接口。
- 该组件将完全隐藏 OPC 的各种基本技术，例如 COM、DCOM、Web 服务、SOAP 和 XML。
- 如果出现错误，该组件将使用连接建立和重复的连接建立来完全隐藏对与 OPC 服务器之间的连接的处理。
- 使用 C-Sharp .NET 和 Visual Basic .NET 对 OPC 客户端应用程序进行的简单开发也适用于 SIMATIC NET .NET OPC 客户端 API。
- 从各种“OPC 数据访问”接口到 .NET 数据类型的 OPC 数据转换。
- 在本地和网络上快速而简单地发现 OPC COM 服务器。
- 通过实现 C++ 中的核心功能，进行客户端与服务器之间的高速而优化的通信。
- OPC UA 的支持及其证书管理。

---

### 说明

使用该 API，仅可建立与 SIMATIC NET 产品系列的 OPC 服务器之间的连接。  
无法与其它 OPC 服务器或其它供应商的 OPC 服务器进行连接。

---

---

### 说明

在连接到 OPC UA

服务器时，根据所选端点，可能必须先在客户端与服务器之间交换证书，然后才能进行连接。为此，使用相关的类别和方法。

---

## 5.2.1 SIMATICNET.OPCDAClient 命名空间

.NET 组件的命名空间 SimaticNet.OpcDaClient 通过“DaServerMgt”对象提供以下功能：

- 连接到 OPC 服务器

使用“Connect”方法，可以建立与 OPC

服务器的连接并可以使用“Disconnect”方法再次终止连接。该连接由 .NET 组件监控。  
如果连接出错，将使用“ServerStateChanged event”报告状态更改。

- 读取和写入“OPC 数据访问”项

使用“Read”和“Write”方法，可以同步、异步读取和写入 OPC 数据项值。

- 指示数据更改

.NET 组件提供一个机制来指示值更改，从而使循环读取变得不必要。

使用“Subscribe”方法，可将数据项注册以用于监控。

这些可以使用“SubscriptionCancel”再次取消。通过“DataChanged”  
事件”报告已更改的值。

- 获取有关地址空间的信息

“Browse”方法可用于在“OPC 数据访问”服务器的空间地址中搜索 OPC 数据项。

使用“GetProperties”方法，可以获取 OPC 数据项的属性。

### 5.2.1.1 数据模型的类别

以下部分描述了由 .NET API 提供的类别。

除了“DaServerMgt”类之外，这些类别只有属性，并因而用于数据封装。

“DaServerMgt”提供用于访问“OPC 数据访问”服务器的 API。

#### DaServerMgt 类

“DaServerMgt”类允许对“OPC 数据访问”服务器进行访问。可在“DaServerMgt”  
对象的接口”中找到有关 API 及其方法的详细描述。

## ServerState 枚举器

“ServerState”枚举器使用“StateChange 事件”进行传送，并提供有关服务器当前状态的信息。

- CONNECTED  
建立与 OPC 服务器之间的连接。
- DISCONNECTED  
不存在与 OPC 服务器的连接。
- ERRORSHUTDOWN  
OPC 服务器已发送“关闭”(Shutdown) 事件。连接已中断。尝试建立新连接。
- ERRORWATCHDOG  
.NET API 已检测到有关与 OPC 服务器的连接的错误。连接已中断。尝试建立新连接。
- UNDEFINED  
OPC 服务器所处的状态与列出的任何状态都不对应。

## ItemIdentifier 类

“ItemIdentifier”类用于“Read”、“Write”、“GetProperties”和“Subscribe”方法。  
该类用于标识 OPC 数据项。使用所有方法将该类的实例作为 `ref` 参数进行传送（输入/输出）。这是必不可少的，因为 .NET 组件在第一次使用对象时将信息存储在对象中，然后将其用于优化 OPC 调用。因此，这些对象将在 `Read` 和 `Write` 调用中被再次使用，且不会针对每个调用进行重新初始化。

该类中也包含错误信息（在调用之后需要检查错误信息，才能检测到数据项错误）。

`ItemIdentifier` 类的属性：

- **string ItemName**  
该属性包含“OPC 数据访问”项的数据项名称 (`ItemID`)。
- **string ItemPath**  
如果是 OPC XML DA 服务器，则可在此处指定可选的 `ItemPath`。如果是 COM“OPC 数据访问”服务器，则该属性将被忽略。
- **object ClientHandle**  
如果生成“Subscribe”方法的 `ItemIdentifier`，则使用“`DataChanged`”将具有数据项更改值的已传送的“`ClientHandle`”传回客户端。由于“`ClientHandle`”属于对象类型，因此可在此处传送任何 .NET

类型，例如文本框控件或包含有关数据继续处理的信息的对象。在“DataChange”句柄方法中，“ClientHandle”允许标识应用程序中的相应对象。

- **int ServerHandle**

使用所有方法将“ItemIdentifier”类的实例作为参考参数（关键字 `ref`）进行传送。

通过“Read”和“Write”方法调用，“int ServerHandle”属性由 .NET API 设置。

如果相同的 ItemIdentifier 实例再次使用“Read”和“Write”来传送，.NET API 可以极大地优化 OPC 服务器调用。

- **System.Type DataType**

可以在此处指定 OPC 服务器使用“Read”或“DataChange”返回数据项值的数据类型。

如果未设置该属性，将返回 OPC 服务器上数据项的本地数据类型。

这种情况下，.NET API 将在第一次使用 ItemIdentifier 实例时设置数据类型。

- **ResultID ResultID**

如果在 OPC 调用期间出现数据项错误（例如，未知的

ItemName、尝试写入只读数据项，等等），相应的错误代码将存储在相应“ItemIdentifier”的对象中。“ResultID”类将提供错误代码

(int)、名称（字符串）以及本地化说明（字符串）。

这样一来，程序便可对错误做出响应。

## ItemValue 类

“ItemValue”类用于“Read”和“Write”方法。

读取时，“ItemValue”将包含值、质量和时间戳。写入时，“ItemValue”仅包含 OPC 数据项的值。

调用“Read”方法时，“ItemValue 数组”是一个输出参数，这表示对象完全由 .NET API 生成。

调用“Write”方法时，“ItemValue 数组”必须由客户端根据“ItemIdentifier 数组”生成，并用要写入的值进行填充。

“ItemValue”类的属性：

### **object Value**

已读取的值或将写入的值。

由于“值”属性属于“对象”类型，因此它可采用或包含任何数据类型。

通常，“值”的类型与使用相应“ItemIdentifier”请求的类型相同。

### **QualityID Quality**

值的质量。“QualityID”类将提供质量代码 (int)、名称（字符串）以及说明（字符串）。

**System.DateTime TimeStamp**

值的时间戳。

**ItemValueCallback 类**

“ItemValueCallback”类源自“ItemValue”类，并具有“ItemValue”中所描述的属性以及以下其它属性：

- **object ClientHandle**

在此，将返回随“Subscribe”或“ReadAsync”方法提供的“ClientHandle”。  
“ClientHandle”允许客户端唯一标识返回值。

- **ResultID ResultID**

在此输入任何与“ClientHandle”所标识的数据项相关的错误。  
“ResultID”类将提供错误代码 (int)、名称 (字符串) 以及本地化说明 (字符串)。  
这样一来，程序便可对错误做出响应。

**ItemResultCallback 类**

“ItemResultCallback”类用于“WriteCompleted Callback”并具有以下属性：

- **object ClientHandle**

在此，将返回随“WriteAsync”方法提供的“ClientHandle”。  
“ClientHandle”允许客户端唯一标识返回值。

- **ResultID ResultID**

在此输入任何与“ClientHandle”所标识的数据项相关的错误。  
“ResultID”类将提供错误代码 (int)、名称 (字符串) 以及本地化说明 (字符串)。  
这样一来，程序便可对错误做出响应。

**BrowseElement 类**

“BrowseElement”类包含与使用“Browse”方法获取的 OPC 数据访问项相关的所有数据。

“BrowseElement”类的属性：

**string Name**

返回元素的名称。通常，该名称用于 OPC 服务器命名空间的树形表示法。

**string ItemName**

元素的 ItemName。

**string ItemPath**

元素的 ItemPath。

**bool HasChildren**

指示元素是否在命名空间中具有子元素。

**bool IsItem**

指示该元素是否是“OPC 数据访问”数据项。

**ItemProperties ItemProperties**

使用“Browse”方法请求的元素的属性。

**BrowseFilter 枚举器**

调用“Browse”方法时，传递“BrowseFilter”将指定要返回的指定元素的子元素类型。

可能的过滤器包括：

- ALL

返回所有元素。

- BRANCH

仅返回分支类型的元素。

- ITEM

仅返回数据项类型的元素。

**ItemProperties 类**

“ItemProperties”类仅由 .NET API 生成。它包含 OPC 数据项的属性。

“ItemProperties”类的属性：

- **ItemProperty[ ] RequestedItemProperties**

“ItemProperty”类的对象数组。该数组包含使用“GetProperties”或“Browse”请求的 OPC 数据项的所有属性。

**ItemProperty 类**

“ItemProperty”类代表 OPC 数据项的属性。

“ItemProperty”类的属性：

- **string ItemName**  
如果 OPC  
服务器支持通过数据项读取和写入属性，属性的“ItemName”将在此处被返回。
- **string ItemPath**  
如果 OPC  
服务器支持通过数据项读取和写入属性，属性的“ItemPath”将在此处被返回。
- **string Description**  
属性的描述。此信息可用于在图形用户界面中表示属性。
- **object Value**  
属性的值。
- **ResultID ResultID**  
如果在获取属性时出错，相应的错误代码将存储在属性中。
- **System.Type DataType**  
属性的数据类型。
- **int PropertyID**  
属性的 ID。

## ResultID 类

“ResultID”类将包含错误代码、其字符串表示法以及已出现的错误的本地化说明。

“ResultID”用于显示数据项级别的错误，并且是 OPCException 的一部分。

“ResultID”类的属性：

- **int Code**  
在进行操作时由服务器传送的代码。
- **string Name**
- 代码的字符串表示法。
- **string Description**  
已出现的错误的本地化说明。
- **bool Succeeded**  
利用该属性，可以查明“ResultID”是否包含成功操作的代码，而不需要详细评估代码本身。

“ResultID”类的实例可具有以下值：

值	代码	名称	说明
WIN_S_OK	0x00000000	S_OK	操作成功。
WIN_S_FALSE	0x00000001	S_FALSE	该函数部分成功。
S_UNSUPPORTEDRATE	0x0004000D	OPC_S_UNSUPPORTEDRATE	服务器不支持请求的数据速率，但会使用最接近的可用速率。
S_INUSE	0x0004000F	OPC_S_INUSE	由于正在引用该对象，因此操作无法执行。

值	代码	名称	说明
S_DATAQUEUEOVERFLOW	0x00040404	OPC_S_DATAQUEUEOVERFLOW	由于服务器的缓冲区已达到极限且必须清除最早的数据，所以不是每个已检测到的更改都会被返回。
S_CLAMP	0x0004000E	OPC_S_CLAMP	已接受传递至写入的值，但会固定输出。
RPC_S_SERVER_UNAVAILABLE	0x800706BA	RPC_S_SERVER_UNAVAILABLE	RPC 服务器不可用。
RPC_S_CALL_FAILED	0x800706BE	RPC_S_CALL_FAILED	远程过程调用失败。
E_UNKNOWNPATH	0xC004000A	OPC_E_UNKNOWNPATH	数据项的访问路径对服务器未知。

值	代码	名称	说明
E_UNKNOWNITEMID	0xC0040007	OPC_E_UNKNOWNITEMID	数据项 ID 不会在服务器地址空间中进行定义，或者不再存在于服务器地址空间中。
E_RATENOTSET	0xC0040405	OPC_E_RATENOTSET	没有为指定数据项设置任何采样速率。
E_RANGE	0xC004000B	OPC_E_RANGE	该值超出范围。
E_PUBLIC	0xC0040005	OPC_E_PUBLIC	不能在公共组上执行请求的操作。
E_NOTSUPPORTED	0xC0040406	OPC_E_NOTSUPPORTED	服务器不支持质量和/或时间戳的写入。

值	代码	名称	说明
E_NOTFOUND	0xC0040011	OPC_E_NOTFOUND	未找到请求的对象（例如公共组）。
E_NOBUFFERING	0xC0040402	OPC_E_NOBUFFERING	服务器不支持数据项的缓冲，而这些数据项是以快于组更新的速率进行收集。
E_INVALIDITEMID	0xC0040008	OPC_E_INVALIDITEMID	该数据项 ID 不符合服务器的语法。
E_INVALIDHANDLE	0xC0040001	OPC_E_INVALIDHANDLE	句柄的值无效。
E_INVALIDFILTER	0xC0040009	OPC_E_INVALIDFILTER	该过滤器字符串无效。
E_INVALIDCONTINUATIONPOINT	0xC0040403	OPC_E_INVALIDCONTINUATIONPOINT	该连续点无效。

值	代码	名称	说明
E_INVALIDCONFIGFILE	0xC0040010	OPC_E_INVALIDCONFIGFILE	服务器的组态文件格式无效。
E_INVALIDARG	0x80070057	E_INVALIDARG	该函数的一个参数无效。
E_INVALID_PID	0xC0040203	OPC_E_INVALID_PID	数据项的指定属性 ID 无效。
E_FAIL	0x80004005	E_FAIL	未指定的错误。
E_DUPLICATENAME	0xC004000C	OPC_E_DUPLICATENAME	不允许重复的名称。
E_DEADBANDNOTSUPPORTED	0xC0040401	OPC_E_DEADBANDNOTSUPPORTED	该数据项不支持死区。
E_DEADBANDNOTSET	0xC0040400	OPC_E_DEADBANDNOTSET	尚未设置该数据项的数据项死区。

值	代码	名称	说明
E_BADTYPE	0xC0040004	OPC_E_BADTYPE	该服务器不能转换指定格式和/或请求的数据类型与标准数据类型之间的数据。
E_BADRIGHTS	0xC0040006	OPC_E_BADRIGHTS	数据项的访问权限不允许该操作。
DISP_E_TYPEMISMATCH	0x80020005	DISP_E_TYPEMISMATCH	类型不匹配。
CONNECT_E_NOCONNECTION	0x80040200	CONNECT_E_NOCONNECTION	客户端未通过 IConnectionPoint::Advise 注册回调。
CONNECT_E_ADVISELIMIT	0x80040201	CONNECT_E_ADVISELIMIT	建议超过该对象的限制。

## QualityID 类

“QualityID”类包含与服务器所传送的质量代码相关的所有信息。

“QualityID”类的属性:

- **int FullCode**  
由服务器传送的代码。
- **int Quality**  
描述传送值的质量的代码。
- **int LimitBits**  
由服务器传送的代码的限制部分。
- **int VendorBits**  
由服务器传送的代码的供应商部分。
- **bool IsGood**  
使用该属性，可以了解读取值的质量是否良好。
- **string Name**  
代码的字符串表示法。
- **string Description**  
质量代码的本地化说明。

## ConnectInfo 类

调用“Connect”方法时，“ConnectInfo”类包含由服务器对象评估的初始化数据。

“ConnectInfo”类的属性:

- **bool RetryAfterConnectionError**  
如果设置此标记，OPC 客户端 API  
将尝试在连接中止后重新建立连接，直到建立成功。  
如果该连接可以重新建立，则会在中断保持有效之前生成组句柄。  
“Eventhandler”方法保持注册以用于这些事件。
- **bool RetryInitialConnect**  
如果设置此标记，则 OPC 客户端 API  
会在第一次连接尝试失败时继续尝试连接到服务器。

- **int KeepAliveTime**

OPC 客户端 API 将检查可用性以及运行期间的服务器连接情况。KeepAliveTime 表示进行验证之前的时间间隔（以毫秒为单位）。

“KeepAliveTime”的初始值为 10000 ms。

重新连接至 OPC 服务器的时间间隔以两倍于“KeepAliveTime”的时间开始，并且当 OPC 服务器较长时间不可用时，最多可增加至“KeepAliveTime”的 10 倍。OPC 服务器关闭后重新连接的时间间隔是一分钟。

- **string LocalID**

LocalID 可用于向服务器发送国家/地区标识符（“us”、“en”等）。设置 LocalID 能够实现以特定语言传送本地化返回值。

如果本地化版本中不存在该值，则将传送默认值。下表显示了 LocalID 的几个示例：

区域设置	LocalID
系统默认	空字符串
英语	en
英语 - 美国	en - us
英语 - 英国	en - gb
德语	de
德语 - 德国	de - de
德语 - 奥地利	de - at

---

### 说明

连接状态的更改应始终使用“ServerStateChanged”事件进行跟踪。

只有在服务器的状态已更改为已连接后，诸如“Browse”、“Read”或“Subscribe”之类的其它方法才能成功执行。

---

- **String SecurityPolicyUri**

SecurityPolicy 定义用于通过 UA SecureChannel

传送的安全的类型，例如未加密或已加密的 Basic128Rsa。

- **byte MessageSecurityMode**

MessageSecurityMode 定义将在消息级别使用的安全性，例如 1 = None、2 = Encrypt、3 = Encrypt&Sign。

- **byte[] ServerCertificate**  
将要连接到的 OPC UA 服务器的 X509 证书。  
例如，该证书可使用“OpcServerEnum::getCertificateForEndpoint”进行加载。
- **byte[] ClientCertificate**  
客户端的 X509 证书。
- **byte[] ClientPrivateKey**  
客户端的 X509 证书的私钥。
- **string CertificateStoreName**  
存储客户端和服务器证书的“WindowsCertificateStore”的名称。 使用“UA 应用程序”。
- **WinStoreLocation CertificateStoreLocation**  
“WindowsCertificateStore”中的位置，例如 LocalMachine

---

#### 说明

仅当与 OPC UA

服务器之间的连接非常安全时，才会需要“WindowsCertificateStore”的证书和设置，换言之，“SecurityPolicy”和“MessageSecurityMode”未设置为“None”。

要使用位置“LocalMachine”，需要管理员权限。

---

## ReturnValue 枚举器

返回“ReturnValue”将显示：调用各种方法时，相关函数是否成功或者值的质量是否不好。

- **SUCCEEDED**  
该函数成功完成。
- **ITEMERROR**  
运行期间出错，并且具有至少一个数据项。  
必须利用函数参数获取所涉及的数据项以及确切的错误代码的用途（例如 ref ItemIdentifier[]）。
- **QUALITYNOTGOOD**  
至少一个数据项的质量不好。  
必须利用函数参数获取所涉及的数据项以及确切的质量代码的用途（例如 ref ItemIdentifier[]）。

- ITEMERRORANDQUALITYBAD

运行期间出错（具有至少一个数据项），并且至少一个数据项的质量不好（相同或不同的数据项）。必须利用函数参数获取所涉及的数据项 (ref ItemIdentifier[])。

- UNSUPPORTEDUPDATERATE

服务器不支持所请求的更新速率。

### 5.2.1.2 DaServerMgt 对象的接口

#### 生成 DaServerMgt 对象

由于 .NET 类型很典型，因此必须首先生成“DaServerMgt”类的实例：

```
DaServerMgt daServerMgt = new DaServerMgt();
```

#### Connect 方法

“Connect”方法将建立与服务器的连接。

“**IsConnected**”属性指示客户端与 .NET API 之间的连接状态。当调用“**connect**”时，.NET API 将检查与 OPC 服务器的连接。如果与服务器的连接中止，.NET API 将尝试重新建立连接。“**ServerState**”属性显示 OPC 服务器的状态。可能的值是 DISCONNECTED、CONNECTED、ERRORSHUTDOWN、ERRORWATCHDOG。如果状态设置为两种错误状态之一，.NET API 将尝试重新建立连接。通过“**ServerStateChanged**”事件将状态更改告知客户端。

参数	功能
<b>string url</b>	OPC 服务器的 URL（参见下文：“URL 的结构”）。
<b>ConnectInfo</b>	“ <b>ConnectInfo</b> ”包含用于初始化 .NET API 的多个属性。 <b>LocalID</b> 字符串：要用于语言特定的返回值的国家/地区设置。如果传送空字符串，将使用默认设置。“ <b>bool ReconnectAfterConnectionError</b> ”/“ <b>bool RetryInitialConnect</b> ”。由于网络连接暂时不可用，则与 OPC 服务器建立连接时失败。通过设置这两个参数，.NET API 将在出现错误时尝试重新建立连接。 始终设置“ <b>IsConnected</b> ”属性，然而客户端仍可创建订阅而不管连接错误如何。 但是，对于订阅中的数据项，不良质量会在“ <b>DataChanged</b> ”中返回，直到可以建立与服务器的连接。 如果连接建立失败且未设置此参数，“ <b>Connect</b> ”方法将失败并报告异常。 “ <b>int keepAliveTime</b> ”是检查与 OPC 服务器之间的连接时所采用的时间间隔（以毫秒为单位）。
<b>int clientHandle</b>	利用此参数，应用程序可以为该服务器对象分配索引。 当出现“ <b>ServerStateChanged</b> ”事件来标识应用程序中的对象时，该“ <b>ClientHandle</b> ”由 .NET API 传递。
<b>out bool connectFailed</b>	指示在已设置“ <b>retryConnect</b> ”参数后连接建立是否失败。 如果连接建立失败且未设置“ <b>retryConnect</b> ”参数，“ <b>connect</b> ”方法将失败并报告异常。

### 所需的 URL

格式可由用户输入，也可从“**SimaticNet.OpcCmn**”命名空间中的“**OpcServerEnum**”类自动获取。

```
string url = "opcda://localhost/OPC.SimaticNET/{b6eacb30-42d5-11d0-9517-0020afaa4b3c}  
";
```

```
string localId = "de";  
  
int clientHandle = 0;  
  
int keepAliveTime = 5000; // 5 seconds  
  
daServerMgt.Connect(url, localId, clientHandle, true, keepAliveTime, out  
connectFailed);
```

### 说明

使用该 API，仅可建立与 SIMATIC NET 产品系列的 OPC 服务器之间的连接。  
无法与其它 OPC 服务器或其它供应商的 OPC 服务器进行连接。

### 说明

如果此 API 用于多线程应用程序，请确保执行 **Connect**  
方法的线程是在与应用程序本身相同的单元状态中执行。

#### 示例：

如果使用“**ApartmentState STA**”开发应用程序，并且 **Connect**  
需要在与基础线程不同的线程中执行，则在开始之前请按如下方式设置新线程的“**ApartmentState**”：

```
Thread MyThread = new Thread(MyThreadStart);  
MyThread.SetApartmentState(ApartmentState.STA);  
“Apartment State MTA”的步骤类似。
```

## URL 的结构

唯一标识服务器的 URL 的结构如下：

- 对于 OPC COM 数据访问

[OpcSpecification]:// [Hostname] / [ServerIdentifier]

- 对于 OPC 统一架构

[Endpoint server URL]

URL 部分	说明
OpcSpecification	指定要使用的 OPC DA 规范 <ul style="list-style-type: none"> <li>• OPC 数据访问 2.05A 或 3.0 (COM) 的 opcda</li> <li>• OPC XML-DA 1.01 的 HTTP</li> </ul>
Hostname	运行 OPC 服务器的计算机的计算机名称或 IP 地址。 对于本地计算机，它是 localhost。对于 OPC-XML-DA 服务器，端口也可以和 IP 地址一起进行指定。
ServerIdentifier	标识指定计算机上的 OPC 服务器。 <ul style="list-style-type: none"> <li>• OPC XML-DA - Web 服务的路径</li> <li>• OPC COM DA – [ProgID]/[optional ClassID]</li> </ul>
Endpoint server URL	opc.tcp://[hostname]:[port] <ul style="list-style-type: none"> <li>• opct.tcp 是 OPC UA 二进制 TCP 协议标识符</li> <li>• hostname 是计算机名称或 IP 地址</li> <li>• port 是端口号</li> </ul>

下面列出正确 URL 的示例：

opcda://PC\_001/OPC.SimaticNET/{b6eacb30-42d5-11d0-9517-0020afaa4b3c}  
 opcda://localhost/OPC.SimaticNet.DP/{625c49a1-be1c-45d7-9a8a-14bedcf5ce6c}/  
 http://192.168.0.120/Opc.Simatic.Net/WebService.asmx  
 opc.tcp://PC1:55101 (对于 OPC.SimaticNET.S7)

## Disconnect 方法

调用该方法将终止与 OPC 服务器的连接。全部现有订阅和资源都将释放。

## IsConnected 属性

利用该属性，可以查询客户端 API 的当前连接状态。使用“ServerState”属性可以查询与 OPC 服务器之间的连接的状态。

```

if(daServerMgt.IsConnected == true)
{
}
  
```

```

daServerMgt.Disconnect();

}

```

## ServerState 属性

利用该属性，可以查询与 OPC 服务器之间的连接的当前状态。

## Read 方法

使用“Read”方法，可以读取 OPC 服务器的数据项。

如果循环读取数据项，则最好创建订阅并使用“DataChanged”事件接收已更改的数据。

参数	功能
<b>int maxAge</b>	要读取的值的最大老化时间（以毫秒为单位）。如果 OPC 服务器缓存中的当前值比 <b>maxAge</b> 指定的值早，则将从设备中读取值。如果传送值 0，则始终从设备中读取值。
<b>ref ItemIdentifier[] itemIdentifiers</b>	“itemIdentifiers”数组指定要读取的 OPC 数据项。任何数据项错误都会在相关“ItemIdentifiers”的“ResultID”对象中进行返回。参数为输入/输出参数，以便 .NET API 能够在对象中返回错误信息和“ServerHandle”。如果“Read”和“Write”调用的相同数据项被使用多次，则仅应创建一次对象，然后将其用于所有调用。这样一来，.NET API 便可使用保存在对象中的信息来优化 OPC 服务器访问。
<b>out ItemValue[] itemValues</b>	“itemValues”数组包含用于读取 OPC 数据项的值、质量和时间戳。

返回类型	功能
<b>ReturnCode</b>	返回代码将指示所出现的错误是否与传送的参数相关，以及所有读取值的质量是否都非常好。 应基于枚举器的值检查“ItemIdentifier[]”。

下列代码是有关如何使用一个“Read”读取两个数据项的示例：

```

ReturnCode result;

int maxAge = 1000;

```

```
ItemIdentifier[] itemIdentifiers = new ItemIdentifier[2];

itemIdentifiers[0] = new ItemIdentifier();

itemIdentifiers[0].ItemName = "S7:[@LOCALSERVER]DB1,B0";

itemIdentifiers[1] = new ItemIdentifier();

itemIdentifiers[1].ItemName = "S7:[@LOCALSERVER]DB1,B1";

ItemValue[] itemValues = null;

result = daServerMgt.Read(maxAge,

ref itemIdentifiers,

out itemValues);

if (result != ReturnCode.SUCCEEDED)

{

// ToDo: implement error handling

}

else

{

if (result != ReturnCode.ITEMERRORANDQUALITYBAD)

{

// all values are correct and can be used

// ToDo: use the values in the application

string strValue1;

string strValue2;

strValue1 = itemValues[0].Value.ToString();

strValue2 = itemValues[1].Value.ToString();

}

else

{
```

```
// ToDo: check the ItemIdentifiers[]

}

}
```

## 说明

### .NET API

的所有方法都允许组调用，换言之，使用一个“Read”调用可以读取任意数量的数据项。

在可能的情况下，应该使用组调用，因为该策略一方面可以将客户端与服务器之间的通信减少至最小，另一方面可以充分利用 API 的优化机制。

## ReadAsync 方法

使用“ReadAsync”方法，可以异步读取 OPC 服务器的数据项。

使用“ReadCompleted”事件发送读取值。

如果循环读取数据项，则最好创建订阅并使用“DataChanged”事件接收已更改的数据。

参数	功能
<b>int transactionHandle</b>	句柄作为“Read”事务的标识。 利用“ReadCompleted”事件，句柄将返回至客户端。
<b>int maxAge</b>	要读取的值的最大老化时间（以毫秒为单位）。如果 OPC 服务器缓存中的当前值比“maxAge”指定的值早，则将从设备中读取值。如果传送值 0，则始终从设备中读取值。
<b>ref ItemIdentifier[] itemIdentifiers</b>	“itemIdentifiers”数组指定要读取的 OPC 数据项。 任何数据项错误都会在相关“ItemIdentifiers”的“ResultID”对象中进行返回。参数为输入/输出参数，以便 .NET API 能够在对象中返回错误信息和“ServerHandle”。 如果“Read”和“Write”调用的相同数据项被使用多次，则仅应创建一次对象，然后将其用于所有调用。这样一来，.NET API 便可使用保存在对象中的信息来优化 OPC 服务器访问。

返回类型	功能
<b>ReturnCode</b>	返回代码将指示所出现的错误是否与传送的参数相关，以及所有读取值的质量是否都非常好。 应基于枚举器的值检查“ItemIdentifier[]”。

### ReadCompleted 事件

使用“ReadCompleted”事件，.NET API 会将“ReadAsync”调用的结果发送至应用程序。

参数	功能
<b>int transactionHandle</b>	句柄由应用程序使用“ReadAsync”传送。
<b>bool allQualitiesGood</b>	指示“ItemValueCallback”数组中没有质量不良的值。
<b>bool noErrors</b>	指示“ItemValueCallback”数组中没有任何错误。
<b>ItemValueCallback[] itemValues</b>	“ReadAsync”调用中具有每个数据项的值、质量和时间戳的数据组。 使用“ValueCallback”数据项中的“ClientHandle”进行分配。 如果“noErrors”的值为“false”，还必须检查“ItemValueCallback”的“ResultID”属性。

要为 OPC 服务器的“DataChanged”事件注册，必须为此事件注册“Handler”方法。

下列代码显示了关于如何为“DataChanged”事件注册“Handler”方法的示例：

```
daServerMgt.ReadCompleted += new DaServerMgt.ReadCompletedEventHandler
(MyReadCompleted);
```

```
void MyReadCompleted(int transactionHandle,
                     bool allQualitiesGood,
                     bool noErrors,
                     ItemValueCallback[] itemValues)
{
    // ToDo: Error check
    // ToDo: Entry of the values in the application
}
```

```
}
```

## Write 方法

参数	功能
<b>ref ItemIdentifier[] itemIdentifiers</b>	“itemIdentifiers”数组指定要写入的 OPC 数据项。任何数据项错误都会在相关“ItemIdentifiers”的“ResultID”对象中进行返回。参数为输入/输出参数，以便 .NET API 能够在对象中返回错误信息和“ServerHandle”。如果“Read”和“Write”调用的相同数据项被使用多次，则仅应创建一次对象，然后将其用于所有调用。这样一来，.NET API 便可使用保存在对象中的信息来优化 OPC 服务器访问。
<b>ItemValue[] itemValues</b>	“itemValues”数组包含要写入的值。

返回类型	功能
<b>ReturnCode</b>	返回值指示是否发生了涉及传递参数的错误。应基于枚举器的值检查“ItemIdentifier[]”。

下列代码是一个显示如何将整数值 11 或 22 写入单个数据项的示例：

```
ItemIdentifier[] itemIdentifiers = new ItemIdentifier[2];

itemIdentifiers[0] = new ItemIdentifier();

itemIdentifiers[0].ItemName = "S7:[@LOCALSERVER]DB1,B0";

itemIdentifiers[1] = new ItemIdentifier();

itemIdentifiers[1].ItemName = "S7:[@LOCALSERVER]DB1,B1";

ItemValue[] itemValues = new ItemValue[2];

itemValues[0] = new ItemValue();

itemValues[0].Value = 11;

itemValues[0] = new ItemValue();

itemValues[0].Value = 22;
```

```
daServerMgt.Write(ref itemIdentifiers, itemValues);
```

### 说明

#### .NET API

的所有方法都允许组调用，换言之，使用一个“Write”调用可以写入任意数量的数据项。

在可能的情况下，应该使用组调用，因为该策略一方面可以将客户端与服务器之间的通信减少至最小，另一方面可以充分利用 API 的优化机制。

## WriteAsync 方法

使用此方法，可将值异步写入 OPC 服务器。

由“WriteCompleted”事件返回有关值是否成功写入的信息。

参数	功能
<b>int transactionHandle</b>	句柄作为“Write”事务的标识。 利用“WriteCompleted”事件，句柄将返回至客户端。
<b>ref ItemIdentifier[] itemIdentifiers</b>	“itemIdentifiers”数组指定要写入的 OPC 数据项。 任何数据项错误都会在相关“ItemIdentifiers”的 ResultID 对象中进行返回。参数为输入/输出参数，以便 .NET API 能够在对象中返回错误信息和“ServerHandle”。 如果“Read”和“Write”调用的相同数据项被使用多次，则仅应创建一次对象，然后将其用于所有调用。这样一来，.NET API 便可使用保存在对象中的信息来优化 OPC 服务器访问。
<b>ItemValue[] itemValues</b>	“itemValues”数组包含要写入的值。

返回类型	功能
<b>ReturnCode</b>	返回值指示是否发生了涉及传递参数的错误。应基于枚举器的值检查“ItemIdentifier[]”。

## WriteCompleted 事件

使用“WriteCompleted”事件，.NET API 会将“WriteAsync”调用的结果发送至应用程序。

参数	功能
<b>int transactionHandle</b>	句柄由应用程序使用“WriteAsync”传送。
<b>bool noErrors</b>	指示“ItemResultCallback”数组中没有任何错误。
<b>ItemResultCallback[] itemResults</b>	“WriteAsync”调用中具有每个数据项的“ClientHandle”和错误代码的数组。 使用“ItemResultCallback”中的“ClientHandle”进行分配。 如果“noErrors”的值为“false”，还必须检查“ItemResultCallback”的“ResultID”属性。

要为 OPC 服务器的“WriteCompleted”事件注册，必须为此事件注册“Handler”方法。

下列代码显示了关于如何为“WriteCompleted”事件注册“Handler”方法的示例：

```
daServerMgt.WriteCompleted += new DaServerMgt.WriteCompletedEventHandler
    (MyWriteCompleted);

void MyWriteCompleted(int transactionHandle,
    bool noErrors,
    ItemResultCallback[] itemResults)
{
    // ToDo: Error check
}
```

## Browse 方法

**Browse** 方法可以用来浏览 OPC 服务器的命名空间。

命名空间通常显示为树形结构，因为此表示法与服务器的数据项和文件夹结构相对应。

参数	功能
<b>string itemName</b>	“itemName”参数指定将会为其找到所有子元素的元素（目录）。 如果要在根级别浏览服务器的命名空间，请传送空字符串。
<b>string itemPath</b>	利用此参数，可以指定 OPC XML DA 服务器元素的“ItemPath”。
<b>ref string continuationPoint</b>	如果由客户端返回的元素数目 (maxElementsReturned 参数) 或由服务器返回的元素数目被限制为特定数量，服务器将使用“continuationPoint”参数返回连续点以对同一元素执行进一步的“Browse”调用。如果 OPC 服务器返回“ContinuationPoint”，则必须使用相同的参数和返回的“ContinuationPoint”来重复调用“Browse”以找到其余的子元素。
<b>int maxElementsReturned</b>	在此，可以指定要在调用该函数时返回的元素的最大数。如果传送值为 0，将返回所有元素。
<b>BrowseFilter browseFilter</b>	在此，必须定义“BrowseFilter”来描述要返回的元素的类型。（“All”、“Items”或“Branch”）
<b>int[] propertyIDs</b>	在此，可传送要在调用“Browse”方法时找到的属性的 ID。系统会在相关的“BrowseElement”中返回属性。
<b>bool returnAllProperties</b>	如果设置此标记，则将自动获取一个数据项的所有属性。系统会在相关的“BrowseElement”中返回属性。
<b>bool returnPropertyValues</b>	如果设置此标记，还会返回请求的属性的值。
<b>out BrowseElement[] browseElements</b>	该数组包含地址空间中传送的元素的子元素。
<b>out bool moreElements</b>	“moreElements”指示是否返回所有子元素。

返回类型	功能
<b>ReturnCode</b>	返回值指示是否发生了涉及传递参数的错误。 应基于枚举器的值检查“BrowseElement[]”。

下列代码显示一个有关如何在没有相应属性的情况下于根级别找到所有数据项和文件夹的示例：

```
BrowseElement[] browseElements = null;

bool moreElements = false;

bool returnAllProperties = false;

bool returnPropertyValues = false;

string continuationPoint = null;

daServerMgt.Browse(string.Empty,
                    string.Empty,
                    ref continuationPoint,
                    0,
                    BrowseFilter.ALL,
                    null,
                    returnAllProperties,
                    returnPropertyValues,
                    out browseElements,
                    out moreElements);
```

## GetProperties 方法

使用“GetProperties”方法，可以获取 OPC 数据项的属性。

参数	功能
<b>ref ItemIdentifier[] itemIdentifiers</b>	“itemIdentifiers”参数描述将要为其找到属性的 OPC 数据项。
<b>int[] propertyIDs</b>	在此，可传送要在调用“GetProperties”方法时找到的属性的 ID。系统会在相应的“ItemProperties”元素中返回属性。
<b>bool returnAllProperties</b>	如果设置此标记，则将自动获取一个数据项的所有属性。系统会在相应的“ItemProperties”元素中返回属性。
<b>out ItemProperties[] itemProperties</b>	该数组包含请求的“ItemProperties”对象。“ItemProperties”对象中的属性与具有相同数组索引的相应“ItemIdentifier”对象相关。

返回类型	功能
<b>ReturnCode</b>	返回值指示是否发生了涉及传递参数的错误。 应基于枚举器的值检查“ItemIdentifier[]”。

下列代码显示一个有关如何获取“OpcItem1”数据项的属性“AccessRights”和“DataType”的示例：

```
ItemProperties[] itemProperties = null;

ItemIdentifier[] itemIdentifiers = new ItemIdentifier[1];

itemIdentifiers[0] = new ItemIdentifier();

itemIdentifiers[0].ItemName = "S7:[@LOCALSERVER]DB1,B0";

Int32[] propertyIDs = new Int32[2];

propertyIDs[0] = (Int32)PropertyID.ACCESSRIGHTS;

propertyIDs[1] = (Int32)PropertyID.DATATYPE;

daServerMgt.GetProperties(ref itemIdentifiers,

ref propertyIDs,
```

```

    false,
    true,
    out itemProperties);

```

### 说明

#### .NET API

的所有方法都允许组调用，换言之，使用一个“**GetProperties**”调用可以查询任意数量的数据项。

在可能的情况下，应该使用组调用，因为该策略一方面可以将客户端与服务器之间的通信减少至最小，另一方面可以充分利用 API 的优化机制。

## Subscribe 方法

使用“**Subscribe**”方法，可将数据项注册以用于监控值的更改。

在创建第一个“**Subscription**”之前，应用程序必须注册 .NET API 的“**DataChanged**”事件。该注册过程将使用“**DataChanged**”事件进行描述。

参数	功能
<b>int clientSubscription</b>	使用“clientSubscription”，应用程序可以指定“Subscription”的索引。 在注册多个“Subscription”时需要该句柄，并且应用程序想要在“DataChanged”事件中区分这些“Subscription”。
<b>bool active</b>	利用“active”标记，该订阅可以创建为激活状态或未激活状态。
<b>int updateRate</b>	“UpdateRate”参数将指定报告值更改的周期。 系统将以毫秒为单位指定“UpdateRate”。
<b>out int revisedUpdateRate</b>	该输出参数将返回由服务器设置的“UpdateRate”。 该值可能与请求的“UpdateRate”存在偏差。
<b>float deadband</b>	“Deadband”参数将指定在报告值的更改之前必须超出的最小偏差。该值为报告值的更改后值范围的百分比偏差（值介于 0.0 到 100.0 之间）。使用 OPC 数据项的“EULow”和“EUHigh”属性指定值范围。 “Deadband”通常仅与模拟值配合使用。

参数	功能
<b>ref ItemIdentifier[] itemIdentifiers</b>	使用“itemIdentifiers”数组指定要在“Subscription”中插入的 OPC 数据项。 任何数据项错误都会在相关“ItemIdentifiers”的“ResultID”对象中进行返回。参数为输入/输出参数，以便 .NET API 能够在对象中返回错误信息和“ServerHandles”。 对象必须能够从“Subscription”中移除各个数据项。
<b>out int serverSubscription</b>	在“serverSubscription”参数中，.NET API 将返回“Subscription”的句柄，而该“Subscription”可使用“Cancel Subscription”方法再次删除。

返回类型	功能
<b>ReturnCode</b>	返回代码将指示所出现的错误是否与传送的参数相关，以及服务器是否支持“UpdateRate”。 应基于枚举器的值检查“ItemIdentifier[]”。

下列代码显示了有关如何创建“Subscription”的示例：

```
ItemIdentifier[] itemIdentifiers = new ItemIdentifier[10];

for (int i = 0; i < 10; i++)
{
    itemIdentifiers[i] = new ItemIdentifier();

    itemIdentifiers[i].ItemName = "S7:[@LOCALSERVER]DB1,B" + i.ToString();
    itemIdentifiers[i].ClientHandle = i;
}

int clientSubscription = 1;

bool active = false;

int updateRate = 500;

int revisedUpdateRate;

float deadBand = 0;

int serverSubscription;
```

```

daServerMgt.Subscribe(clientSubscription,
    active,
    updateRate,
    out revisedUpdateRate,
    deadBand,
    ref itemIdentifiers,
    out serverSubscription);

```

### 说明

为了能够处理所有“DataChanged”事件，必须在生成第一个订阅之前使用 .NET API 注册应用程序的“Eventhandler”。

## SubscriptionModify 方法

使用“SubscriptionModify”方法，可以更改订阅的属性。

参数	功能
<b>int</b> serverSubscription	标识 .NET API 中的订阅。 使用“Subscribe”方法创建订阅时，返回此句柄。
<b>bool</b> active	利用“active”标记，该订阅可以切换为激活状态或未激活状态。 如果传送零，该属性将不会更改。
<b>int</b> updateRate	“updateRate”参数将指定报告值更改的周期。 系统将以毫秒为单位指定“UpdateRate”。 如果传送零，该属性将不会更改。
<b>out int</b> revisedUpdateRate	该输出参数将返回由服务器设置的“UpdateRate”。 该值可能与请求的“UpdateRate”存在偏差。
<b>float</b> deadband	“deadband”参数将指定在报告值的更改之前必须超出的最小偏差。 该值为报告值的更改后值范围的百分比偏差（值介于 0.0 到 100.0 之间）。使用 OPC 数据项的“EULow”和“EUHigh”属性指定值范围。 “Deadband”通常仅与模拟值配合使用。 如果传送零，该属性将不会更改。

返回类型	功能
<b>ReturnCode</b>	返回代码将指示所出现的错误是否与传送的参数相关，以及服务器是否支持“UpdateRate”。 应基于枚举器的值检查“ItemIdentifier[]”。

有“SubscriptionModify”方法的其它 3 个过载可用，“Subscription”的“Active”、“UpdateRate”和“Deadband”设置可在这些过载中进行单独更改。

### SubscriptionAddItems 方法

使用“SubscriptionAddItems”方法，可向现有订阅中添加数据项。

参数	功能
<b>int serverSubscription</b>	标识 .NET API 中的订阅。 使用“Subscribe”方法创建订阅时，返回此句柄。
<b>ref ItemIdentifier[] itemIdentifiers</b>	使用“itemIdentifiers”数组指定要在“Subscription”中插入的 OPC 数据项。 任何数据项错误都会在相关“ItemIdentifiers”的“ResultID”对象中进行返回。参数为输入/输出参数，以便 .NET API 能够在对象中返回错误信息和“ServerHandles”。 对象必须能够从“Subscription”中移除各个数据项。

返回类型	功能
<b>ReturnCode</b>	返回值指示是否发生了涉及传送参数的错误。 应基于枚举器的值检查“ItemIdentifier[]”。

## SubscriptionRemoveItems 方法

使用“SubscriptionRemoveItems”方法，可从现有订阅中移除数据项。

参数	功能
<b>int</b> serverSubscription	标识 .NET API 中的订阅。 使用“Subscribe”方法创建订阅时，返回此句柄。
<b>ref ItemIdentifier[]</b> itemIdentifiers	通过“itemIdentifiers”数组指定“Subscription”中要移除的 OPC 数据项。 任何数据项错误都会在相关“ItemIdentifiers”的“ResultID”对象中进行返回。 此处应使用“Subscribe”或“SubscriptionAddItems”要返回的相同对象来确定带“ServerHandle”的正确对象。 参数为输入/输出参数，这样 .NET API 就可在对象中返回错误信息。

返回类型	功能
<b>ReturnCode</b>	返回值指示是否发生了涉及传递参数的错误。 应基于枚举器的值检查“ItemIdentifier[]”。

## SubscriptionCancel 方法

使用“SubscriptionCancel”方法，可取消“DataChanged”的订阅注册。

同时，必须传递“Subscribe”中包含的“serverSubscription”参数。

参数	功能
<b>int</b> serverSubscription	标识 .NET API 中的订阅。 使用“Subscribe”方法创建订阅时，返回此句柄。

以下代码举例说明了如何再次取消“Subscription”：

```
daServerMgt.SubscriptionCancel(serverSubscription);
```

## DataChanged 事件

使用“DataChanged”事件，.NET API

会将“Subscription”的数据项更改值发送到应用程序。

参数	功能
<b>int clientSubscription</b>	创建订阅时由应用程序传送的句柄。
<b>bool allQualitiesGood</b>	指示“ItemValueCallback”数组中没有质量不良的值。
<b>bool noErrors</b>	指示“ItemValueCallback”数组中没有任何错误。
<b>ItemValueCallback[] itemValues</b>	每个已更改数据项的值、质量和时间戳的数组。 使用“ItemValueCallback”中的“ClientHandle”进行分配。 如果“noErrors”的值为“false”，还必须检查“ItemValueCallback”的“ResultID”属性。

要为 OPC 服务器的“DataChanged”事件注册，必须为此事件注册“Handler”方法。

下列代码显示了关于如何为“DataChanged”事件注册“Handler”方法的示例：

```
daServerMgt.DataChanged += new DaServerMgt.DataChangedEventHandler(MyDataChanged);

void MyDataChanged(int clientSubscription,
                   bool allQualitiesGood,
                   bool noErrors,
                   ItemValueCallback[] itemValues)
{
    if (noErrors)
    {
        if (allQualitiesGood)
        {
            // All items are correct

            // ToDo: Entry of the values in the application
        }
    }
}
```

```

{
    / The quality is not good for at least one item

    // ToDo: Troubleshooting

}

}

else

{

    // At least one item has an error

    // ToDo: Troubleshooting

}

}

```

## ServerStateChanged 事件

通过“ServerStateChanged”事件，.NET API 可向应用程序告知对连接状态所做的更改。

参数	功能
<b>int clientHandle</b>	句柄由应用程序通过“Connect”传送，用于连接至 OPC 服务器。
<b>ServerState state</b>	与 OPC 服务器之间连接的当前状态。

要注册 OPC 服务器的“ServerStateChanged”事件，必须为此事件注册“Handler”方法。

下列代码显示了为“ServerStateChanged”事件注册“Handler”方法的示例：

```

daServerMgt.ServerStateChanged += new
DaServerMgt.ServerStateChangedEventHandler(MyServerStateChanged);

void MyServerStateChanged(int clientHandle,
    ServerState state)
{
    // Identification of the connection
    // ToDo: Reaction to the new status of the server
}

```

### 5.2.1.3 故障排除

出现的任何错误都将通过异常报告给用户。此处，使用两种异常类型。

传送错误参数、空参数或参数的非法组合所引起的错误由 .NET API 系统异常返回。

由底层 OPC 服务器返回的错误将作为 OPC 异常，由 .NET API 返回。

#### OPCExceptions

“OpcException”类源于“ApplicationException”。

作为附加参数，“OPCException”具有“ResultID”。

#### SystemExceptions

.NET OPC API 产生三类不同的“SystemExceptions”。

- ArgumentNullException – 方法不允许采用零参数时使用。
- ArgumentOutOfRangeException – 例如，将大于 100 的数设置为“Deadband”时。
- ArgumentException – 例如，在调用 Write() 时，数组“ItemIdentifiers”和“Values”的长度不同。

```

int ErrorCode;
string ErrorName;
string ErrorDescription;
try
{
    daServerMgt.Write( ref itemIdentifiers,
                      itemValues );
}
catch(OPCEexception opcex)
{
    // Error
    ErrorCode = opcex.ResultID.Code;
    ErrorName = opcex.ResultID.Name;
    ErrorDescription = opcex.ResultID.Description;

    // OPC-specific error handling
    // ToDo. . .
}

```

```

        catch(System.ArgumentException sysarex)
        {
            // Error
            ErrorDescription = sysarex.Message;
            // Special error handling for ArgumentException
            // ToDo. . .
        }

        catch(System.Exception)
        {
            // Error handling for unexpected exception
            // ToDo. . .
        }
    }
}

```

## 5.2.2 命名空间 SIMATICNET.OPCDACLIENT 的示例

本例介绍了将使用 OPCDACLIENT API 的客户端连接至不同 OPC 服务器类型的方法。有关详细信息，请参见所示部分。

### 向 OPC DA 和 OPC UA 服务器（有/无加密）建立连接的示例

```

//Server instance initialization
//Information can be found on this in the section "DaServerMgt"
DaServerMgt m_Server = new DaServerMgt();

//Server identifier instance initialization
//The section "ServerIdentifier class" contains information
//about this
ServerIdentifier m_ServerId =
    new ServerIdentifier(m_EndpointUrl, m_EndpointIdentifier);

//Only connect to server if no connection exists yet
if (m_Server.IsConnected == false)
{
    //Fill in ConnectInfo instance
    //The section "ConnectInfo class" contains information
    //about this
    ConnectInfo connectInfo          = new ConnectInfo();
    connectInfo.RetryAfterConnectionError = true;
    connectInfo.RetryInitialConnection = true;
    connectInfo.KeepAliveTime         = 1000;

    //Set UA-specific parameters if the server to be connected
    //is a UA server
    if (m_ServerId.Category == ServerCategory.OPCUA)

```

```

{
    //Fill in ConnectInfo instance
    //The section "ConnectInfo class" contains information
    //about this
    connectInfo.MessageSecurityMode      =
        m_ServerId.Endpoint.MessageSecurityMode;
    connectInfo.SecurityPolicyUri       =
        m_ServerId.Endpoint.SecurityPolicyUri;
    connectInfo.ServerCertificate      =
        m_ServerId.Endpoint.ServerCertificate;

    //Setting parameters necessary for connection establishment
    //with encrypted UA servers
    if (1 < connectInfo.MessageSecurityMode)
    {
        //Fill in ConnectInfo instance
        //The section "ConnectInfo class" contains information
        //about this
        //Example of storage path in Windows store
        connectInfo.CertificateStoreName   =
            Application.ProductName;
        connectInfo.CertificateStoreLocation =
            WinStoreLocation.CurrentUser;

        //Certificate handling is performed so that
        //each time the application starts
        //a new certificate does not need to be generated that
        in turn
        //needs to be accepted by the server.
        //To achieve this, the thumbprints of the certificates
        used
        //must be stored, for example in the file system.

        #region Certificate Handling Client

        //The thumbprint is necessary to find the client
        certificate
        //again in the Windows store
        byte[] clientThumbprint   = null;
        //Client certificate - Information on this can be found
        //in the section "PkiCertificate class"
        PkiCertificate cert_Client = null;

        //ToDo: Reading out the stored thumbprint of the client
        //"clientThumbprint" must occur here

        //Fetch existing certificate from the Windows store
        if (clientThumbprint != null)
        {
            try
            {
                cert_Client =

```

```
PkiCertificate.fromWindowsStoreWithPrivateKey(
    connectInfo.CertificateStoreLocation,
    connectInfo.CertificateStoreName,
    clientThumbprint);
}
catch
{
    cert_Client = null;
}
}

//If there is not yet a certificate in the Windows store
//it needs to be created
if (cert_Client == null)
{
    string Computername =
        System.Windows.Forms.SystemInformation.ComputerName;
    cert_Client = new PkiCertificate(
        Computername +
        ":CLIENTAPISAMPLE",
        "", Computername, 94608000,
        "CLIENTAPISAMPLE",
        "SIEMENS AG", "UNIT", "LOCAL",
        "COUNTRY", "LANG", 1024);
    cert_Client.toWindowsStoreWithPrivateKey(
        connectInfo.CertificateStoreLocation,
        connectInfo.CertificateStoreName);

    //ToDo: Save the thumbprint
    //"cert_Client.Thumbprint"
    //must occur here
}

#endregion

#region Certificate Handling Server
//The thumbprint is necessary to find the server
certificate
//again in the Windows store
byte[] serverThumbprint      = null;
//Server certificate - Information on this can be found
//in the section "PkiCertificate class"
PkiCertificate cert_Server  = null;

//ToDo: Reading out the stored server thumbprint
//serverThumbprint" must occur here

//Fetch existing certificate from Windows store
if (serverThumbprint != null)
{
    try
    {
```

```
cert_Server =
    PkiCertificate.fromWindowsStore(
        connectInfo.CertificateStoreLocation,
        connectInfo.CertificateStoreName,
        serverThumbprint);
}
catch
{
    cert_Server = null;
}
}

//If there is not yet a certificate in the Windows store
//the certificate of the server to be connected must
//be stored in the same Windows Store as the client
//certificate so that it counts as accepted
if (cert_Server == null)
{
    //Here it is possible to query whether the new
    //certificate will be accepted

    cert_Server =
        PkiCertificate.fromDER(connectInfo.ServerCertificate);

    cert_Server.toWindowsStore(
        connectInfo.CertificateStoreLocation,
        connectInfo.CertificateStoreName);

    //ToDo: Save the thumbprint
    //"cert_Server.Thumbprint" must occur here
}
else
{
    //Here it is possible to query whether the current
    //server certificate and the stored certificate
    //match
}

#endregion

//Set private key
connectInfo.ClientPrivateKey = cert_Client.PrivateKey;

//Encrypt certificate
connectInfo.ClientCertificate = cert_Client.toDER();
}
}

bool connectFailed;
```

```
//Send connection job to the server
m_Server.Connect(m_ServerId.Url, m_ClientHandle,
    ref connectInfo, out connectFailed);

if (connectFailed)
{
    //Here, errors in the connection establishment to the
    //server
    //can be handled
}
else
{
    //Server is already connected
}
```

## 5.2.3 SIMATICNET.OPCCMN 命名空间

“SimaticNet.OpcCmn”命名空间将 OPC 的常用功能组合到一起。尤其是包含了浏览 OPC 服务器以及获取建立连接所需信息的功能。

### 5.2.3.1 OPCServerEnum 对象的接口

使用“OpcServerEnum”类，可在计算机上搜索 COM OPC 服务器。还可获取与 OPC 服务器之间建立连接所需的信息。

#### 生成 OPCServerEnum 对象

要使用“OpcServerEnum”类，必须先生成类的实例。

下列代码显示了如何创建“OpcServerEnum”对象：

```
OpcServerEnum opcServerEnum = new OpcServerEnum();
```

## EnumComServers 方法

“EnumComServers”方法用于寻找计算机上的 OPC 服务器。此处可区分开不同的 OPC 规范，如“OPC 数据访问”、“OPC 历史数据访问”或“OPC 报警和事件”。

参数	功能
<b>string nodeName</b>	要从中浏览 OPC 服务器的计算机的名称或 IP 地址。（例如 localhost、PCTest、192.168.0.120 等。）
<b>bool returnAllServers</b>	指定是否应返回计算机中的所有 OPC 服务器。 如果此参数设置为“true”，则不会评估数组“serverCategories”。
<b>ServerCategory[] serverCategories</b>	通过此参数，可以指定所返回 OPC 服务器的类型。
<b>out ServerIdentifier[] serverIdentifiers</b>	“ServerIdentifier”对象的列表。对象包含与 OPC 服务器建立连接所需的信息。

以下代码可返回本地计算机上的所有 OPC 数据访问服务器：

```
ServerCategory[] serverCategories = new ServerCategory[1];

serverCategories[0] = ServerCategory.OPCDA;

ServerIdentifier[] serverIdentifier = null;

opcServerEnum.EnumComServer("localhost",
false,
serverCategories,
out serverIdentifier);
```

## ClsidFromProgID 方法

“**ClsidFromProgID**”方法可根据传送的计算机名称和“**ProgID**”参数获取 OPC 服务器的相应“**CLSID**”。

参数	功能
<b>string nodeName</b>	要从中获取“ <b>CLSID</b> ”的计算机的名称或 IP 地址。 (例如 localhost、PCTest、192.168.0.120 等。)
<b>string progID</b>	要获取相应“ <b>CLSID</b> ”的“ <b>ProgID</b> ”。
<b>out string clsID</b>	请求的“ <b>CLSID</b> ”

以下代码可返回本地计算机中“**ProgID** OPC.SimaticNET”的“**CLSID**”：

```
string progID = "OPC.SimaticNET";
string clsID = null;
opcServerEnum.ClsidFromProgId("localhost",
    progID,
    out clsID);
```

## getCertificateForEndpoint 方法

“**getCertificateForEndpoint**”方法可将其本身连接至传送的端点 URL 并加载作为输出参数返回的服务器证书。  
此前，在发现服务器中进行浏览可找到输入参数。

参数	功能
<b>string endpointUrl</b>	UA 端点的 URL (例如 opc.tcp://localhost:4841)
<b>string securityPolicyUri</b>	端点的 SecurityPolicy (如“http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15”)
<b>Byte messageSecurityMode</b>	端点的 MessageSecurityMode None、Sign 或 SignAndEncrypt
<b>out byte[] serverCertificate</b>	OPC UA 服务器的证书。

以下代码可从参数指定的端点加载证书：

```
string url = "opc.tcp://localhost:55101";
string securityPolicyUri =
"http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa";
byte messageSecurityMode = 3; // Sign and Encrypt
byte[] serverCertificate = null;
opcServerEnum.getCertificateForEndpoint(Url,
                                         securityPolicyUri,
                                         messageSecurityMode
                                         out serverCertificate);
```

### 5.2.3.2 ServerIdentifier 类

“ServerIdentifier”类可返回向 OPC 服务器建立连接所需的所有数据。

“ServerIdentifier”类的属性：

- **string Url**

OPC 服务器的完整 URL。 可通过“Connect”方法对此进行传送。 另请参见 Url 的结构。

- **ServerCategory Category**

服务器可划分到的类别，例如 OPC DA、OPC DX 等。

- **string ProgID**

OPC 服务器的“ProgID”。

- **string CLSID**

OPC 服务器的“CLSID”。

- **string HostName**

发现 OPC 服务器的计算机的名称。

- **EndpointIdentifier**

连接至 OPC UA 服务器的必需附加信息。

### 5.2.3.3 ServerCategory 枚举器

通过“ServerCategory”枚举器，可指定 OPC 服务器的类型。 其中包含下列元素：

- **OPCDA**

对应于 OPC 规范“OPC DA 2.05A”和“OPC DA 3.00”

- **OPCUA**

对应于“OPC 统一架构”规范

- **OPCXMLDA**

对应于 OPC 规范“OPC XML DA 1.01”（当前不支持）

- OPCDX  
对应于 OPC 规范“OPC DX 1.00”（当前不支持）
- OPCAE  
对应于 OPC 规范“OPC AE 1.10”（当前不支持）
- OPCHDA  
对应于 OPC 规范“OPC HDA 1.10”（当前不支持）

### 说明

与 COM OPC 服务器不同，OPC XML DA 服务器不在计算机上注册，因此不能使用“OpcServerEnum”获取。要连接 OPC XML DA 服务器，必须知道 URL。

#### 5.2.3.4 EndpointIdentifier 类

“EndpointIdentifier”类可返回包括建立 UA 服务器连接所需数据在内的所有所需数据。

“EndpointIdentifier”类的属性：

- **string SecurityPolicy**  
用于保护连接的 SecurityPolicy 的 URI。
- **byte MessageSecurityMode**  
消息级别连接安全的类型。
- **byte ServerCertificate**  
OPC-UA 服务器的 X509 证书。
- **string ApplicationUri**  
OPC UA 服务器应用程序的 URI。
- **string ProductUri**  
OPC UA 服务器应用程序的产品 URI。
- **string ApplicationName**  
OPC UA 服务器的名称。

#### 5.2.3.5 PkiCertificate 类

“PkiCertificate”类可封装 X509 证书并可用于简便地生成此类证书。

还可提供访问“WindowsCertificateStore”以加载和保存证书的方法。

只能使用“属性”来读取证书的字段。

### 5.2.3.6 创建新证书

“**PkiCertificate**”类可提供构造函数作为其中含有生成证书所需全部信息的参数。

参数	功能
<b>string URI</b>	证书的“ApplicationURI”。证书的唯一标识符，例如： urn:<computername>:<company>:<productname>
<b>string IP</b>	运行应用程序所在计算机的 IP 地址。 仅在无可用“域名”时使用。
<b>string DNS</b>	运行应用程序所在计算机的“域名”（计算机名称）。
<b>int validTime</b>	证书有效时间（秒）。
<b>string CommonName</b>	证书的显示名称。
<b>string Organization</b>	组织或公司（例如 Siemens）
<b>string OrganizationUnit</b>	组织或公司的部门。
<b>string Locality</b>	发行机构的位置。
<b>string State</b>	发行机构的州或民族国家。
<b>string Country</b>	国家/地区代码，例如 DE、US...
<b>int KeyStrength</b>	密钥长度，例如 1024 位、2048 位...

以下代码显示了如何生成新的“**PkiCertificate**”的示例：

```
PkiCertificate clientCert = new PkiCertificate(
    "urn:my_machine:Siemens:SimaticNETSampleApplication",
    "",
    "Computer1",
    31536000, // 1 year = 31536000 seconds
    "SampleApplication",
    "Siemens",
    "SimaticNET",
    "Karlsruhe",
    "Baden-Württemberg",
    "DE",
    2048);
```

### 5.2.3.7 toDER 方法

使用此方法，可通过“**PkiCertificate**”生成 DER 编码字节数组。诸如 **EndpointIdentifier** 类会需要 DER 编码字节数组形式的证书。

返回类型	功能
<b>byte[]</b>	包含证书 DER 编码数据的字节数组，或零（转换失败时）。

以下代码显示了如何通过现有 **PkiCertificate** 生成 DER 编码字节数组的示例：

```
// Load client certificate from Windows Store
PkiCertificate clientCert;
...
byte[] clientCertificateDER = clientCertificate.toDER();
```

### 5.2.3.8 fromDER 方法

使用此静态方法，可通过 DER 编码字节数组生成“**PkiCertificate**”。

参数	功能
<b>byte[] DERData</b>	包含 X509 证书数据的 DER 编码字节数组。

返回类型	功能
<b>PkiCertificate</b>	新创建的“ <b>PkiCertificate</b> ”，或为空（转换失败时）。

以下代码显示了如何通过现有 **byte[]** 生成“**PkiCertificate**”的示例：

```
Byte[] DERdata;
...
PkiCertificate cert = PkiCertificate.fromDER(DERdata);
```

### 5.2.3.9 toWindowsStore 方法

使用此方法，可将“PkiCertificate”保存到“WindowsCertificateStore”中。

参数	功能
<b>WinStoreLocation location</b>	用于保存到“WindowsCertificateStore”中的用户上下文。
<b>string StoreName</b>	用于保存到“WindowsCertificateStore”中的“Store”的名称。

以下代码示例显示了将现有“PkiCertificate”保存在“WindowsCertificateStore”中的方法：

```
PkiCertificate cert;
...
Cert.toWindowsStore(WinStoreLocation.CurrentUser, "UA
Applications");
```

#### 说明

建议将证书存储在 StoreName“UA Applications”下的“WinStoreLocation.LocalMachine”中，这是 OPC 基金会的建议。

### 5.2.3.10 toWindowsStoreWithPrivateKey 方法

使用此方法，可将包含私钥的“PkiCertificate”保存到“WindowsCertificateStore”中。

参数	功能
<b>WinStoreLocation location</b>	用于保存到“WindowsCertificateStore”中的用户上下文。
<b>string StoreName</b>	用于保存到“WindowsCertificateStore”中的“Store”的名称。

以下代码示例显示了将现有“PkiCertificate”保存在“WindowsCertificateStore”中的方法：

```
PkiCertificate cert;
...
Cert.toWindowsStore(WinStoreLocation.CurrentUser, "MyStoreName");
```

### 5.2.3.11 fromWindowsStore 方法

使用此方法，可从“WindowsCertificateStore”中加载“PkiCertificate”。

参数	功能
<b>WinStoreLocation location</b>	用于从“WindowsCertificateStore”中进行加载的用户上下文。
<b>string StoreName</b>	用于从“WindowsCertificateStore”中进行加载的“Store”的名称。
<b>byte[] Thumbprint</b>	证书的“Thumbprint”。 可通过组态文件等途径加载。

返回类型	功能
<b>PkiCertificate</b>	新创建的“PkiCertificate”，或为空（加载失败时）。

以下代码示例显示了从“WindowsCertificateStore”中加载“PkiCertificate”的方法：

```
byte[] Thumbprint;
// ... load Thumbprint from configuration ...
PkiCertificate cert = PkiCertificate.fromWindowsStore(
    WinStoreLocation.CurrentUser,
    "MyStoreName",
    Thumbprint);
```

### 5.2.3.12 fromWindowsStoreWithPrivateKey 方法

使用此方法，可从“WindowsCertificateStore”中加载“PkiCertificate”及关联私钥。  
私钥是“PkiCertificate”的一个属性。

参数	功能
<b>WinStoreLocation location</b>	用于从“WindowsCertificateStore”中进行加载的用户上下文。
<b>string StoreName</b>	用于从“WindowsCertificateStore”中进行加载的“Store”的名称。
<b>byte[] Thumbprint</b>	证书的“Thumbprint”。 可通过组态文件等途径加载。

返回类型	功能
<b>PkiCertificate</b>	新创建的“PkiCertificate”，或为空（加载失败时）。

以下代码示例显示了从“WindowsCertificateStore”中加载“PkiCertificate”及关联私钥的方法：

```
byte[] Thumbprint;
// ... load Thumbprint from configuration ...
PkiCertificate cert =
PkiCertificate.fromWindowsStoreWithPrivateKey(
    WinStoreLocation.CurrentUser,
    "MyStoreName",
    Thumbprint);
```

#### 5.2.3.13 **fromWindowsStoreWithPrivateKey** 方法

使用此方法，可从“WindowsCertificateStore”中加载“PkiCertificate”及关联私钥。私钥是“PkiCertificate”的一个属性。

参数	功能
<b>WinStoreLocation location</b>	用于从“WindowsCertificateStore”中进行加载的用户上下文。
<b>string StoreName</b>	用于从“WindowsCertificateStore”中进行加载的“Store”的名称。
<b>string ApplicationURI</b>	证书的“Thumbprint”。可通过组态文件等途径加载。

返回类型	功能
<b>PkiCertificate</b>	新创建的“PkiCertificate”，或为空（加载失败时）。

以下代码示例显示了从“WindowsCertificateStore”中加载“PkiCertificate”及关联私钥的方法：

```
String sApplicationUri;
// ... load ApplicationUri from configuration ...
```

```
PkiCertificate cert =
PkiCertificate.fromWindowsStoreWithPrivateKey(
    WinStoreLocation.CurrentUser,
    "MyStoreName",
    sApplicationUri);
```

### 5.2.3.14 属性

此类的属性均为只读属性。它们代表了内部维护的 X509 证书的不同的框。

- PrivateKey [只读]
- CommonName [只读]
- Thumbprint [只读]
- IPAddress [只读]
- ApplicationURI [只读]
- ValidFrom [只读]
- ValidTo [只读]
- SerialNumber [只读]
- SignatureAlgorithm [只读]
- CipherStrength [只读]
- Subject [只读]
- Issuer [只读]

### 5.2.3.15 枚举器 WinStoreLocation

使用“枚举器

`WinStoreLocation`”，可指定在“`WindowsCertificateStore`”中加载或保存证书所在的用户上下文。

其中包含下列元素：

- LocalMachine  
证书存储在注册表中的以下位置：  
“`HKEY_LOCAL_MACHINE\Software\Microsoft\SystemCertificates`”
- CurrentUser  
证书存储在注册表中的以下位置：  
“`HKEY_CURRENT_USER\Software\Microsoft\SystemCertificates`”

- CurrentService

证书存储在注册表中的以下位置:

“HKEY\_LOCAL\_MACHINE\Software\Microsoft\Cryptography\Services\<ServiceName>\SystemCertificates”

- Services

证书存储在注册表中的以下位置:

“HKEY\_LOCAL\_MACHINE\Software\Microsoft\Cryptography\Services\<ServiceName>\SystemCertificates”

- Users

证书存储在注册表中的以下位置:

“HKEY\_USERS\<UserName>\Software\Microsoft\SystemCertificates”





# 示例程序

本部分包含使用自定义或自动化接口的示例程序。

在安装了 SIMATIC NET PC 软件后，您将会在以下位置找到示例程序：

“<安装路径>\SIEMENS\SIMATIC.NET\opc2\samples”

## 6.1 VB.NET 中的 OPC 自动化接口（同步通信）

该示例包括 Visual Basic，并使用 OPC 的 Data Access V2.0 的自动化接口同步读写数据。

该说明分为以下几个部分：

### 6.1.1 激活仿真连接

运行此程序前，必须激活仿真连接以使程序中所使用的演示变量可用。请按照下面列出的步骤进行操作：

1. 从“开始”(Start) 菜单中启动“通信设置”组态程序：

“开始 > 所有程序 > Siemens Automation > SIMATIC > SIMATIC NET > 通信设置”(Start > All Programs > Siemens Automation > SIMATIC > SIMATIC NET > Communication Settings)。

响应： 打开“通信设置”组态程序。

2. 在左侧的导航窗口中，打开条目“OPC 协议选择”(OPC protocol selection)。

“SIMATIC NET 设置 > OPC 设置 > OPC 协议选择”(SIMATIC NET Settings > OPC Settings > OPC Protocol Selection)。

3. 启用待仿真协议的复选框。

本例使用 S7 协议。

因此，选中“名称： S7”(Name: S7) 之下的复选框，并单击 S7 协议旁边的箭头符号。

响应： 打开扩展参数列表。

4. 选中“为仿真提供虚拟模块 (DEMO)”(Provide virtual module (DEMO) for the simulation) 复选框。

## 6.1 VB.NET 中的 OPC 自动化接口 (同步通信)

5. 单击“应用”(Apply) 确认。
6. 关闭“通信设置”(Communication Settings) 组态程序。

---

### 说明

在更改生效之前，必须首先关闭所有 OPC 客户端！

---

## 6.1.2 使用示例程序

该程序位于硬盘的

“<安装路径>\SIEMENS\SIMATIC.NET\opc2\samples\automation\sync.net”中

启动程序时，仅启用“启动示例”(Start Sample) 按钮：

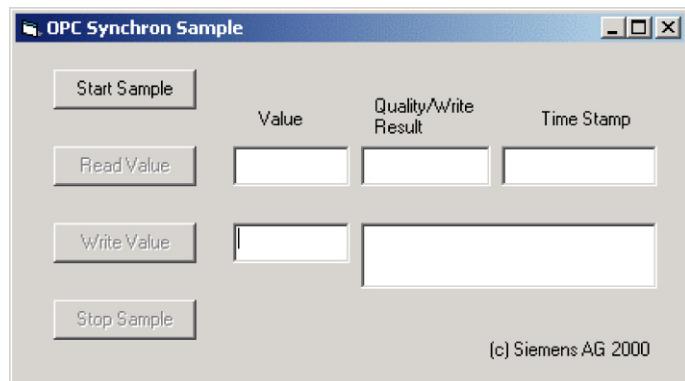


图 6-1 启动 OPC 自动化接口的示例程序后出现的对话框

单击“启动示例”(Start Sample) 按钮后，程序将生成所需的 OPC 对象。

然后，便可使用其它按钮。

在单击“读取值”(Read Value) 按钮后，所读取的值会显示在相关文本框中：

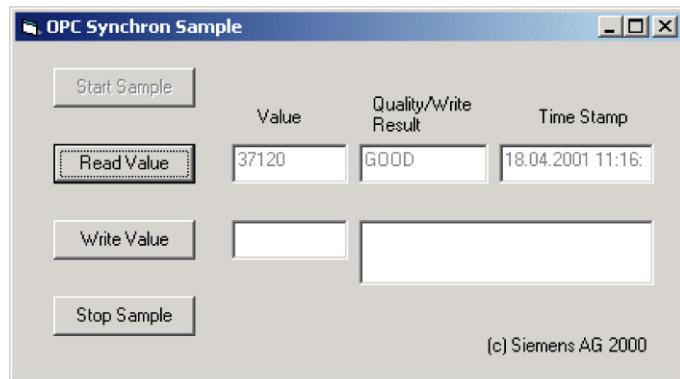


图 6-2 单击“读取值”按钮后出现的对话框

按照实际情况来说，示例程序专用于通过演示连接进行操作。

如果要在真实环境下运行示例程序，必须更改程序代码中的以下行：

```
Set ItemObj = GroupObj.OPCItems.AddItem("S7:[DEMO]MW1", 1)
```

有关 **ItemID** 结构的详细信息，可参考若干示例和“适用于 SIMATIC NET 的 OPC 过程变量 (页 23)”部分。

要写入一个值，必须将其输入到相关文本框中。单击“写入值”(Write Value)按钮后，程序会显示一条有关此操作的结果的消息：

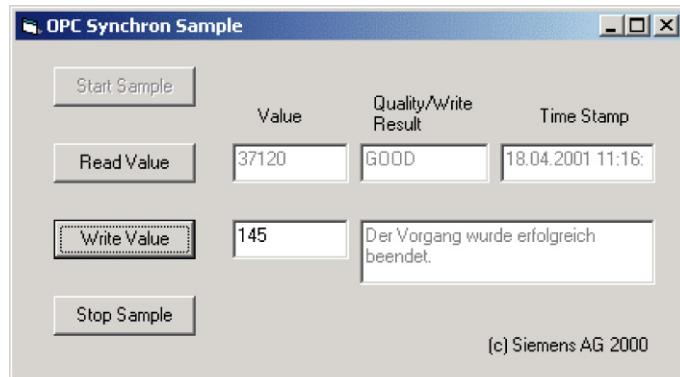


图 6-3 将一个值写入相关文本框并单击“写入值”按钮后出现的对话框

#### “停止示例”(Stop Sample)

按钮将关闭程序：换言之，所有对象将被删除，并且相应资源会被再次释放。

### 6.1.3 如何运行程序？

由于 OPC 的类模型，在调用 OPC 对象的方法时必须遵守特定的顺序。为了能够创建 *OPCItem* 类的实例，*OPCGroup* 类的对象必不可少。然而，只有在 *OPCServer* 类的实例已存在并已建立与该服务器的连接时，才能实现该操作。

下图阐述了创建和删除 OPC 对象的指令的基本顺序。使用示例程序的变量名称。

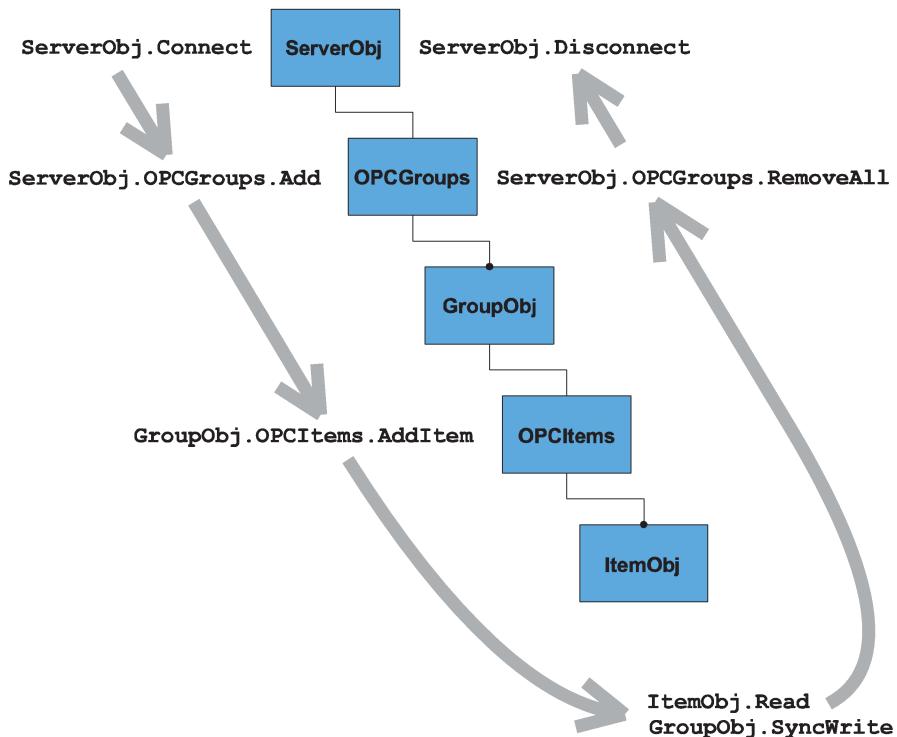


图 6-4 创建和删除 OPC 对象的命令的顺序

该示例程序包含通常存在于典型客户端应用程序中的所有组件。其中包括建立与 OPC 服务器的连接、使用变量创建组、读取和写入数据项值。

### 6.1.4 将 OPC 自动化接口与 .NET-Framework 配合使用

#### 简介

此部分描述了用于在 OPC 自动化接口上通过 Visual Basic.NET 的 .NET-Framework 进行同步读写的示例程序的使用和移植。

安装 SIMATIC NET 软件后，您会在硬盘的下列文件夹中找到该程序：

“<安装路径>\SIEMENS\SIMATIC.NET\opc2\samples\automation\sync.net”

此示例的使用方式和执行方式与先前的示例相同。以下部分仅描述了必要的更改。

### 使用示例程序的要求

.NET-Framework 版本 4.0 或更高版本必须安装在要运行示例程序的计算机上。

在现有 VB.NET 项目中，已经增加了 COM 组件“Siemens OPC DA Automation 2.0”：

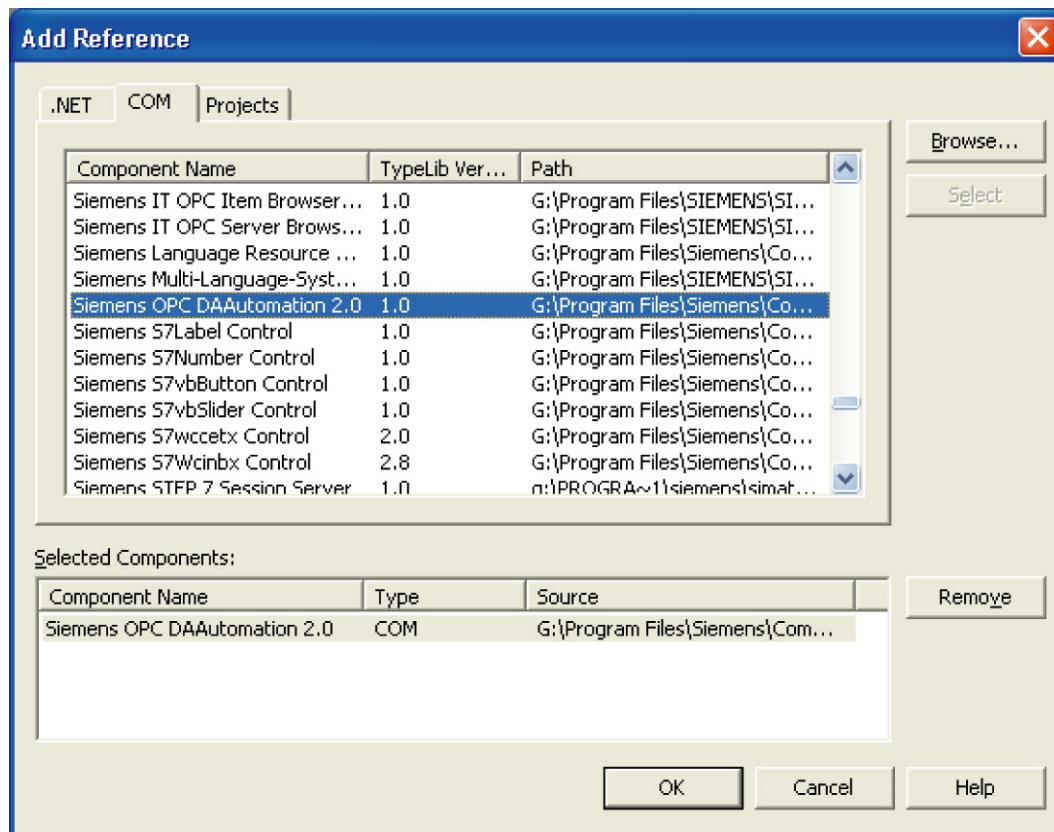


图 6-5 在 VB.NET 项目中增加“Siemens OPC DA Automation 2.0”COM 组件

### 使用命令

```
Imports OPCSiemensDAAutomation
```

将 OPC 自动化接口的命名空间和方法与 .NET Framework 配合使用将十分简单。

## 示例程序

### 6.1 VB.NET 中的 OPC 自动化接口 (同步通信)

#### 与 OPC 服务器建立连接，并添加一个组和一个数据项

使用 *Connect*、*Add* 和 *AddItem* 方法，该程序与先前的示例相似：

```
ServerObj = New OPCServer  
ServerObj.Connect("OPC.SimaticNET")  
  
GroupsObj = ServerObj.OPCGroups  
GroupObj = GroupsObj.Add("MyOPCGroup")  
  
ItemObj = GroupObj.OPCItems.AddItem("S7:[DEMO]MW1", 1)
```

#### 同步读取

为同步读取值，需要调用 **OPCItem** 类的 *Read* 方法：

```
ItemObj.Read(OPCDevice, myValue, myQuality, myTimeStamp)
```

#### 同步写入

---

##### 说明

Visual Basic.NET 使用值 0（而不是作为先前 Visual Basic 中的最小数组索引的值 1）作为最小数组索引，并根据 OPC 自动化接口进行调整。

---

因此，已导入命名空间的数组对象 *OPCSiemensDAAutomation* 以索引 0 开始。在示例中，已为传递值创建特殊数组（数组限制是 1）：

```
Dim Dims() As Integer = New Integer() {1}  
Dim Bounds() As Integer = New Integer() {1}  
Dim Serverhandles As Array =  
    Array.CreateInstance(GetType(Integer),  
        Dims,  
        Bounds)  
  
Dim MyErrors As Array =  
    Array.CreateInstance(GetType(Integer),  
        Dims,  
        Bounds)  
  
Dim MyValues As Array =  
    Array.CreateInstance(GetType(Object),  
        Dims,  
        Bounds)
```

仍然必须初始化这些值。添加数据项时，服务器句柄收到值 1：

```
Serverhandles.SetValue(ItemObj.ServerHandle, 1)  
MyErrors.SetValue(0, 1)
```

然后，用户输入文本框的 **Text** 属性的值将被分配给 *MyValues* 框的第一个组件：

```
MyValues.SetValue(Edit_WriteVal.Text, 1)
```

在已声明和初始化全部所需的本地变量后，调用 **OPCGroup** 类的 **SyncWrite** 方法：

```
GroupObj.SyncWrite(1, Serverhandles, MyValues, MyErrors)
```

## 6.2 C++ 中的 OPC 自定义接口 (同步通信)

Visual C++ 中的示例使用 Microsoft Foundation Classes (MFC) 和 OPC 的 Data Access V2.0 的自定义接口同步读写数据。

该说明分为以下几个部分：

- 激活仿真连接
- 使用示例程序
- 如何运行程序？
- 程序说明
- 有关编写您自己的程序的注意事项

### 6.2.1 激活仿真连接

运行此程序前，必须激活仿真连接以使程序中所使用的演示变量可用。请按照下面列出的步骤进行操作：

- 从“开始”(Start) 菜单中启动“通信设置”组态程序：

“开始 > 所有程序 > Siemens Automation > SIMATIC > SIMATIC NET > 通信设置”(Start > All Programs > Siemens Automation > SIMATIC > SIMATIC NET > Communication Settings)。

响应：打开“通信设置”组态程序。

- 在左侧的导航窗口中，打开条目“OPC 协议选择”(OPC protocol selection)。

“SIMATIC NET Configuration > OPC 设置 > OPC 协议选择”(SIMATIC NET Configuration > OPC Settings > OPC Protocol Selection)。

- 单击 S7 协议旁边的箭头符号，并选中“使虚拟模块 (DEMO) 可用于仿真”(make virtual module (DEMO) available for simulation) 复选框。

- 单击“应用”(Apply) 确认。

- 关闭“通信设置”(Communication Settings) 组态程序。

#### 说明

在更改生效之前，必须首先关闭所有 OPC 客户端！

### 6.2.2 使用示例程序

启动程序时，仅启用“启动示例”(Start Sample) 按钮。单击此按钮后，程序将生成所需的 OPC 对象。然后，便可使用其它按钮。

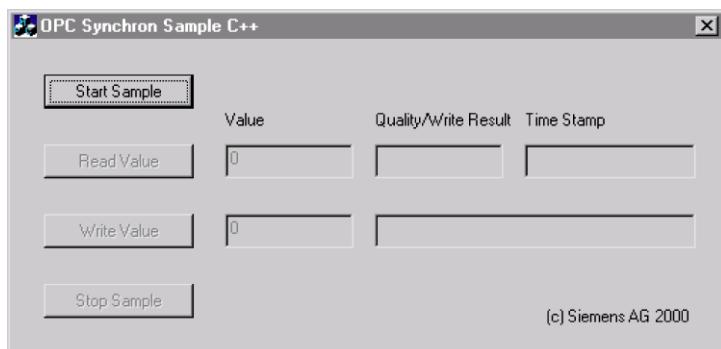


图 6-6 启动 OPC 自定义接口的示例程序后出现的对话框

在单击“读取值”(Read Value) 按钮后，所读取的值会显示在相关文本框中。

按照实际情况来说，示例程序专用于通过演示连接进行操作。

如果要在真实环境下运行示例程序，必须更改“OpcSyncDlg.cpp”文件中的以下行：

```
LPWSTR szItemID = L"S7:[DEMO]MW1";
```

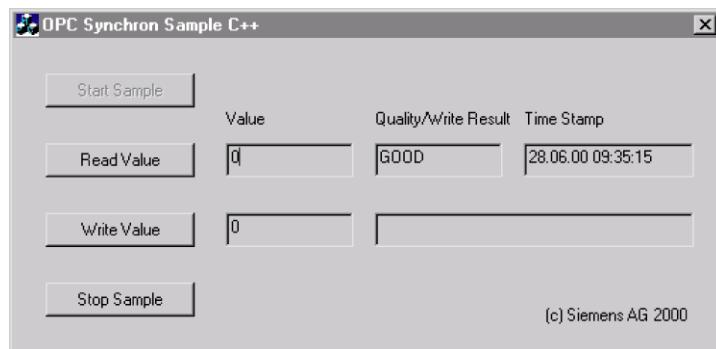


图 6-7 在单击“读取值”(Read Value) 按钮后显示读取值

要写入一个值，必须将其输入到相关文本框中。单击“写入值”(Write Value) 按钮后，程序会显示一条有关此操作的结果的消息：

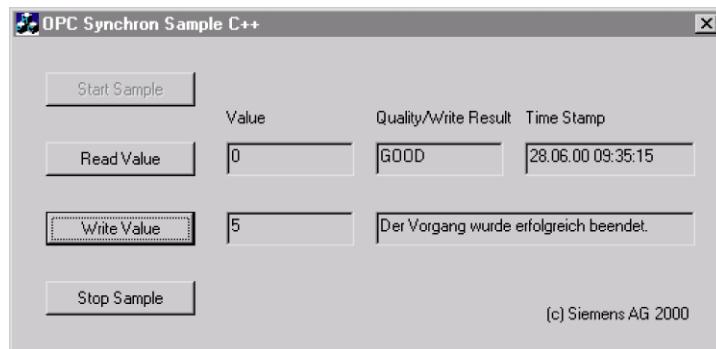


图 6-8 在单击“写入值”按钮后显示结果

#### “停止示例”(Stop Sample)

按钮将关闭程序：换言之，所有对象将被删除，并且相应资源会被再次释放。

### 6.2.3 如何运行程序?

由于 OPC 的类模型，在调用 OPC 对象的方法时必须遵守特定的顺序。为了能够创建 *OPCItem* 类的实例，*OPCGroup* 类的对象必不可少。然而，只有在 *OPCServer* 类的实例已存在并已建立与该服务器的连接时，才能实现该操作。

下图阐述了创建和删除 OPC 对象的指令的基本顺序。使用示例程序的变量名称。

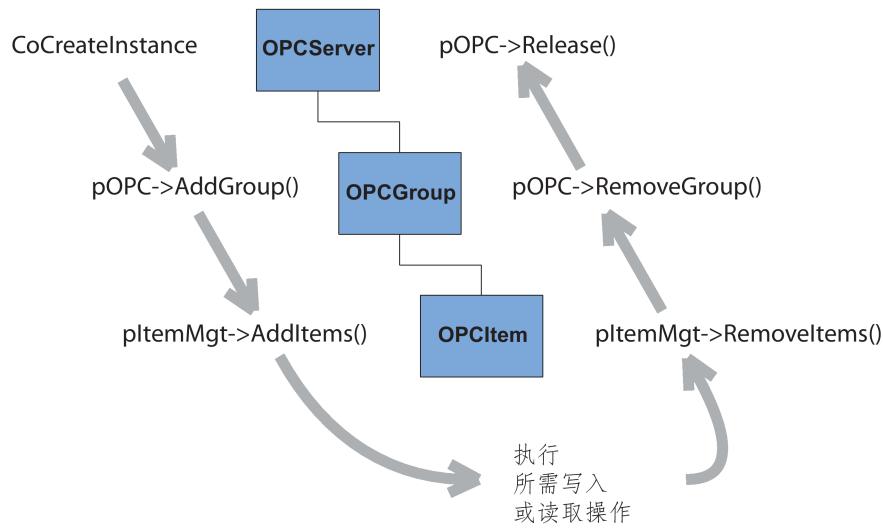


图 6-9 创建和删除 OPC 对象的命令的顺序

示例程序包含通常情况下存在于典型客户端应用程序中的所有组件。其中包括建立与 OPC 服务器的连接、创建变量组、读取和写入数据项值。

在详细说明源代码之前，我们会查看 OPC 应用程序的基本结构。

步骤	说明
1	注册 COM
2	将 ProgID 转换为 CLSID
3	建立与 OPC 服务器的连接
4	创建 OPC 组
5	添加数据项
6	请求 IOPCSyncIO 的接口指针
7	删除对象并释放内存

## 步骤 1：注册 COM

想要使用 COM 库函数的每个程序必须首先自行注册 COM。这是 *CoInitialize()* 函数的用途：

```
HRESULT r1;
r1 = CoInitialize(NULL);
```

## 步骤 2：将 ProgID 转换为 CLSID

要对其进行标识，每个 COM 服务器都有一个被分配给世界上唯一的 *CLSID* (128 位代码) 的 *ProgID*。由于 *CLSID* 需要作为以后函数的参数，其必须从具有 *CLSIDFromProgID()* 函数的 *ProgID* 中获得。SIMATIC NET 的 OPC 服务器的 *ProgID* 是 L"OPC.SimaticNET"：

```
r1 = CLSIDFromProgID(L"OPC.SimaticNET", &clsid);
```

## 步骤 3：建立与 OPC 服务器的连接

*CoCreateInstance()* 函数将创建已指定 *CLSID* 的类的实例。

```
r1 = CoCreateInstance(clsid, NULL, CLSCTX_LOCAL_SERVER ,
IID_IOPCServer, (void**)&m_pIOPCServer);
```

此程序段的结果是 OPC 服务器类的对象。*CoCreateInstance* 还会提供指向服务器对象的 *IOPCServer* 接口的指针 (*m\_pIOPCServer* 变量)：

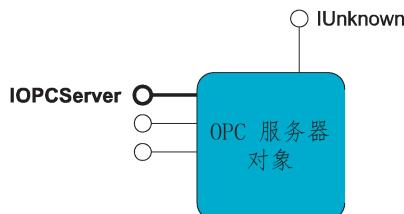


图 6-10 具有指向 *IOPCServer* 接口的指针的 OPC 服务器类的对象

## 步骤 4：创建 OPC 组

*IOPCServer* 接口具有用于创建组的 *AddGroup()* 方法：

```
r1 = m_pIOPCServer->AddGroup(L"grp1", TRUE, 500, 1,
&TimeBias, &PercentDeadband, LOCALE_ID,
&m_GrpSrvHandle, &RevisedUpdateRate,
```

```
IID_IOPCItemMgt,
(LPUNKNOWN*)&m_pIOPCItemMgt);
```

该程序段的结果是一个具有指定名称和所需属性的组。

此外，指向组对象的请求接口的指针将作为返回参数（在这种情况下，是 *IOPCItemMgt*）而存在（*m\_pIOPCItemMgt* 变量）：

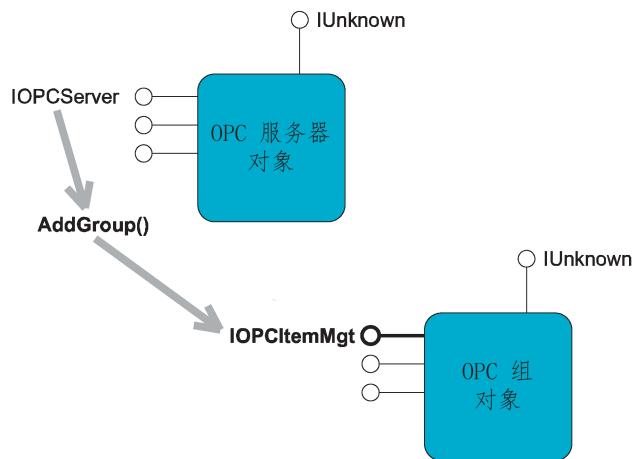


图 6-11 在组对象的请求的 *IOPCItemMgt* 接口上生成 OPC 组

## 步骤 5：添加数据项

*IOPCItemMgt* 接口具有用于创建 OPC 数据项的 *AddItems()* 方法：

```
r1 = pItemMgt->AddItems(1, m_Items, &m_pItemResult, &m_pErrors);
```

此步骤的结果是具有指定属性的所需数量的数据项。此外，还会对结果结构的变量 *m\_pItemResult*（服务器句柄、目标系统上的数据项类型，等等）进行初始化。

## 步骤 6：请求 *IOPCSyncIO* 的接口指针

要允许使用同步读写的方法，指向 *IOPCSyncIO* 接口的指针必不可少。要求使用指向 *IOPCItemMgt* 接口的已存在的指针：

```
r1 = m_pIOPCItemMgt->QueryInterface(IID_IOPCSyncIO,
                                         (void**)&m_pIOPCSyncIO);
```

使用接口的 *Read()* 和 *Write()* 方法，可读取或写入数据项的值：

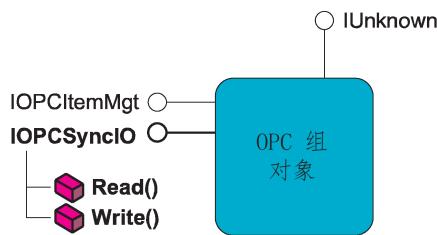


图 6-12 具有指向 *IOPCSyncIO* 接口的指针的 OPC 组对象及其方法 *Read()* 和 *Write()*

### 步骤 7：删除对象并释放内存

退出程序之前，必须删除已创建的 OPC 对象并释放为其保留的内存。相关的函数是先前使用的接口的重要组成部分。

```

r1 = m_pIOPCItemMgt->RemoveItems(1, phServer, &pErrors);
CoTaskMemFree(m_pItemResult);
m_pItemResult=NULL;
CoTaskMemFree(m_pErrors);
m_pErrors = NULL;
m_pIOPCSyncIO->Release();
m_pIOPCSyncIO = NULL;
m_pIOPCItemMgt->Release();
m_pIOPCItemMgt = NULL;
r1 = m_pIOPCServer->RemoveGroup(m_GrpSrvHandle, TRUE);
m_GrpSrvHandle = NULL;
m_pIOPCServer->Release();
m_pIOPCServer = NULL;
CoUninitialize();

```

#### 6.2.4 OPCDA\_SyncDlg.cpp 程序的说明

“OPCDA\_SyncDlg.cpp”文件可以分为以下两部分：

- 全局变量的声明
- COPCDA\_SyncDlg 类的方法

## 全局变量

在该模块的开始处，使用数据项标识符对全局变量进行初始化。

为确保示例程序正确运行，该变量必须可读且可写。

```
const LPWSTR szItemID = L"S7:[DEMO]MW1";
```

有关 **ItemID** 结构的详细信息，可参考若干示例和手册。

## COPCDA\_SyncDlg 类的方法

该模块的主要组件是由 MFC 调用的主对话框按钮的事件过程。此外，还必须在 *OnInitDialog* 方法中进行一些设置：

- **OnInitDialog**
- **OnStart**
- **OnRead**
- **OnWrite**
- **OnStop**
- **DestroyWindow**

### 6.2.4.1 **OnInitDialog**

有多个类变量可用于访问 COM 机制，而该机制是在初始化主对话框时进行初始化的。这些变量可具有以下用途：

- **m\_pIOPCServer** 是一个指向 OPC 服务器类的 IOPCServer 接口的指针。
- **m\_pIOPCItemMgt** 是一个指向 OPC 组类的 IOPCItemMgt 接口的指针。
- **m\_pIOPCSyncIO** 是一个指向 OPC 组类的 IOPCSyncIO 接口的指针。

```
m_pIOPCServer = NULL;  
m_pIOPCItemMgt = NULL;  
m_pIOPCSyncIO = NULL;
```

在程序开始时，用户只能单击“启动示例”(Start Sample) 按钮。

所有其它按钮都将被禁用。这会确保在执行写入或读取操作前，正确地设置全部所需的 OPC 对象。这将使用以下程序段来实现：

```
m_CtrlStop.EnableWindow(FALSE);  
m_CtrlRead.EnableWindow(FALSE);  
m_CtrlWrite.EnableWindow(FALSE);
```

### 6.2.4.2 OnStart

*OnStart* 将设置运行程序所需的 OPC 对象。首先，声明并初始化 OPC 对象的方法所需的局部变量。

```
void COpcSyncDlg::OnStart()
{
    HRESULT r1;
    CLSID clsid;
    LONG TimeBias = 0;
    FLOAT PercentDeadband = 0.0;
    DWORD RevisedUpdateRate;
    LPWSTR ErrorStr;
    char str[100];
    Cstring szErrorText;
```

然后执行下列操作：

### CoInitialize

*CoInitialize()* 初始化 COM 库。输入参数必须始终为 *NULL*。

```
r1 = CoInitialize(NULL);
if (r1 != S_OK)
{
    if (r1 == S_FALSE)
        MessageBox("COM Library already initialized",
                  "Error CoInitialize()", MB_OK+MB_ICONEXCLAMATION);
    else
        szErrorText.Format(
            "Initialisation of COM Library failed.\n
             Error Code= %4x", r1);
    MessageBox(szErrorText, "Error CoInitialize()", MB_OK+MB_ICONERROR);
    SendMessage(WM_CLOSE);
    return;
}
```

## 示例程序

### 6.2 C++ 中的 OPC 自定义接口（同步通信）

#### CLSIDFromProgID

*CLSIDFromProgID()* 将从指定的 ProgID 中检索全球唯一的类别标识符或 *CLSID*。  
*CoCreateInstance* 函数需要此方法。

此方法声明如下：

```
HRESULT CLSIDFromProgID(LPCOLESTR lpszProgID, LPCLSID pclsid);
```

参数	说明
lpszProgID	指向 ProgID [输入参数] 的指针
pclsid	指向检索的类别标识符 [返回值] 的指针

在示例程序中，该方法检索 SIMATIC NET 的 OPC 服务器的类别标识符：

```
r1 = CLSIDFromProgID(L"OPC.SimaticNET", &clsid);
if (r1 != S_OK)
{
    MessageBox("Retrieval of CLSID failed",
               "Error CLSIDFromProgID()", MB_OK+MB_ICONERROR);
    CoUninitialize();
    SendMessage(WM_CLOSE);
    return;
}
```

#### CoCreateInstance

*CoCreateInstance()* 将创建已指定类别标识符的类的实例。此方法声明如下：

```
STDAPI CoCreateInstance (REFCLSID rclsid,
                         LPUNKNOWN pUnkOuter,
                         DWORD dwClsContext,
                         REFIID riid,
                         LPVOID *ppv);
```

参数	说明
rclsid	所需对象的类别标识符
pUnkOuter	待创建的对象不是汇聚对象的一部分时的 NULL 指针。 并非 NULL 指针的被解释为汇聚对象的 <b>IUnknown</b> 接口的指针。

参数	说明
dwClsContext	描述待创建对象的运行环境。 该参数将指定创建和管理此类别的对象的可执行代码是否在本地机器上运行，以及是否会为其创建一个单独的过程。在 CLSCTX 枚举数据类型中指定可用于此过程的常量。 在示例中，使用了 CLSCTX_LOCAL_SERVER；换言之，用于管理服务器对象的可执行代码在与示例程序相同的计算机上运行（但是，以其自身的程序进行运行）。
riid	与对象进行通信的接口的标识符（在示例程序 <i>IOPCServer</i> 中）。
ppv	在成功调用该方法后包含所需接口指针的指针变量的地址。

以下方法调用将创建 OPC 服务器类的对象，并将指针返回到 *IOPCServer* 接口：

```
r1 = CoCreateInstance (clsid, NULL, CLSCTX_LOCAL_SERVER,
IID_IOPCServer, (void**)&m_pIOPCServer);
if (r1 != S_OK)
{
    MessageBox("Creation of IOPCServer-Object failed",
        "Error CoCreateInstance()",
        MB_OK+MB_ICONERROR);

    m_pIOPCServer = NULL;
    CoUninitialize();
    SendMessage(WM_CLOSE);
    return;
}
```

## AddGroup

*IOPCServer* 的 *AddGroup()* 方法将创建一个 OPC 组并声明如下：

```
HRESULT AddGroup (LPWCSTR szName,
                  BOOL bActive,
                  DWORD dwRequestedUpdateRate,
                  OPCHANDLE hClientGroup,
                  LONG *pTimeBias,
                  FLOAT *pPercentDeadband,
                  DWORD dwLCID,
                  OPCHANDLE *phServerGroup,
```

```

    DWORD *pRevisedUpdateRate,
    REFIID riid,
    LPUNKNOWN *ppUnk);

```

参数	说明
szName	可以由客户端自由分配但在客户端内必须唯一的组名称。
bActive	如果该组在创建时将处于未激活状态，则为 FALSE。 如果该组在创建时将处于激活状态，则为 TRUE。
dwRequestedUpdateRate	指定将数据项的值或状态的更改告知客户端后的最短时间间隔。 指定适合的时间间隔将防止信息被发送至客户端的速度超过客户端能够处理的速度。
hClientGroup	客户端可以自由选择的以及由服务器利用某些通知返回的代码编号。这样一来，客户端便可标识其数据。 客户端使用此句柄来标识组。
pTimeBias	服务器时间与 UTC（世界标准时间，Universal Time Convention）的偏差
pPercentDeadband	指定在发生值更改时不会导致通知的带宽的百分比。 必须知道值的上限值和下限值，才能够计算带宽。 在本示例程序中，情况并非如此。
dwLCID	选择返回文本时要由服务器使用的语言
phServerGroup	必须指定为带有某些函数调用的参数的、由服务器分配的句柄（例如， <i>RemoveGroup</i> ）。服务器需要它来标识该组。
pRevisedUpdateRate	将数据项值或状态的更改告知客户端后、由服务器返回的最短时间间隔。
riid	在创建组之后，指向 OPC 组对象接口之一的标识符的指针将可用。 此参数无需另行调用 <i>QueryInterface</i> 方法。
ppUnk	指向所需接口的指针。

在示例程序中调用 *AddGroup* 时，会进行以下设置：

- *bActive* 参数被设置为值 FALSE。在创建它之后，该组即会处于未激活状态；换言之，系统没有为该组生成任何 *OnDataChange* 回调。
- 所需的更新时间间隔的持续时间为 500 毫秒。
- 因为只有一个组正在被使用，所以可以指定任何客户端句柄。
- 在示例程序中，不会抑制有关特定范围内的值更改的通知。因此，为 *pPercentDeadband* 参数输入值“0.0”。
- *AddGroup* 将返回一个指向 *IOPCItemMgt* 接口的指针作为返回值。

```
r1 = m_pIOPCServer->AddGroup(L"grp1",
                                TRUE,
                                500,
                                1,
                                &TimBias,
                                &PercentDeadband,
                                LOCALE_ID,
                                &m_GrpSrvHandle,
                                &RevisedUpdateRate,
                                IID_IOPCItemMgt,
                                (LPUNKNOWN*)&m_pIOPCItemMgt);

if (r1 == OPC_S_UNSUPPORTEDRATE)
{
    szErrorText.Format ("Revised Update Rate %d is \
                        different from Requested Update Rate 500",
                        RevisedUpdateRate );
    AfxMessageBox(szErrorText);
}

else
{
    if (FAILED(r1))
        MessageBox("Can't add Group to Server!", "Error
                    AddGroup()", MB_OK+MB_ICONERROR);

    m_pIOPCServer->Release();
    m_pIOPCServer = NULL;
    CoUninitialize();
    SendMessage(WM_CLOSE);
}
```

```

    return;
}

```

## AddItems

*IOPCItemMgt* 接口的 *AddItems()* 方法将创建一个 OPC 数据项并声明如下：

```

HRESULT AddItems (DWORD dwNumItems,
                  OPCITEMDEF *pItemArray,
                  OPCITEMRESULT **ppAddResults,
                  HRESULT **ppErrors);

```

参数	说明
<b>dwNumItems</b>	待插入的数据项数
<b>pItemArray</b>	具有 OPCITEMDEF 类型的元素的数组。 此类型的结构变量包含由服务器创建数据项时所需的全部信息。
<b>ppAddResults</b>	具有 OPCITEMRESULT 类型的元素的数组。由 OPC 服务器发送的返回值是此类型的结构变量。
<b>ppErrors</b>	具有 HRESULT 类型元素的数组。 这些变量将返回无法成功创建数据项时的错误代码，或返回有关成功的方法调用的信息。

在调用 *AddItems* 之前，必须利用有效值创建并初始化一个具有 OPCITEMDEF 类型的元素的数组。进行此操作时，必须考虑以下几种情况：

- 该示例中数据项的访问路径并非必不可少，并且空字符串将会在此指定。
- 在模块 *OPCDA\_SyncDlg.cpp* 的开始处指定数据项标识符，并将其分配到变量 **szItemID**。
- 数据项将在创建后处于激活状态。
- 该示例程序使用客户端句柄 1。
- SIMATIC NET 的 OPC 服务器不需要 *BinaryLargeObjects*，因此结构组件 **dwBlobSize** 的值为“0”。
- 对于数据项，服务器必须在对应于数据项原始数据类型的类型中发送返回值。

```

m_Items[0].szAccessPath = L"";
m_Items[0].szItemID = szItemID;
m_Items[0].bActive = TRUE;

```

```

m_Items[0].hClient = 1;
m_Items[0].dwBlobSize = 0;
m_Items[0].pBlob = NULL;
m_Items[0].vtRequestedDataType = vtDataTypeItem;

```

*m\_pItemResult* 指针作为 *COPCDA\_SyncDlg* 类的一个属性而存在。

由可以使用此变量访问的服务器返回的结果。*m\_pErrors* 是一个指向错误代码的指针：

```
r1 = m_pIOPCItemMgt->AddItems(1, m_Items, &m_pItemResult, &m_pErrors);
```

如果 *AddItems* 调用未成功，将中止程序。但是，程序首先必须释放其正在使用的资源：

```

if ( (r1 != S_OK) && (r1 != S_FALSE) )
{
    MessageBox("AddItems failed!", "Error AddItems()",
               MB_OK+MB_ICONERROR);

    m_pIOPCItemMgt->Release();

    m_pIOPCItemMgt = NULL;

    m_GrpSrvHandle = NULL;

    m_pIOPCServer->Release();

    m_pIOPCServer = NULL;

    CoUninitialize();

    SendMessage(WM_CLOSE);

    return;
}

```

## GetErrorString

如果 *AddItems()* 的返回值指示已出现错误，*IOPCServer* 接口的 *GetErrorString()* 方法将会获取相应的错误消息。此方法声明如下：

```

HRESULT GetErrorString (HRESULT dwError,
                       LCID dwLocale,
                       LPWSTR *ppString);

```

参数	说明
<i>dwError</i>	由服务器返回的错误代码
<i>dwLocale</i>	错误消息的语言标识符
<i>ppString</i>	指向空终止字符串（ <i>GetErrorString</i> 将错误消息返回到此字符串）的双重指针

```

    else
    {
        m_pIOPCServer ->GetErrorString(m_pErrors[0], LOCALE_ID,
                                         &ErrorStr);

        sprintf(str, "%ls\n", ErrorStr);
        MessageBox(str, "Result AddItems()", MB_OK+MB_ICONEXCLAMATION);
        CoTaskMemFree(ErrorStr);
    }
}

```

## QueryInterface

### *IUnknown* 接口的 QueryInterface()

方法将返回一个指向其标识符被指定为输入参数的接口的指针。*QueryInterface()* 声明如下：

```
HRESULT QueryInterface (REFIID iid, void **ppvObject);
```

参数	说明
iid	所需接口的标识符
ppvObject	在成功调用该方法后包含所需接口指针的指针变量的地址。 如果对象不支持此接口，将返回错误代码和 NULL 指针。

### *QueryInterface* 获得一个指向 *IOPCSyncIO*

接口（此接口提供用于同步读写的方法）的指针：

```

rl = m_pIOPCItemMgt->QueryInterface(IID_IOPCSyncIO,
                                         (void**)&m_pIOPCSyncIO);

if (rl < 0)
{
    MessageBox("No IOPCSyncIO found!",
              "Error QueryInterface()", MB_OK+MB_ICONERROR);

    CoTaskMemFree(m_pItemResult);

    m_pIOPCItemMgt->Release();
    m_pIOPCItemMgt = NULL;
    m_GrpSrvHandle = NULL;
    m_pIOPCServer->Release();
    m_pIOPCServer = NULL;
    CoUninitialize();
}

```

```

    SendMessage (WM_CLOSE) ;
    return;
}

```

## 启用按钮

在 *OnButtonStart()* 已设置所有必要的 OPC 对象后，其即会禁用“启动示例”(Start Sample) 按钮。所有其它按钮都将被启用。此策略确保 *OnButtonStart()* 只执行一次。这样就无需程序中的其它查询。

```

m_CtrlStop.EnableWindow (TRUE) ;
m_CtrlRead.EnableWindow (TRUE) ;
m_CtrlWrite.EnableWindow (TRUE) ;
m_CtrlStart.EnableWindow (FALSE) ;

```

## OPCITEMDEF 结构

OPCITEMDEF 具有下列结构：

```

typedef struct {
    LPWSTR szAccessPath;
    LPWSTR szItemID;
    BOOL bActive;
    OPCHANDLE hClient;
    DWORD dwBlobSize;
    BYTE *pBlob;
    VARTYP vtRequestedDataType;
    Word wReserved;
} OPCITEMDEF;

```

## OPCITEMDEF 的变量

变量	说明
szAccessPath	数据项的可选访问路径 ( <b>SIMATIC NET</b> 不需要此路径)
szItemID	由客户端分配的 ItemID
bActive	如果将激活组中的数据项值是否发生更改告知客户端，则为 TRUE； 如果没有告知客户端，则为 FALSE。

变量	说明
hClient	由客户端分配的数据项的句柄。 服务器通过客户端调用传送客户端句柄（例如， <i>OnDataChange</i> ），以便客户端能够访问其结构中的相关变量。
dwBlobSize	服务器上存储区（此存储区用于存储有关更快地访问数据项数据的其它信息）的大小。
pBlob	指向上述存储区的指针
vtRequestedDataType	客户端请求的数据类型

## OPCITEMRESULT 结构

OPCITEMRESULT 具有下列结构：

```
typedef struct {
    OPCHANDLE hServer;
    VARTYPE vtCanonicalDataType;
    WORD wReserved;
    DWORD dwAccessRights;
    DWORD dwBlobSize;
    BYTE *pBlob;
} OPCITEMRESULT;
```

## OPCITEMRESULT 的变量

变量	说明
hServer	由服务器分配的数据项的句柄。 客户端通过服务器调用传送服务器句柄，以便服务器能够访问其结构中的相关变量。
vtCanonicalDataType	由服务器使用的数据项的数据类型
dwAccessRights	有关数据项是只能读取、只能写入还是可读写的信息。
dwBlobSize	服务器上存储区（此存储区用于存储有关更快地访问数据项数据的其它信息）的大小。
pBlob	指向上述存储区的指针

### 6.2.4.3 OnRead

*OnRead* 执行同步读取作业。可以执行下列操作：

#### 变量声明

首先，声明几个局部变量：

```
void COpcSyncDlg::OnRead()
{
    OPCHANDLE *phServer;
    OPCITEMSTATE *pItemValue;
    HRESULT *pErrors;
    HRESULT r1;
    LPWSTR ErrorStr;
    char str[100];
    UINT qnr;
```

将要读取值的数据项的服务器句柄需要作为 *Read()* 方法的参数。

由服务器分配的标识号存在于 *pItemResult[0]* 全局变量的 *hServer* 组件中。*AddItem()* 方法在类型 *OPCITEMRESULT* 的结构变量中输入其返回值。

```
phServer = new OPCHANDLE[1];
phServer[0] = m_pItemResult[0].hServer;
```

#### Read

*Read* 同步读取 OPC 数据项的值。此方法声明如下：

```
HRESULT Read (OPCDATASOURCE dwSource,
               DWORD dwNumItems,
               OPCHANDLE *phServer,
               OPCITEMSTATE **ppItemValues,
               HRESULT **ppErrors);
```

参数	说明
<b>dwSource</b>	数据源。如果使用 <b>OPC_DS_CACHE</b> ，数据将从 OPC 服务器缓存中进行读取；使用 <b>OPC_DS_DEVICE</b> ，读取作业将通过网络加以执行。
<b>dwNumItems</b>	读取值的数据项的数目。
<b>phServer</b>	具有服务器句柄的数组。

参数	说明
pplItemValues	具有类型 OPCITEMSTATE 的元素的数组，用于读取值和有关读取作业的其它信息。
ppErrors	具有 HRESULT 类型元素的数组。这些变量将返回无法成功调用 <i>Read()</i> 时的错误代码，或返回有关成功的方法调用的信息。

*Read* 将被作为参数使用先前初始化的变量进行调用：

```
r1 = m_pIOPCSyncIO->Read(OPC_DS_DEVICE, 1, phServer, &pItemValue,
                           &pErrors);
```

*Read* 方法提供已读取的值及 *pItemValue* 数组中的其它信息。

如果读取成功，这些数据会显示在主对话框的相关文本框中：

```
if (r1 == S_OK)
{
    m_ReadValue = pItemValue[0].vDataValue.lVal;
    qnr = pItemValue[0].wQuality;
    m_szReadQuality = GetQualityText(qnr);
    m_szTimeStamp =
        COleDateTime(pItemValue[0].ftTimeStamp).Format();
    UpdateData(FALSE);
}
```

返回值 *S\_FALSE* 表示一个错误代码已被输入到 *pErrors* 数组中。*GetErrorString* 获得在对话框中显示的相应错误消息：

```
if (r1 == S_FALSE)
{
    m_pIOPCServer->GetErrorString(pErrors[0], LOCALE_ID, &ErrorStr);
    sprintf(str, "%S\n", ErrorStr);
    MessageBox(str, "Error Read()", MB_OK+MB_ICONERROR);
    CoTaskMemFree(ErrorStr);
}
```

如果没有错误消息并且方法调用未成功，则会显示一个适合的对话框。最后，释放 *Read* 所使用的资源：

```
if (FAILED(r1))
{
    MessageBox("Read failed!", "Error Read()", MB_OK+MB_ICONERROR);
}
```

```

    else
    {
        CoTaskMemFree(pErrors);
        CoTaskMemFree(pItemValue);
    }
}

```

## OPCITEMSTATE

```

typedef struct {
    OPCHANDLE hClient;
    FILETIME ftTimeStamp;
    WORD wQuality;
    WORD wReserved;
    VARIANT vDataValue;
} OPCITEMSTATE;

```

### OPCITEMSTATE 的变量

变量	说明
hClient	数据项的客户端句柄。
ftTimeStamp	时间戳（接收来自 OPC 服务器的数据的时间）。
wQuality	有关数据完整性的信息。
vDataValue	为数据项读取的值。

#### 6.2.4.4 OnWrite

OnWrite 可执行同步写入作业。可以执行下列操作：

### 变量声明

首先，声明几个局部变量：

```

void COpcSyncDlg::OnWrite()
{
    OPCHANDLE *phServer;
    HRESULT *pErrors;
    VARIANT values[1];
    HRESULT r1;
    LPWSTR ErrorStr;
    CString szOut;
}

```

将被写入值并被要求作为 *Write* 方法的参数的数据项服务器句柄。

由服务器分配的标识号存在于 *m\_pItemResult[0]* 全局变量的 *hServer* 组件中。 *AddItems* 方法在类型 **OPCITEMRESULT** 的结构变量中输入其返回值：

```
phServer = new OPCHANDLE[1];
phServer[0] = m_pItemResult[0].hServer;
```

*values* 变量用于存储将写入的值。 *CWnd* 类的 *UpdateData* 方法将所有控件的内容传送到相关的成员变量。 这将初始化 *values* 结构变量的 *iVal* 组件。*2byte integer* 被指定为数据类型：

```
UpdateData(TRUE);
values[0].vt = VT_I2;
values[0].iVal = m_WriteValue;
```

## Write

*Write* 为 OPC 数据项同步写入值。 此方法声明如下：

```
HRESULT Write (DWORD dwNumItems,
               OPCHANDLE *phServer,
               VARIANT *pItemValues,
               HRESULT **ppErrors);
```

参数	说明
<i>dwNumItems</i>	写入值的数据项数。
<i>phServer</i>	具有服务器句柄的数组
<i>pItemValues</i>	具有待写入值的数组
<i>ppErrors</i>	具有 <b>HRESULT</b> 类型元素的数组。 这些变量将返回无法成功调用 <i>Write()</i> 时的错误代码，或返回有关成功的方法调用的信息。

*Write* 将被作为参数使用先前初始化的变量进行调用：

```
r1 = m_pIOPCSyncIO->Write(1, phServer, values, &pErrors);
```

*GetErrorString* 获取属于已分配给成员变量 *m\_WriteRes* 的 *pErrors* 返回值的错误消息。但是，*GetErrorString*

增加了两个特殊字符（换行符），而这些字符必须先予以删除，然后 *GetErrorString* 所返回的内容才能显示在文本框中。

最后，释放 *Write* 所使用的资源：

```

delete [] phServer;
if (FAILED(r1))
{
    szOut.Format("Method call IOPCSyncIO::Write \
failed with error code %x", r1);
    MessageBox(szOut, "Error Writing Item",
    MB_OK+MB_ICONERROR);
}
else
{
    m_pIOPCServer->GetErrorString(pErrors[0], LOCALE_ID,
    &ErrorStr);
    m_szWriteResult = ErrorStr;
    m_szWriteResult.Remove('\r');
    m_szWriteResult.Remove('\n');
    UpdateData(FALSE);
    CoTaskMemFree(pErrors);
    CoTaskMemFree(ErrorStr);
}

```

#### 6.2.4.5 OnStop

*OnStop* 可移除程序中使用的 OPC 对象并释放相应资源。单击“停止示例”(Stop Sample)按钮后、执行 *OnDestroy* 方法或发送 WM\_CLOSE 消息后（在单击按钮关闭对话框后或明确调用 *SendMessage* 后）会调用该方法。

以多个步骤释放资源：

- RemoveItems
- RemoveGroup
- Release
- CoUninitialize

#### RemoveItems

IOPCItemMgt 接口的 *RemoveItems* 方法可移除 OPC 数据项并按如下方式声明：

## 示例程序

### 6.2 C++ 中的 OPC 自定义接口 (同步通信)

```
HRESULT RemoveItems (DWORD dwCount,
                      OPCHANDLE *phServer,
                      HRESULT **ppErrors);
```

参数	说明
dwCount	待移除的数据项数目
phServer	含有待移除数据项的服务器句柄的数组
ppErrors	具有 HRESULT 类型元素的数组。这些变量将返回未成功调用 <i>RemoveItems()</i> 时的错误代码或返回有关方法成功调用的信息。

```
void COpcSyncDlg::OnStop()
{
    HRESULT r1;
    OPCHANDLE *phServer;
    HRESULT *pErrors;
    LPWSTR ErrorStr;
    char str[100];

    phServer = new OPCHANDLE[1];
    phServer[0] = m_pItemResult[0].hServer;
    r1 = m_pIOPCItemMgt->RemoveItems(1, phServer, &pErrors);
    if ( (r1 != S_OK) && (r1 != S_FALSE) )
    {
        MessageBox("RemoveItems failed!",
                  "Error RemoveItems()", MB_OK+MB_ICONERROR);
    }
    else
    {
        m_pTOPCServer->GetErrorString(pErrors[0], LOCALE_ID,
                                         &ErrorStr);
        sprintf(str, "%ls\n", ErrorStr);
        MessageBox(str, "Result RemoveItems()", MB_OK+MB_ICONEXCLAMATION);
        CoTaskMemFree(ErrorStr);
    }
}
```

## RemoveGroup

*RemoveGroup()* 可用于移除服务器中的组并按如下方式声明：

```
HRESULT RemoveGroup (OPCHANDLE hServerGroup,
                     BOOL bForce);
```

参数	说明
hServerGroup	待移除的组的服务器句柄。
bForce	指定当组的引用仍然存在时是否能够移除该组。

```
r1=m_pIOPCServer->RemoveGroup(m_GrpSrvHandle, TRUE);
if (r1 != S_OK)
{
    MessageBox("RemoveGroup failed!",
               "Error Remove Group ()",
               MB_OK+MB_ICONERROR);
}
```

## Release

*IOPCServer* 和 *IMalloc* 接口有用于释放接口所使用资源的 *Release* 方法:

```
m_pIOPCSyncIO->Release();
m_pIOPCSyncIO = NULL;
m_pIOPCIItemMgt->Release();
m_pIOPCIItemMgt = NULL;
m_pIOPCServer->Release();
m_pIOPCServer = NULL;
```

除此之外，必须再次释放所有其它请求的资源:

```
delete[] phServer;
CoTaskMemFree(pErrors);
CoTaskMemFree(m_pItemResult);
m_pItemResult=NULL;
CoTaskMemFree(m_pErrors);
m_pErrors = NULL;
m_GrpSrvHandle = NULL;
```

除了“启动示例”(Start Sample) 之外的所有按钮都处于未激活状态:

```
m_CtrlStart.EnableWindow(TRUE);
m_CtrlRead.EnableWindow(FALSE);
m_CtrlStop.EnableWindow(FALSE);
m_CtrlWrite.EnableWindow(FALSE);
```

## CoUninitialize

*CoUninitialize* 方法可关闭 COM 库并释放线程的所有资源：

```
CoUninitialize();  
}
```

### 6.2.4.6 DestroyWindow

*CWnd* 类的此方法将被覆盖以便对话框窗口关闭时可以执行所有的“清除作业”。

*OnButtonStop()* 将删除所有的 OPC 对象并释放相应的资源。此后，将调用父类的 *DestroyWindow* 来释放窗口对象使用的存储器：

```
BOOL COPCDA_SyncDlg::DestroyWindow()  
{    if (m_pIOPCServer)  
    {        OnStop();  
        return CDialog::DestroyWindow();  
    }  
}
```

### 6.2.4.7 GetQualityText

*GetQualityText* 提供了一个 *CString* 对象作为指定错误代码的错误消息：

```
CString GetQualityText(UINT qnr)  
{    CString qstr;  
    switch(qnr)  
    {        case OPC_QUALITY_BAD:  
            qstr = "BAD";  
            break;  
        case OPC_QUALITY_UNCERTAIN:  
            qstr = "UNCERTAIN";  
            break;  
        case OPC_QUALITY_GOOD:  
            qstr = "GOOD";  
            break;  
        case OPC_QUALITY_NOT_CONNECTED:  
            qstr = "NOT_CONNECTED";  
            break;  
        case OPC_QUALITY_DEVICE_FAILURE:  
            qstr = "DEVICE_FAILURE";  
            break;  
        case OPC_QUALITY_SENSOR_FAILURE:  
            qstr = "SENSOR_FAILURE";  
            break;  
    }  
    return qstr;  
}
```

```
    qstr = "SENSOR_FAILURE";
    break;

    case OPC_QUALITY_LAST_KNOWN:
        qstr = "LAST_KNOWN";
        break;

    case OPC_QUALITY_COMM_FAILURE:
        qstr = "COMM_FAILURE";
        break;

    case OPC_QUALITY_OUT_OF_SERVICE:
        qstr = "OUT_OF_SERVICE";
        break;

    case OPC_QUALITY_LAST_USABLE:
        qstr = "LAST_USABLE";
        break;

    case OPC_QUALITY_SENSOR_CAL:
        qstr = "SENSOR_CAL";
        break;

    case OPC_QUALITY_EGU_EXCEEDED:
        qstr = "EGU_EXCEEDED";
        break;

    case OPC_QUALITY_SUB_NORMAL:
        qstr = "SUB_NORMAL";
        break;

    case OPC_QUALITY_LOCAL_OVERRIDE:
        qstr = "LOCAL_OVERRIDE";
        break;

    default: qstr = "UNKNOWN ERROR";
}

return qstr;
}
```

## 6.2.5 有关编写您自己的程序的注意事项

必须满足几项要求，您自己的程序才能使用自定义接口。请按照下面列出的步骤进行操作：

1. 启动 Visual C++ 开发环境。
2. 使用 MFC 类向导创建所需类型的项目。

3. 从示例程序中将文件“pre\_opc.h”和“pre\_opc.cpp”复制到项目目录中，并将文件添加至该项目（“项目 > 添加至项目 > 文件...”(Project > Add to project > Files...)）。  
这些文件包含 OPC 特定的定义和 include 语句。
4. 将下列语句添加至项目的所有实现文件中（扩展名为“\*.cpp”）： `#include pre_opc.h`
5. 在使用 `AddGroup()` 或 `GetErrorString()` 方法的实现文件中，添加下列语句： `#define LOCALE_ID0x409`
6. 在主对话框中排列所需操作员界面元素，或创建合适的应用程序窗口并对相应的事件过程进行编程。

## 6.3 C++ 中的 OPC 自定义接口（异步通信）

该示例使用 OPC 的数据访问 V2.0 的自定义接口。

### 6.3.1 激活仿真连接

运行此程序前，必须激活仿真连接以使程序中所使用的演示变量可用。请按照下面列出的步骤进行操作：

1. 从“开始”(Start) 菜单中启动“通信设置”(Communication Setting) 程序：

“开始 > 所有程序 > Siemens Automation > SIMATIC > SIMATIC NET > 组态控制台”(Start > All Programs > Siemens Automation > SIMATIC > SIMATIC NET > Configuration Console)。

响应： 打开“通信设置”(Communication Settings) 组态程序。

2. 在左侧的导航窗口中，打开条目“OPC 协议选择”(OPC protocol selection)：

“PC 站 > 应用程序 > OPC 设置 > OPC 协议选择”(PC Station > Applications > OPC Settings > OPC Protocol Selection)。

3. 启用待仿真协议的复选框。

本例使用 S7 协议。

因此，选择“名称： S7”(Name: S7) 下的复选框并单击“更改详细信息...”(Change details...)。

响应： 打开“协议详细信息”(Protocol details) 窗口。

4. 选中“为仿真提供虚拟模块 (DEMO)”(Provide virtual module (DEMO) for the simulation) 复选框。

5. 单击“应用”(Apply) 确认。
6. 关闭“通信设置”(Communication Settings) 程序。

#### 说明

在更改生效之前，必须首先关闭所有 OPC 客户端！

### 6.3.2 使用示例程序

该程序包括多个元素，操作员可使用这些元素触发以下动作：

操作员元素	效果
启动示例	启动程序
读取或写入数据项 0	读取和写入数值
组激活	激活或禁用组
停止示例	停止程序

### 6.3.3 启动程序

该程序位于硬盘上的：“<installationpath>\SIEMENS\SIMATIC.NET\opc2\samples\custom\async”中。

启动程序时，仅启用“启动示例”(Start Sample) 按钮。单击此按钮后，程序将生成所需的 OPC 对象。然后，便可使用其它按钮。

### 6.3.4 读取和写入数值

单击“读取数据项 0”(Read Item 0)

按钮后，所读取的值以及状态信息会显示在相关的文本框中。

按照实际情况来说，示例程序专用于通过演示连接进行操作。

如果要在真实环境下运行示例程序，则必须更改 OPCDA\_AsyncDlg.cpp 程序中的以下行：

```
const LPWSTR szItemID0 = L"S7:[DEMO]MB1";
const LPWSTR szItemID1 = L"S7:[DEMO]MW1";
```

### 6.3 C++ 中的 OPC 自定义接口 (异步通信)

若要写入值，可在“值”(Value) 文本框中输入值并单击“写入值”(Write Value) 按钮。程序会显示有关写入操作结果的消息。

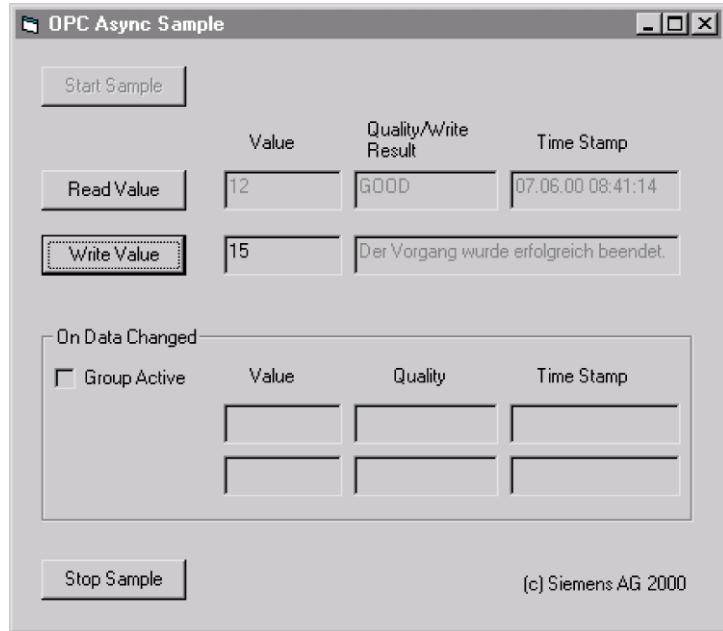


图 6-13 单击“写入值”(Write Value) 按钮后出现的对话框

#### 6.3.5 激活组

异步操作不是直接返回结果（例如：作为函数结果或返回参数）而是利用事件。

可通过选择“组激活”(Group Active) 复选框激活示例程序中的 OPC 组。此外，还会生成 *OnDataChange* 事件。在此事件出现时执行的步骤会在“关于已更改数据”(On Data Changed) 文本框中显示值和状态信息。

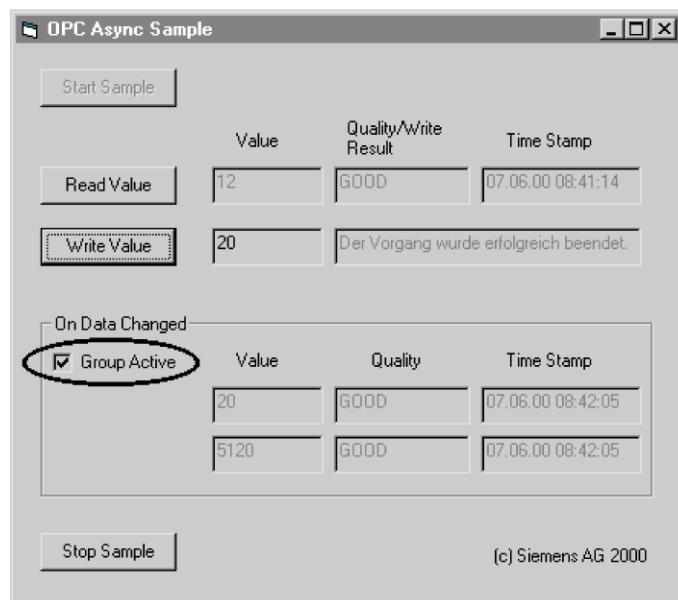


图 6-14 通过“组激活”(Group Active) 复选框启用 OPC 组后，在“关于已更改数据”(On Data Changed) 文本框中显示值和状态信息

未激活组也会生成用于完成写入或读取操作的事件。  
这表示，即使未选择该复选框也能显示写入和读取的结果。

### 6.3.6 停止程序

#### “停止示例”(Stop Sample)

按钮将关闭程序：换言之，所有对象将被删除，并且相应资源会被再次释放。

### 6.3.7 如何运行程序？

由于 OPC 的类模型，在调用 OPC 对象的方法时必须遵守特定的顺序。为了能够创建 *OPCItem* 类的实例，*OPCGroup* 类的对象必不可少。然而，只有在 *OPCServer* 类的实例已存在并已建立与该服务器的连接时，才能实现该操作。

下图阐述了创建和删除 OPC 对象的指令的基本顺序。使用示例程序的变量名称：

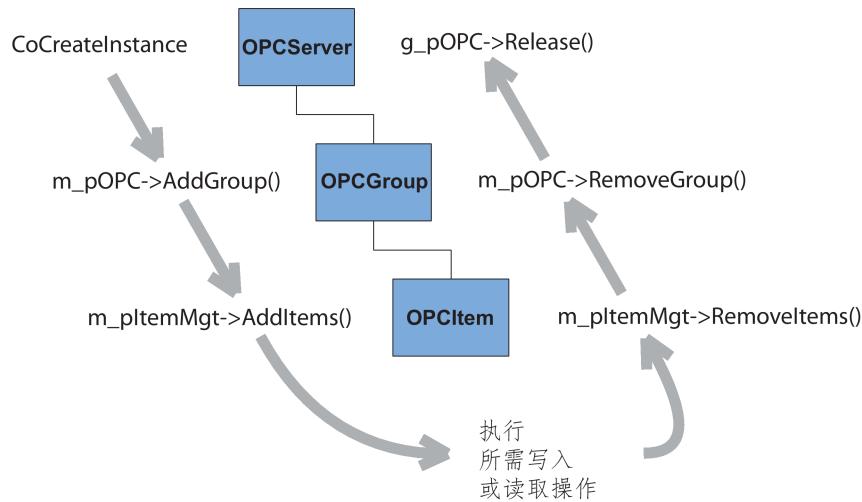


图 6-15 创建和删除 OPC 对象的命令的顺序

示例程序包含通常情况下存在于典型客户端应用程序中的所有组件。其中包括建立与 OPC 服务器的连接、创建变量组、读取和写入数据项值。

在详细说明源代码之前，首先要了解 OPC 应用程序的基本结构。

步骤	说明
1	注册 COM
2	将 ProgID 转换为 CLSID
3	建立与 OPC 服务器的连接
4	创建 OPC 组
5	添加数据项
6	请求 IOPCGroupStateMgt 的接口指针
7	请求 IOPCAsyncIO2 的接口指针
8	创建回调对象
9	连接 OPC 服务器与客户端的回调对象
10	执行所需的写入和读取操作
11	接收 OPC 服务器的通知
12	删除对象并释放内存

## 步骤 1：注册 COM

想要使用 COM 库函数的每个程序必须首先自行注册 COM。此为 `CoInitialize` 函数的用途：

```
HRESULT r1;
r1 = CoInitialize(NULL);
```

## 步骤 2：将 ProgID 转换为 CLSID

为了对其进行标识，每个 COM 服务器都有一个分配给全球唯一 *CLSID* 的 *ProgID*。使用 `CLSIDFromProgID()` 函数可实现上述操作。SIMATIC NET 的 OPC 服务器的 *ProgID* 是 L"OPC.SimaticNET"：

```
r1 = CLSIDFromProgID(L"OPC.SimaticNET", &clsid);
```

## 步骤 3：建立与 OPC 服务器的连接

`CoCreateInstance()` 函数将创建已指定 *CLSID* 的类的实例。

```
r1 = CoCreateInstance(clsid, NULL, CLSCTX_LOCAL_SERVER,
                      IID_IOPCServer, (void**)&m_pIOPCServer);
```

此程序段的结果是 OPC 服务器类的对象。`CoCreateInstance` 还会提供指向服务器对象的 *IOPCServer* 接口的指针 (*m\_pIOPCServer* 变量)：

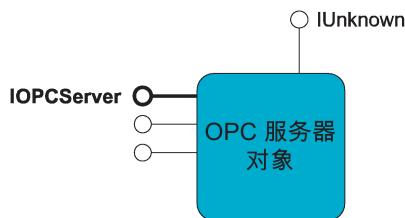


图 6-16 具有指向 *IOPCServer* 接口的指针的 OPC 服务器类的对象

## 步骤 4：创建 OPC 组

*IOPCServer* 接口具有用于创建组的 `AddGroup()` 方法：

```
HRESULT r1;
r1 = m_pIOPCServer ->AddGroup(L"grp1", FALSE, 500, 1,
                                &TimBias, &PercentDeadband, LOCALE_ID, &m_GrpSrvHandle,
                                &RevisedUpdateRate, IID_IOPCIItemMgt,
                                (LPUNKKNOWN*) &m_pIOPCIItemMgt);
```

## 6.3 C++ 中的 OPC 自定义接口（异步通信）

该程序段的结果是一个具有指定名称和所需属性的组。 *AddGroup* 还会将指向组对象的所请求接口的指针作为返回参数进行返回，在这种情况下，会返回 *IOPCItemMgt* (*m\_pIOPCItemMgt* 变量)：

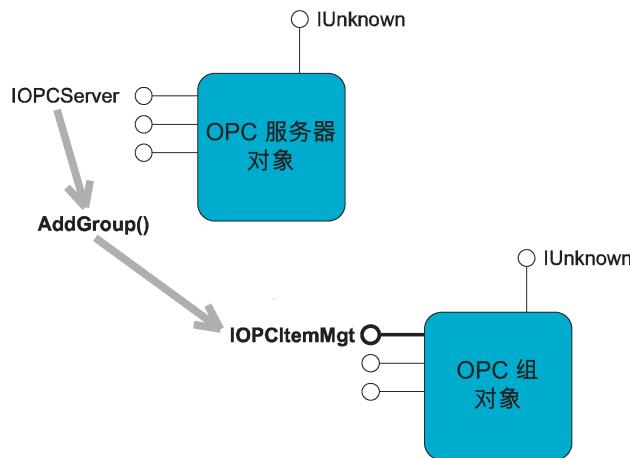


图 6-17 在组对象的请求的 *IOPCItemMgt* 接口上生成 OPC 组

## 步骤 5：添加数据项

*IOPCItemMgt* 接口具有用于创建 OPC 数据项的 *AddItems()* 方法：

```

HRESULT r1;
r1 = m_pIOPCItemMgt->AddItems(2, m_Items, &m_pItemResult,
                                &m_pErrors);
    
```

该程序段的结果是服务器添加两个具有 *m\_Items* 参数中所指定属性的数据项。

此外，还会对结果结构的变量

*m\_pItemResult* (服务器句柄、目标系统上的数据项类型，等等) 进行初始化。

步骤 6：请求 *IOPCGroupStateMgt* 的接口指针

要使用 *SetState* 方法就必须有指向 *IOPCGroupStateMgt* 接口的指针。稍后需使用 *SetState* 方法激活和禁用组。还需要该接口指针作为 *AtAdvise* 和 *AtUnadvise* 方法的参数：

```

HRESULT r1;
r1 = m_pIOPCItemMgt->QueryInterface(IID_IOPCGroupStateMgt,
                                         (void**)&m_pIOPCGroupStateMgt);
    
```

该程序段的结果是指向组对象的 *IOPCGroupStateMgt* 接口的指针 (*m\_pIOPCGroupStateMgt* 变量)：

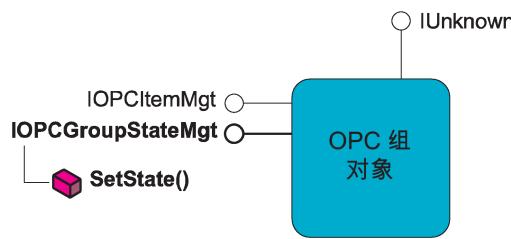


图 6-18 具有指向 *IOPCGroupStateMgt* 接口的指针的 OPC 组对象及其 *SetState()* 方法

### 步骤 7：请求 *IOPCAsyncIO2* 的接口指针

同样，该程序也会请求指向 *IOPCAsyncIO2* 接口的指针（*m\_pIOPCAsyncIO2* 变量）。该接口提供了用于异步读取和写入值的方法：

```
r1 = m_pIOPCItemMgt ->QueryInterface(IID_IOPCAsyncIO2, (void**)&m_pIOPCAsyncIO2);
```

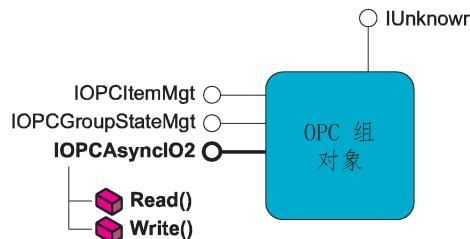


图 6-19 具有指向 *IOPCAsyncIO2* 接口的指针的 OPC 组对象及其用于异步读取和写入值的方法

### 步骤 8：创建回调对象

为了允许 OPC 服务器返回异步操作的值，必须在客户端上执行 *IOPCDataCallback* 接口。在示例程序中，该操作由源于 *IOPCDataCallback* 和 *CComObjectRoot* 的 *COPCDataCallback* 类处理。

*CreateInstance* 方法将创建 *CComObject* 模板类的对象：

```
CComObject<COPCDataCallback>* pCOPCDataCallback;
CComObject<COPCDataCallback>::CreateInstance
&pCOPCDataCallback;
```

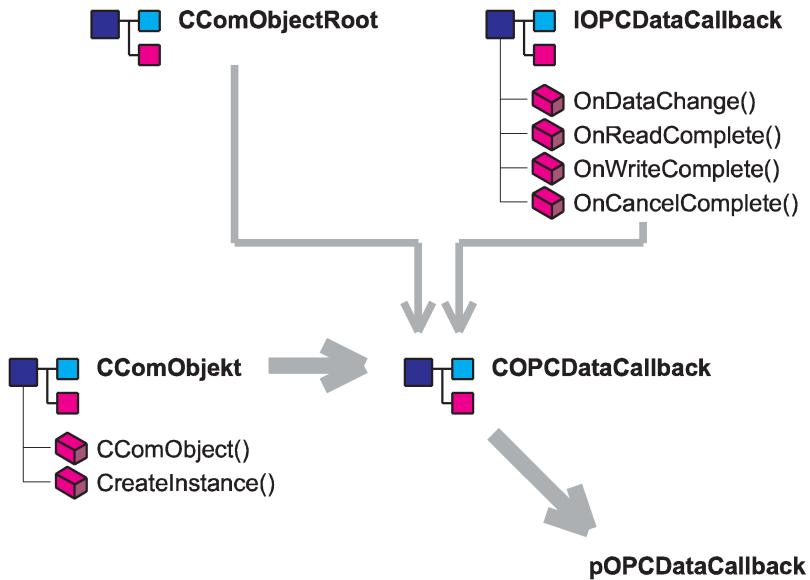


图 6-20 客户端的 *IOPCDataCallback* 接口; *CreateInstance* 方法将生成模板类对象 *CComObject*

### 步骤 9: 连接 OPC 服务器与客户端的回调对象

*AtAdvise()* 方法将创建 OPC 服务器与回调对象之间的连接。*GetUnknown()* 方法将首先获取指向回调对象的 *IUnknown* 接口的指针: *AtAdvise* 需要该指针作为输入参数:

```

LPUNKNOWN pCbUnk;
pCbUnk = pCOPCDataCallback->GetUnknown();
HRESULT hRes = AtAdvise(m_pIOPCGroupStateMgt, pCbUnk,
                        IID_IOPCDataCallback, &m_dwAdvise);

```

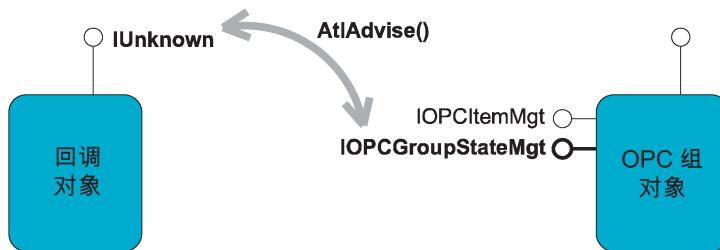


图 6-21 使用 *AtAdvise()* 方法创建 OPC 服务器与回调对象之间的连接

## 步骤 10：执行所需的写入和读取操作

可通过在步骤 7 中创建的 `m_pIOPCAsyncIO2` 指针访问 `IOPCAsyncIO2` 接口的 `Read()` 和 `Write()` 方法：

```
r1 = m_pIOPCAsyncIO2->Read(1, phServer, 1, &dwCancelID,
                             &pErrors);
```

## 步骤 11：接收 OPC 服务器的通知

如果激活组中激活数据项的数据发生了更改，则服务器将调用回调对象的 `OnDataChange` 方法。完成读取操作后，服务器会调用 `OnReadComplete` 等方法。服务器会将返回值作为参数传送给相关方法。

```
STDMETHODIMP COPCDataCallback::OnDataChange(
    WORD dwTransid,
    OPCHANDLE hGroup,
    HRESULT hrMasterquality,
    RESULT hrMastererror,
    DWORD dwCount,
    OPCHANDLE __RPC_FAR *phClientItems,
    VARIANT __RPC_FAR *pvValues,
    WORD __RPC_FAR *pwQualities,
    FILETIME __RPC_FAR *pftTimeStamps,
    HRESULT __RPC_FAR *pErrors)
```

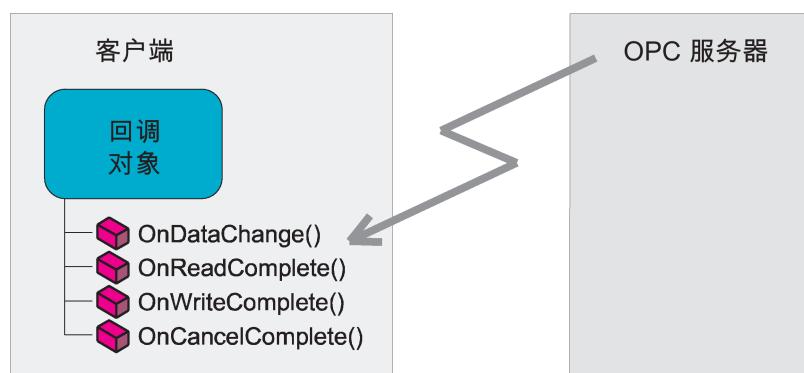


图 6-22 OPC 服务器调用回调对象的方法（此处为：`OnDataChange()`）

## 步骤 12：删除对象并释放内存

退出程序之前，必须删除已创建的 OPC 对象并释放为其保留的内存。

相关的函数是先前使用的接口的重要组成部分。*AtlUnadvise* 将终止 OPC 服务器与回调对象之间的连接：

```
HRESULT hRes = AtlUnadvise(m_pIOPCGroupStateMgt,
                            IID_IOPCDataCallback, m_dwAdvise);
m_pIOPCGroupStateMgt->Release();
...
r1 = m_pIOPCItemMgt->RemoveItems(2, phServer,
                                    &pErrors);
...
CoTaskMemFree(pErrors);
CoTaskMemFree(m_pItemResult);
m_pItemResult=NULL;
...
...
```

### 释放内存

使用 COM 时，客户端有时需要释放服务器所请求的内存。

相关规则如下：

- **[out]**: 将从服务器请求具有该属性的参数的内存。
- **[in, out]**: 将从客户端请求此类参数的内存。  
服务器可以再次释放内存并对其进行重新分配。
- **[in]**: 客户端将请求内存。服务器不负责释放内存。

在这三种情况下，客户端都将有效地释放所保留的内存。

## 6.3.8 程序说明

本说明将解释以下两个文件的内容：

- “OPCDA\_AsyncDlg.cpp”是程序特定对话框类的实现文件。  
在此，可找到有关各个按钮的事件过程。它们的主体可能已由 MFC 类向导创建。  
在此也定义了初始化对话框和显示数据的方法。
- “Callback.cpp”是程序特定回调类的实现文件。  
在本模块中定义了客户端必须提供的所有方法，这些方法可便于客户端接收来自 OPC 服务器的通知。

### 6.3.9 “OPCDA\_AsyncDlg.cpp”文件

“OPCDA\_AsyncDlg.cpp”文件可以分为以下两部分：

- 全局变量的声明
- *CAboutDlg* 类
- *COPCDA\_AsyncDlg.cpp* 类的方法

#### 用于存储 **ItemID** 的全局变量的声明

在该模块的开始部分，使用数据项标识符对这两个变量进行初始化：

```
const LPWSTR szItemID0 = L"S7:[DEMO]MB1";
const LPWSTR szItemID1 = L"S7:[DEMO]MW1";
```

对于程序功能而言，可读取和写入第一个变量是很重要的，但实际上，可读取第二个变量即可。

有关 **ItemID** 结构的详细信息，可参考若干示例和手册。

如果选择了内存映射中相互重叠的两个变量，则当写入其中之一时两个变量都会发生更改。

#### **CAboutDlg** 类

此后，定义的 *CAboutDlg* 将显示一个信息框。此类中无 OPC 特定的代码，因此不会对它作详细说明。

#### **COPCDA\_AsyncDlg** 类的方法

*COPCDA\_AsyncDlg* 类除了构造函数和析构函数外还有以下方法：

- DoDataExchange
- OnInitDialog
- OnSysCommand
- OnPaint
- OnQueryDragIcon
- OnButtonStart()
- OnButtonRead()

### 6.3 C++ 中的 OPC 自定义接口 (异步通信)

- OnButtonWrite()
- OnCheckActivategroup()
- OnButtonStop()
- OnDestroy
- DisplayData
- UpdateData

#### OnInitDialog

有多个类变量可用于访问 COM 机制，而该机制是在初始化主对话框时进行初始化的。这些变量可具有以下用途：

- m\_pIOPCServer 是一个指向 OPC 服务器类的 IOPCServer 接口的指针。
- m\_pIOPCItemMgt 是一个指向 OPC 组类的 IOPCItemMgt 接口的指针。
- m\_pIOPCGroupStateMgt 是一个指向 OPC 组类的 IOPCGroupStateMgt 接口的指针。

```
m_pIOPCServer = NULL;  
m_pIOPCItemMgt = NULL;  
m_pIOPCGroupStateMgt = NULL;
```

在程序开始时，用户只能单击“启动示例”(Start Sample) 按钮。

所有其它按钮和复选框都将禁用。

这会确保在执行写入或读取操作前，正确地设置全部所需的 OPC 对象。

成员变量已分配给用于更改状态的相关控件：

```
m_CtrlStop.EnableWindow(FALSE);  
m_CtrlRead.EnableWindow(FALSE);  
m_CtrlWrite.EnableWindow(FALSE);  
m_CtrlChkActive.EnableWindow(FALSE);
```

#### OnButtonStart()

*OnButtonStart()* 将设置程序运行所需的 OPC 对象。首先，声明 OPC 对象的方法所需的若干局部变量：

```
void COPCDA_AsyncDlg::OnButtonStart()  
{
```

```
HRESULT r1;
CLSID clsid;
LONG TimBias = 0;
FLOAT PercentDeadband = 0.0;
DWORD RevisedUpdateRate;
LPWSTR ErrorStr1,ErrorStr2;
Cstring szErrorText;
m_pItemResult = NULL;
...
```

然后执行下列操作：

### CoInitialize

*CoInitialize()* 初始化 COM 库。 输入参数必须始终为 *NULL*。

```
r1 = CoInitialize(NULL);
if (r1 != S_OK)
{   if (r1 == S_FALSE)
    {   MessageBox("COM Library already initialized",
                 "Error CoInitialize()", MB_OK+MB_ICONEXCLAMATION);
    }
else
{   szErrorText.Format("Initialisation of COM Library \
failed.ErrorCode= %4x", r1);
    MessageBox(szErrorText, "Error CoInitialize()", MB_OK+MB_ICONERROR);
    SendMessage(WM_CLOSE);
    return;
}
}
```

### CLSIDFromProgID

*CLSIDFromProgID()* 将从指定的 ProgID 中检索全球唯一的类别标识符或 CLSID。  
*CoCreateInstance* 函数需要此方法。

此方法声明如下：

## 示例程序

### 6.3 C++ 中的 OPC 自定义接口 (异步通信)

```
HRESULT CLSIDFromProgID(LPCOLESTR lpszProgID,  
                         LPCLSID pclsid);
```

参数	说明
lpszProgID	指向 ProgID [输入参数] 的指针。
pclsid	指向检索的类别标识符 [返回值] 的指针。

在示例程序中，该方法检索 SIMATIC NET 的 OPC 服务器的类别标识符：

```
r1 = CLSIDFromProgID(L"OPC.SimaticNET", &clsid);  
if (r1 != S_OK)  
{    MessageBox("Retrieval of CLSID failed",  
           "Error CLSIDFromProgID()",  
           MB_OK+MB_ICONERROR);  
    CoUninitialize();  
    SendMessage(WM_CLOSE);  
    return;  
}
```

## CoCreateInstance

*CoCreateInstance()* 将创建已指定类别标识符的类的实例。此方法声明如下：

```
STDAPI CoCreateInstance (REFCLSID rclsid,  
                         LPUNKNOWN pUnkOuter,  
                         DWORD dwClsContext,  
                         REFIID riid,  
                         LPVOID *ppv);
```

参数	说明
rclsid	所需对象的类别标识符。
pUnkOuter	待创建的对象不是汇聚对象的一部分时的 NULL 指针。并非 NULL 指针的被解释为汇聚对象的 <b>IUnknown</b> 接口的指针。

参数	说明
dwClsContext	描述待创建对象的运行环境。 该参数将指定创建和管理此类别的对象的可执行代码是否在本地机器上运行，以及是否会为其创建一个单独的过程。在 CLSCTX 枚举数据类型中指定可用于此过程的常量。在示例中，使用了 CLSCTX_LOCAL_SERVER；换言之，用于管理服务器对象的可执行代码在与示例程序相同的计算机上运行（但是，以其自身的程序进行运行）。
riid	与对象进行通信的接口的标识符（在示例程序 <i>IOPCServer</i> 中）。
ppv	在成功调用该方法后包含所需接口指针的指针变量的地址。

以下方法调用将创建 OPC 服务器类的对象，并将指针返回到 *IOPCServer* 接口：

```
r1 = CoCreateInstance (clsid, NULL, CLSCTX_LOCAL_SERVER,
                      IID_IOPCServer,
                      (void**)&m_pIOPCServer);

if (r1 != S_OK)
{
    MessageBox("Creation of IOPCServer-Object failed",
              "Error CoCreateInstance()",
              MB_OK+MB_ICONERROR);

    m_pIOPCServer = NULL;
    CoUninitialize();
    SendMessage(WM_CLOSE);
    return;
}
```

### AddGroup

*IOPCServer* 的 *AddGroup()* 方法将创建一个 OPC 组并声明如下：

```
HRESULT AddGroup (LPWCSTR szName,
                  BOOL bActive,
                  DWORD dwRequestedUpdateRate,
                  OPCHANDLE hClientGroup,
                  LONG *pTimeBias,
                  FLOAT *pPercentDeadband,
                  DWORD dwLCID,
                  OPCHANDLE *phServerGroup,
                  DWORD *pRevisedUpdateRate,
```

```
REFIID riid,
LPUNKNOWN *ppUnk);
```

参数	说明
szName	可以由客户端自由分配但在客户端内必须唯一的组名称。
bActive	如果该组在创建时将处于未激活状态，则为 FALSE。如果该组在创建时将处于激活状态，则为 TRUE。
dwRequestedUpdateRate	指定将数据项的值或状态的更改告知客户端后的最短时间间隔。 指定适合的时间间隔将防止信息被发送至客户端的速度超过客户端能够处理的速度。
hClientGroup	客户端可以自由选择的以及由服务器利用某些通知返回的代码编号。这样一来，客户端便可标识其数据。 客户端使用此句柄来标识组。
pTimeBias	服务器时间与 UTC（世界标准时间，Universal Time Convention）的偏差
pPercentDeadband	指定数值在其中发生更改而不会引发通知的死区。 但只有在使用模拟值时该属性才有效。数值的上限值和下限值必须为已知值。 在本示例程序中，不存在这种情况。
dwLCID	选择返回文本时要由服务器使用的语言
phServerGroup	必须指定为带有某些函数调用的参数的、由服务器分配的句柄（例如， <i>RemoveGroup</i> ）。 服务器需要它来标识该组。
pRevisedUpdateRate	将数据项值或状态的更改告知客户端后、由服务器返回的最短时间间隔。
riid	在创建组之后，指向 OPC 组对象接口之一的标识符的指针将可用。 此参数无需另行调用 <i>QueryInterface</i> 方法。
ppUnk	指向所需接口的指针。

在示例程序中调用 *AddGroup* 时，会进行以下设置：

- *bActive* 参数被设置为值 FALSE。在创建它之后，该组即会处于未激活状态；换言之，系统没有为该组生成任何 *OnDataChange* 回调。
- 所需的更新时间间隔的持续时间为 500 毫秒。
- 因为只有一个组正在被使用，所以可以指定任何客户端句柄。
- 在示例程序中，不会抑制有关特定范围内的值更改的通知。因此，为 *pPercentDeadband* 参数输入值“0.0”。
- *AddGroup* 将返回一个指向 *IOPCItemMgt* 接口的指针作为返回值。

```
r1 = m_pIOPCServer->AddGroup(L"grp1",
                                FALSE,
                                500,
                                1,
                                &TimBias,
                                &PercentDeadband,
                                LOCALE_ID,
                                &m_GrpSrvHandle,
                                &RevisedUpdateRate,
                                IID_IOPCItemMgt,
                                (LPUNKNOWN*) &m_pIOPCItemMgt);
```

### AddItems

*IOPCItemMgt* 接口的 *AddItems()* 方法将创建一个 OPC 数据项并声明如下：

```
HRESULT AddItems (DWORD dwNumItems,
                  OPCITEMDEF *pItemArray,
                  OPCITEMRESULT **ppAddResults,
                  HRESULT **ppErrors);
```

参数	说明
<i>dwNumItems</i>	待插入的数据项数。
<i>pItemArray</i>	具有 <i>OPCITEMDEF</i> 类型的元素的数组。 此类型的结构变量包含由服务器创建数据项时所需的全部信息。

参数	说明
ppAddResults	具有 OPCITEMRESULT 类型的元素的数组。由 OPC 服务器发送的返回值是此类型的结构变量。
ppErrors	具有 HRESULT 类型元素的数组。 这些变量将返回无法成功创建数据项时的错误代码，或返回有关成功的方法调用的信息。

在调用 *AddItems* 之前，必须利用有效值创建并初始化一个具有 OPCITEMDEF 类型的元素的数组。进行此操作时，必须考虑以下几种情况：

- 该示例中数据项的访问路径并非必不可少，并且空字符串将会在此指定。
- 在模块 OPCDA\_AsyncDlg.cpp 的开始处指定数据项标识符，并将其分配给变量 szItemID1 和 szItemID2。
- 数据项将在创建后处于激活状态。
- 示例程序使用客户端句柄 0 和 1。
- SIMATIC NET 的 OPC 服务器不需要 *BinaryLargeObjects*，因此结构组件 dwBlobSize 的值为“0”。
- 对于第一个数据项，服务器必须在对应于数据项原始数据类型的类型中发送返回值。
- 对于第二个数据项，服务器必须将返回值作为双字节整数进行传送。

```
m_Items[0].szAccessPath = L"";  
m_Items[0].szItemID = szItemID1;  
m_Items[0].bActive = TRUE;  
m_Items[0].hClient = 0;  
m_Items[0].dwBlobSize = 0;  
m_Items[0].pBlob = NULL;  
m_Items[0].vtRequestedDataType = 0;  
  
m_Items[1].szAccessPath = L"";  
m_Items[1].szItemID = szItemID2;  
m_Items[1].bActive = TRUE;  
m_Items[1].hClient = 1;  
m_Items[1].dwBlobSize = 0;  
m_Items[1].pBlob = NULL;  
m_Items[1].vtRequestedDataType = VT_I2;
```

*m\_pItemResult* 指针作为 *COPCDA\_AsyncDlg* 类的一个属性存在。

由可以使用此变量访问的服务器返回的结果。*m\_pErrors* 是一个指向错误代码的指针：

```
r1 = m_pIOPCItemMgt->AddItems(2, m_Items, &m_pItemResult, &m_pErrors);
```

如果 *AddItems* 调用未成功，将中止程序。但是，程序首先必须释放其正在使用的资源：

```
if ( (r1 != S_OK) && (r1 != S_FALSE) )
{
    MessageBox("AddItems failed!", "Error AddItems()", MB_OK+MB_ICONERROR);

    m_pIOPCItemMgt->Release();
    m_pIOPCItemMgt = NULL;
    m_GrpSrvHandle = 0;
    m_pIOPCServer->Release();
    m_pIOPCServer = NULL;
    CoUninitialize();
    SendMessage(WM_CLOSE);
    return;
}
```

### GetErrorString

如果 *AddItems()* 的返回值指示已出现错误，*IOPCServer* 接口的 *GetErrorString()* 方法将会获取相应的错误消息。此方法声明如下：

```
HRESULT GetErrorString (HRESULT dwError,
                        LCID dwLocale,
                        LPWSTR *ppString);
```

参数说明：

参数	说明
<b>dwError</b>	由服务器返回的错误代码。
<b>dwLocale</b>	错误消息的语言标识符。
<b>ppString</b>	指向空终止字符串 ( <i>GetErrorString</i> 将错误消息返回到此字符串) 的双重指针。

```
else
{
    m_pIOPCServer ->GetErrorString(m_pErrors[0], LOCALE_ID,
                                      &ErrorStr1);

    m_pIOPCServer ->GetErrorString(m_pErrors[1], LOCALE_ID,
```

```

    &ErrorStr2);

CString szOut;
szOut.Format("Item %ls :\n %ls\n\nItem %ls :\n %ls\n",
             szItemID0,
             ErrorStr1,
             szItemID1,
             ErrorStr2);

MessageBox(szOut, "Result AddItems()", MB_OK+MB_ICONEXCLAMATION);
CoTaskMemFree(ErrorStr1);
CoTaskMemFree(ErrorStr2);
}

```

## QueryInterface

*IUnknown* 接口的 *QueryInterface()*

方法将返回一个指向其标识符被指定为输入参数的接口的指针。*QueryInterface()* 声明如下：

```
HRESULT QueryInterface (REFIID iid, void **ppvObject);
```

参数说明：

参数	说明
iid	所需接口的标识符。
ppvObject	在成功调用该方法后包含所需接口指针的指针变量的地址。 如果对象不支持此接口，将返回错误代码和 NULL 指针。

*QueryInterface* 获得一个指向 *IOPCAsyncIO2*

接口（该接口提供了用于异步读取和写入的方法）的指针：

```

r1 = m_pIOPCItemMgt ->QueryInterface(IID_IOPCAsyncIO2,
                                         (void**)& m_pIOPCAsyncIO2);

if (r1 < 0)
{
    MessageBox("No IOPCAsyncIO found!",
              "Error QueryInterface()", MB_OK+MB_ICONERROR);

    ...
}

return;
}

```

## 创建回调对象

必须在客户端程序中执行 `IOPCDataCallback` 接口，以便能够接收来自 OPC 服务器的通知。在示例程序中，该实现过程利用了“活动模板库”(Active Template Library, ATL)。将特定于示例的 `COPCDataCallback` 类用作 `CComObject` 模板类的参数。

`CreateInstance` 方法可创建 `CComObject` 类的对象，可通过 `pOPCDataCallback` 指针访问该对象：

```
CComObject<COPCDataCallback>* pCOPCDataCallback;
CComObject<COPCDataCallback>::CreateInstance (&pCOPCDataCallback);
```

## 建立对话框与回调对象之间的连接

`OnButtonStart()` 可调用对话框类的 `InformAboutDialog`

方法并向其传递一个指向当前对话框的指针（当前指针）。`InformAboutDialog` (在 `Callback.cpp` 模块中) 可将该指针存储在 `m_pCDlgClass` 成员变量中。

`OnReadComplete()`、`OnWriteComplete()` 和 `OnDataChange` 使用该指针来调用对话框的 `DisplayData` 方法。若没有 `m_pCDlgClass` 中的信息，就不会有用于指示将在其中显示数据的对话框的引用。

```
pCOPCDataCallback->InformAboutDialog (this);
```

下图说明了 `COPCDA_AsyncDlg` 和 `IOPCDataCallback` 类方法之间的关系：

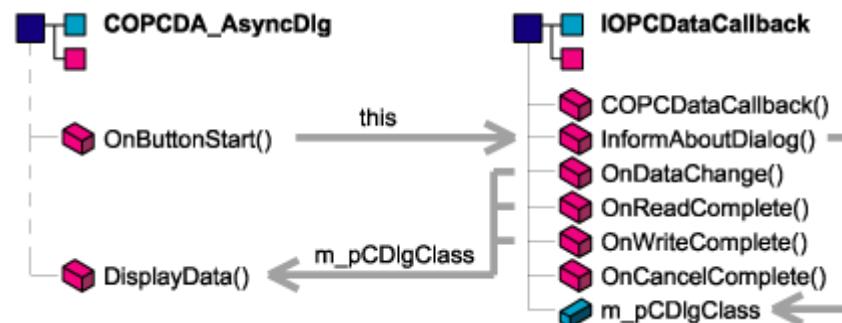


图 6-23 用于链接对话框和回调对象的 `COPCDA_AsyncDlg` 和 `IOPCDataCallback` 类方法之间的关系

## 连接 OPC 服务器与客户端的回调对象

`GetUnknown()` 方法将获取一个指向回调对象的 `IUnknown` 接口的指针：

### 6.3 C++ 中的 OPC 自定义接口 (异步通信)

```
LPUNKNOWN pCbUnk;
pCbUnk = pCOPCDataCallback->GetUnknown();
```

*AtlAdvise()* 方法将创建 OPC 服务器与回调对象之间的连接。 *AtlAdvise* 声明如下：

```
HRESULT AtlAdvise (IUnknown *pUnkCP,
                    IUnknown *pUnk,
                    const IID& iid,
                    LPDWORD pdw)
```

参数	说明
pUnkCP	指向客户端希望与其建立连接的 <b>IUnknown</b> 接口的指针。
pUnk	指向回调对象的 <b>IUnknown</b> 接口的指针。
iid	连接点的标识符。 正常情况下，会指定客户端上执行回调功能的接口标识符。
pdw	返回参数。指向用于标识连接的代码的指针。当以 <i>AtlUnadvise</i> 终止连接时需要此值。

如果 *AtlAdvise* 不能成功执行，则对话框中将显示如下内容：

```
HRESULT hRes = AtlAdvise(m_pIOPCGroupStateMgt, pCbUnk,
                           IID_IOPCDataCallback, &m_dwAdvise);

if (hRes != S_OK)
{
    AfxMessageBox("Advise failed!");

    ...
    return;
}
```

#### 启用按钮

在 *OnButtonStart()* 设置完所有必要的 OPC 对象后，将禁用“启动示例”(*Start Sample*)按钮。所有其它按钮都将被启用。此策略确保 *OnButtonStart()* 只执行一次。这样就无需程序中的其它查询。

```
m_CtrlStop.EnableWindow(TRUE);
m_CtrlRead.EnableWindow(TRUE);
m_CtrlWrite.EnableWindow(TRUE);
m_CtrlChkActive.EnableWindow(TRUE);
m_CtrlStart.EnableWindow(FALSE);
```

## OPCITEMDEF 结构

*OPCITEMDEF* 具有下列结构:

```
typedef struct {
    LPWSTR szAccessPath;
    LPWSTR szItemID;
    BOOL bActive;
    OPCHANDLE hClient;
    DWORD dwBlobSize;
    BYTE *pBlob;
    VARTYP vtRequestedDataType;
    Word wReserved;
} OPCITEMDEF;
```

*OPCITEMDEF* 的变量:

变量	说明
<b>szAccessPath</b>	数据项的可选访问路径 (SIMATIC NET 不需要此路径)。
<b>szItemID</b>	由客户端分配的 ItemID。
<b>bActive</b>	如果将激活组中的数据项值是否发生更改告知客户端，则为 TRUE; 如果没有告知客户端，则为 FALSE。
<b>hClient</b>	由客户端分配的数据项的句柄。 服务器通过客户端调用传送客户端句柄 (例如, <i>OnDataChange</i> )，以便客户端能够访问其结构中的相关变量。
<b>dwBlobSize</b>	服务器上存储区 (此存储区用于存储有关更快地访问数据项数据的其它信息) 的大小。
<b>pBlob</b>	指向上述存储区的指针。
<b>vtRequestedDataType</b>	客户端请求的数据类型。

## OPCITEMRESULT 结构

*OPCITEMRESULT* 具有下列结构:

```
typedef struct {
    OPCHANDLE hServer;
    VARTYPE vtCanonicalDataType;
    WORD wReserved;
} OPCITEMRESULT;
```

```

    DWORD dwAccessRights;
    DWORD dwBlobSize;
    BYTE *pBlob;
} OPCITEMRESULT;

```

*OPCITEMRESULT* 的变量：

变量	说明
hServer	由服务器分配的数据项的句柄。 客户端通过服务器调用传送服务器句柄，以便服务器能够访问其结构中的相关变量。
vtCanonicalDataType	由服务器使用的数据项的数据类型。
dwAccessRights	有关数据项是只能读取、只能写入还是可读写的信息。
dwBlobSize	服务器上存储区（此存储区用于存储有关更快地访问数据项数据的其它信息）的大小。
pBlob	指向上述存储区的指针。

### OnButtonRead()

*OnButtonRead()* 可启动异步读取作业。可以执行下列操作：

#### 变量声明和似然性检查

在若干局部变量声明之后，程序将检查数据项是否存在。在 *OnButtonStart* 方法中成功调用 **AddItems** 后，将初始化 *m\_pItemResult* 变量。*m\_pErrors[0]* 数组元素的值必须为 *S\_OK*，否则说明该数据项未正确创建：

```

void COPCDA_AsyncDlg::OnButtonRead()
{
    OPCHANDLE *phServer;
    DWORD dwCancelID;
    HRESULT *pErrors;
    HRESULT r1;
    LPWSTR ErrorStr;
    CString szOut

    if (m_pErrors[0] != S_OK)
    {
        MessageBox("OPC Item0 not available!",
                  "Error Read async",
                  MB_OK+MB_ICONERROR);
    }
}

```

```

    return;
}
}
}

```

## Read

*Read* 可异步读取 OPC 数据项的值并声明如下：

```

HRESULT Read (DWORD dwCount,
               OPCHANDLE * phServer,
               DWORD dwTransactionID,
               DWORD *pdwCancelID,
               HRESULT ** ppErrors);

```

参数	说明
<b>dwCount</b>	读取值的数据项的数目。
<b>phServer</b>	含有所读取值的数据项服务器句柄的数组。
<b>dwTransactionID</b>	客户端分配的用来标识异步事务的代码。由服务器使用 <i>OnReadComplete</i> 调用返回此代码。
<b>pdwCancelID</b>	事务将被取消时必须指定由服务器生成的代码。
<b>ppErrors 数组</b>	具有 <b>HRESULT</b> 类型元素的数组。 这些变量将返回无法成功调用 <i>Read()</i> 时的错误代码，或返回有关成功的方法调用的信息。

程序为含 **OPCHANDLE** 类型元素的数组设置存储器，并使用 *m\_pItemResult* 变量的结构组件 *hServer* 将唯一的数组元素进行初始化。当调用 *AddItems* 时，OPC 服务器在此为数据项输入一个服务器句柄：

```

phServer = new OPCHANDLE[1];
phServer[0] = m_pItemResult [0].hServer;
r1 = m_pIOPCAsyncIO2 ->Read(1, phServer, 1, &dwCancelID,
                             &pErrors);

```

返回值 *S\_FALSE* 指示有一个或多个数据项无法读取。在这种情况下，*GetErrorString* 方法将获取相应的错误消息并将其显示在对话框中：

```

if (r1 == S_FALSE)
{
    m_pIOPCServer ->GetErrorString(pErrors[0], LOCALE_ID,
                                      &ErrorStr);
}

```

### 6.3 C++ 中的 OPC 自定义接口 (异步通信)

```

szOut.Format ("%s", ErrorStr);
MessageBox(szOut, "Error Reading Item0",
MB_OK+MB_ICONERROR);
CoTaskMemFree(ErrorStr);
}

```

在程序结束之前，会再次释放由 *OnButtonRead()* 保留的内存。利用 *phServer* 数组，可使用删除完成该操作，因为该存储器是使用新建创建的。*pErrors* 数组的存储区由 *CoTaskMemAlloc* 进行组织，因此要使用 *CoTaskMemFree* 来对其进行释放：

```

delete[] phServer;
CoTaskMemFree(pErrors);
}

```

### OnButtonWrite()

*OnButtonWrite()* 可启动异步写入作业。可以执行下列操作：

变量声明和似然性检查

变量声明与 *OnButtonRead()* 方法相对应。此外，可为读取操作的结果声明 VARIANT 类型的数组。要检查数据项是否存在，可使用如 *OnButtonRead()* 中用到的 *hServer* 结构组件：

```

void COPCDA_AsyncDlg::OnButtonWrite()
{
    OPCHANDLE *phServer;
    DWORD dwCancelID;
    VARIANT values[1];
    HRESULT *pErrors;
    HRESULT r1;
    LPWSTR ErrorStr;
    CString szOut;

    if (m_pErrors[0] != S_OK)
    {
        MessageBox("OPC Item not available!", "Error Write async",
MB_OK+MB_ICONERROR);
        return;
    }
    phServer = new OPCHANDLE[1];

```

```
phServer[0] = m_pItemResult[0].hServer;
```

```
...
```

## Write

*Write* 可异步写入 OPC 数据项的值并声明如下：

```
HRESULT Write (DWORD dwCount,
               OPCHANDLE *phServer,
               VARIANT *pItemValues,
               DWORD dwTransactionID,
               DWORD *pdwCancelID
               HRESULT **ppErrors);
```

参数	说明
<b>dwCount</b>	写入值的数据项数。
<b>phServer</b>	含有所写入值的数据项服务器句柄的数组。
<b>pItemValues</b>	具有待写入值的数组。
<b>dwTransactionID</b>	客户端分配的用来标识异步事务的代码。
<b>pdwCancelID</b>	事务将被取消时必须指定由服务器生成的代码。
<b>ppErrors</b>	具有 <b>HRESULT</b> 类型元素的数组。 这些变量将返回无法成功调用 <i>Write()</i> 时的错误代码，或返回有关成功的方法调用的信息。

该示例不使用 **dwTransactionID** 参数，所以任何值的设置方式如下：

```
r1 = m_pIOPCAsyncIO2 ->Write(1, phServer, values, 2,
                               &dwCancelID, &pErrors);
```

如果 *Write* 返回 **S\_FALSE**，则表示发生了错误。

程序会获取相应的错误消息。如果无法执行

*Write*，则对话框中会显示一条有关该结果的错误消息。

*delete[] phServer* 将释放由 *OnButtonWrite()* 保留的内存：

```
delete[] phServer;
if (r1 == S_FALSE)
{
    m_pOPCServer->GetErrorString(pErrors[0], LOCALE_ID,
                                   &ErrorStr);

    m_WriteResult = ErrorStr;
    UpdateData(FALSE);
```

### 6.3 C++ 中的 OPC 自定义接口 (异步通信)

```
CoTaskMemFree(ErrorStr);  
}  
  
if (FAILED(r1))  
{    szOut.Format("Method call IOPCAsyncIO2::Write failed \  
with error code %x", r1);  
    MessageBox(szOut, "Error Writing Item0", MB_OK+MB_ICONERROR);  
}  
else  
{    CoTaskMemFree(pErrors);  
}
```

#### 应用对话框中的值

带有 TRUE 参数的 *UpdateData*

方法调用可将所有控制变量的内容传送给相关的成员变量。IDC\_EDIT\_WRITEVAL 文本框的 *m\_WriteVal* 成员变量值存储在唯一的 *values[]* 元素的 *iVal* 组件中。*vt* 组件被分配为 *VT\_I2* 类型；换言之，*iVal* 将被解释为长度为两个字节的整数：

```
UpdateData(TRUE);  
values[0].vt = VT_I2;  
values[0].iVal = m_WriteVal;
```

#### OnCheckActivategroup()

单击“组激活”(Group Active) 复选框后会调用 *OnCheckActivategroup*。

激活或取消激活该组取决于复选框的状态。

可以执行下列操作：

#### 采用对话框中的值

带有 TRUE 参数的 *UpdateData*

方法调用将所有控制元素的内容传送给相关的成员变量：

```
void COPCDA_AsyncDlg::OnCheckActivategroup()  
{    UpdateData(TRUE);
```

#### SetState

*SetState* 可指定一个组的若干属性并声明如下：

```
HRESULT SetState (DWORD * pRequestedUpdateRate,  
                  DWORD * pRevisedUpdateRate,
```

```

    BOOL *pActive,
    LONG * pTimeBias,
    FLOAT * pPercentDeadband
    DWORD * pLCID,
    OPCHANDLE *phClientGroup);

```

参数	说明
pRequestedUpdateRate	客户端所需的用于数据项的数据和状态更改的更新间隔。
pRevisedUpdateRate	服务器可执行的最短的组更新间隔。
pActive	组状态: 状态为 TRUE 时激活组。 状态为 FALSE 时禁用组。
pTimeBias	服务器时间与 UTC (世界标准时间, Universal Time Convention) 的偏差
pPercentDeadband	指定数值在其中发生更改而不会引发通知的死区。 但只有在使用模拟值时该属性才有效。 还必须知道数值的上限和下限。 在此示例程序中, 不会出现这种情况。
pLCID	用于组的局部调整的标识号 (语言等)。
phClientGroup	组的新客户端句柄。

只应激活组, 所以仅为 *pActive* 参数赋值。必须有一个合适的变量用于 *pRevisedUpdateRate* 返回参数:

```

DWORD dwRevUpdateRate;
HRESULT r1= m_pIOPCGroupStateMgt ->SetState
            (NULL,
             &dwRevUpdateRate,
             &m_bChkActivateGroup,
             NULL, NULL, NULL, NULL);

if (FAILED(r1))
{
    MessageBox("Set State failed", "Error",
              MB_OK+MB_ICONERROR);

    return;
}

```

**OnButtonStop()**

*OnButtonStop()* 可移除程序中使用的 OPC 对象并释放相应资源。单击“停止示例”(Stop Sample) 按钮后、执行 *OnDestroy* 方法或发送 WM\_CLOSE 消息后（在单击按钮关闭对话框后或明确调用 *SendMessage* 后）会调用该方法。

可以执行下列操作：

**AtlUnadvise**

*AtlUnadvise* 可终止 OPC 服务器与回调对象之间的连接并声明如下：

```
HRESULT AtlUnadvise (IUnknown*pUnkCP,
                      const IID &iid,
                      DWORD dw);
```

参数	说明
pUnkCP	指向与客户端相连接的对象的 IUnknown 接口的指针。
iid	连接点的标识符。 正常情况下，会指定客户端上执行回调功能的接口标识符。
dw	用于标识连接的标识号。

```
void COPCDA_AsyncDlg::OnButtonStop()
{
    HRESULT r1;
    OPCHANDLE *phServer;
    HRESULT *pErrors;
    HRESULT hRes = AtlUnadvise(m_pIOPCGroupStateMgt,
                               IID_IOPCDataCallback,
                               m_dwAdvise);
```

**Release**

每个 COM 接口都具有能够释放接口所使用资源的 *Release* 方法。实际上，*Release* 不会从内存中移除任何对象，只会使接口的参考计数器递减。

只有当不再引用该接口时，才会真正地删除这些对象。

```
m_pIOPCGroupStateMgt ->Release();
```

**RemoveItems**

*IOPCItemMgt* 接口的 *RemoveItems* 方法可移除 OPC 数据项并按如下方式声明：

```
HRESULT RemoveItems (DWORD dwCount,
                      OPCHANDLE *phServer,
                      HRESULT **ppErrors);
```

参数	说明
<i>dwCount</i>	待移除的数据项数目。
<i>phServer</i>	含有待移除数据项的服务器句柄的数组。
<i>ppErrors</i>	具有 <b>HRESULT</b> 类型元素的数组。这些变量将返回未成功调用 <i>RemoveItems()</i> 时的错误代码或返回有关方法成功调用的信息。

该程序段为含 **OPCHANDLE** 类型元素的数组设置存储器，并使用 *m\_pItemResult* 变量的结构组件 *hServer* 将数组元素进行初始化。当调用 *AddItems* 时，OPC 服务器会在此输入数据项的服务器句柄。在调用 *RemoveItems* 之后，会再次释该数组的存储器：

```
phServer = new OPCHANDLE[2];
phServer[0] = m_pItemResult[0].hServer;
phServer[1] = m_pItemResult[1].hServer;
r1 = m_pIOPCItemMgt ->RemoveItems(2, phServer, &pErrors);
if ( (r1 != S_OK) && (r1 != S_FALSE) )
{
    MessageBox("RemoveItems failed!",
               "Error RemoveItems ()",
               MB_OK+MB_ICONERROR);
}
delete[] phServer;
```

可使用 *CoTaskMemFree* 方法释放 *pErrors* 和 *m\_pItemResult* 的存储区，因为它们是通过 COM 机制获取的。使用 *Release* 方法来释放 *m\_pIOPCItemMgt* 和 *m\_pIOPCAsyncIO2* 的资源：

```
CoTaskMemFree(pErrors);
CoTaskMemFree(m_pItemResult);
m_pItemResult =NULL;
CoTaskMemFree(m_pErrors);
m_pErrors =NULL;
m_pIOPCAsyncIO2->Release();
m_pIOPCAsyncIO2 = NULL;
```

## 示例程序

### 6.3 C++ 中的 OPC 自定义接口 (异步通信)

```
m_pIOPCItemMgt->Release();  
m_pIOPCItemMgt = NULL;
```

#### RemoveGroup

*RemoveGroup()* 可用于移除服务器中的组并按如下方式声明:

```
HRESULT RemoveGroup (OPCHANDLE hServerGroup, BOOL bForce);
```

参数	说明
hServerGroup	待移除的组的服务器句柄。
bForce	指定当组的引用仍然存在时是否能够移除该组。

```
r1 = m_pIOPCServer->RemoveGroup(m_GrpSrvHandle, TRUE);  
if (r1 != S_OK)  
{    MessageBox("RemoveGroup failed!",  
           "Error RemoveGroup()",  
           MB_OK+MB_ICONERROR);  
}  
m_GrpSrvHandle = NULL;
```

使用 **Release** 方法来释放 OPC 服务器对象的资源, *CoUninitialize* 将关闭 COM 库:

```
m_pIOPCServer->Release();  
m_pIOPCServer = NULL;  
CoUninitialize();
```

最后, 除“启动程序”(Start Sample) 按钮外, 该程序会禁用所有的控件:

```
m_CtrlStart.EnableWindow(TRUE);  
m_CtrlStop.EnableWindow(FALSE);  
m_CtrlRead.EnableWindow(FALSE);  
m_CtrlWrite.EnableWindow(FALSE);  
m_CtrlChkActive.EnableWindow(FALSE);  
}
```

#### OnDestroy

**CWnd** 类的此方法将被覆盖以便对话框窗口关闭时可以执行所有的“清除作业”。

*OnButtonStop()* 将删除所有的 OPC 对象并释放相应的资源。此后, 将调用父类的 *OnDestroy* 来释放窗口对象使用的存储器:

```
void COPCDA_AsyncDlg::OnDestroy()
{
    if (m_pIOPCServer)
        OnButtonStop();
    CDialog::OnDestroy();
}
```

## DisplayData

该对话框类有三种方法可用来在接收到来自 OPC 服务器的消息之后显示数据。由于服务器的各个通知会返回不同的数据，所以需要不同的方法。

可以使用以下方法：

- 数据更改后（每个数据项有一种方法）：

```
void DisplayData0 (CString ReadVal, CString ReadQuality, CString ReadTS) void
DisplayData1 (CString ReadVal, CString ReadQuality, CString ReadTS)
```

- 读取操作完成后：

```
void DisplayData (long Value, CString Quality, CString TimeStamp)
```

- 写入操作完成后：

```
void DisplayData (HRESULT ErrorCode)
```

### DisplayData0 (CString ReadVal, CString ReadQuality, CString ReadTS)

使用回调类的 *OnDataChange* 方法调用此版本的 *DisplayData*，并更新“关于已更改数据”(On Data Changed) 框中文本框显示的内容。

---

#### 说明

在这种情况下，会更新 Item0 文本框中的内容。为了更新 Item1 文本框，需要执行类似的 *DisplayData1* 方法。

---

## 示例程序

### 6.3 C++ 中的 OPC 自定义接口（异步通信）

```
void COPCDA_AsyncDlg::DisplayData0(CString ReadVal,  
                                    CString ReadQuality,  
                                    CString ReadTS)
```

参数	说明
ReadVal	已读取的数值。
ReadQuality	有关数据完整性的信息。
ReadTS	时间戳（接收来自 OPC 服务器的数据的时间）。

将这些参数值分配给对话框的成员变量。 *UpdateData* *UpdateData*  
将新数据从成员变量传送至相应的控件：

```
{    m_Quality0 = ReadQuality;  
    m_TimeStamp0 = ReadTS;  
    m_Value0 = ReadVal;  
    UpdateData(FALSE);  
}
```

### DisplayData (long Value, CString Quality, CString TimeStamp)

回调类的 *OnReadComplete()* 方法将调用此方法以更新“读取值”(Read Value)  
按钮右侧文本框中的显示内容：

```
void COPCDA_AsyncDlg::DisplayData(long Value,  
                                    CString Quality,  
                                    CString TimeStamp)
```

参数	说明
Value	已读取的值。
Quality	有关数据完整性的信息。
TimeStamp	从 OPC 服务器中接收数据的时间。

将这些参数分配给对话框的相关成员变量。 *UpdateData* *UpdateData*  
将新数据从成员变量传送至相应的控件：

```
{    m_ReadValue = Value;  
    m_QualityRead = Quality;  
    m_TimeStampRead = TimeStamp;
```

```

    UpdateData(FALSE);
}

```

## DisplayData (HRESULT ErrorCode)

回调类的 *OnWriteComplete()* 方法将调用此方法来显示“写入值”(Write Value)按钮右侧文本框中的关于写入操作结果的状态消息：

```
void COPCDA_AsyncDlg::DisplayData (HRESULT ErrorCode)
```

参数	说明
ErrorCode	OPC 服务器返回的错误代码。

*IOPCServer* 接口的 *GetErrorString()* 方法将获得错误代码的错误消息。

将该消息分配给成员变量 *m\_WriteResult*。在 *UpdateData*

将此消息显示在文本框中之前，必须使用 *Remove()* 移除换行符：

```

{   LPWSTR ErrorStr;
    m_pIOPCServer->GetErrorString(ErrorCode, LOCALE_ID,
                                    &ErrorStr);

    m_WriteResult = ErrorStr;
    m_WriteResult.Remove('\r');
    m_WriteResult.Remove('\n');
    UpdateData(FALSE);
    CoTaskMemFree(ErrorStr);
}

```

## UpdateData

在某些情况下，访问资源期间更改线程可能产生不一致数据或未定义的状态。

这种情况下，必须明确地阻止线程更改。为此，可使用 *CCriticalSection* 类。

创建完此类对象后，只有在调用了 *Unlock()* 方法后才能再次更改线程。

示例程序使用此机制对控件实现无错更新。程序启动后会创建一个 *CCriticalSection* 类的对象。紧随父类的 *UpdateData* 方法调用之后，*Unlock* 将再次释放其它线程的资源：

```
BOOL COPCDA_AsyncDlg::UpdateData( BOOL bSaveAndValidate )
{
    BOOL bRes;
    m_cCritSec.Lock();
    bRes = CDialog::UpdateData( bSaveAndValidate );
}
```

```

m_cCritSec.Unlock();
return bRes;
}

```

### 6.3.10 Callback.cpp 和 Callback.h

在示例程序中，*COPCDataCallback* 类是 *IOPCDataCallback* 接口的实现。  
该类提供以下方法用来接收来自 OPC 服务器的通知：

- *OnDataChange()*
- *OnReadComplete()*
- *OnWriteComplete()*
- *InformAboutDialog*

文件中还包括辅助函数 *GetQualityText*

#### OnDataChange

OPC 服务器会在数据发生更改时调用此方法：

```

STDMETHODIMP COPCDataCallback::OnDataChange(
    DWORD dwTransid,
    OPCHANDLE hGroup,
    HRESULT hrMasterquality,
    HRESULT hrMastererror,
    DWORD dwCount,
    OPCHANDLE __RPC_FAR *phClientItems,
    VARIANT __RPC_FAR *pvValues,
    WORD __RPC_FAR *pwQualities,
    FILETIME __RPC_FAR *pftTimeStamps,
    HRESULT __RPC_FAR *pErrors)

```

参数	说明
<i>dwTransid</i>	如果是数值更改引起了方法调用，则此类型方法调用的代码为零，如果代码不为零，则此方法调用是由刷新触发的。
<i>hGroup</i>	组的客户端句柄（允许客户端标识组）

参数	说明
hrMasterquality	关于数据完整性的信息：如果所有数据项的 OPC_QUALITY_MASK 值都是 OPC_QUALITY_GOOD，则为 S_OK，否则为 S_FALSE。
hrMastererror	如果所有数据项的错误消息都是 S_OK，则为 S_OK，否则为 S_FALSE。
dwCount	其值存在的数据项数目。
phClientItems	含有其值已更改的数据项客户端句柄的数组。
pvValues	类型为 VARIANT 的数组，其中包含已更改数据项的值。
pwQualities	含有关于各个数据项值完整性信息的数组
pftTimeStamps	含有各个数据项时间信息的数组
pErrors	含有数据项错误消息的数组

此方法的任务是将 OPC 服务器传送的值作为参数转换为 **CString** 类型的对象。

*DisplayData* 方法要求输入参数为此类型。

程序为 **CString** 对象创建三个数组用来存储数据项的值、值的质量信息以及时间信息：

```
{    CString szReadVal;
    CString szReadQuality;
    CString szReadTS;
```

*CComVariant* 是 VARIANT 数据类型的包装类。将为回路中的每个数据项创建一个 **CComVariant** 对象。为此，将包含 VARIANT 值的构造函数用作参数：

```
CCoVariant *pvarOut;
DWORD i;
for (i = 0; i < dwCount; i++)
{    pvarOut = new CComVariant(pvValues[i]);
```

在下一步中，*ChangeType* 方法将已读取的值转换为字符串。由于这仍然是一个 **CComVariant** 类的对象，因此可通过 *pvarOut* 的 *bstrVal* 组件访问该字符串。*Format* 方法的结果是所需的 **CString** 对象：

```
pvarOut->ChangeType(VT_BSTR);
szReadVal.Format("%ls", pvarOut->bstrVal);
```

### 6.3 C++ 中的 OPC 自定义接口 (异步通信)

由于我们只关注 *szReadVal* 变量，所以 *pvarOut* 对象会在用过之后被立即删除：

```
delete pvarOut;
```

*GetQualityText* 可将有关已读取值的质量信息作为 **CString** 对象返回。此方法的参数是 OPC 服务器的一个标识号：

```
szReadQuality = GetQualityText(pwQualities[i]);
```

OPC 服务器可将数据项的时间标记作为 **FILETIME** 类型的变量返回。**COleDateTime** 类的 *Format* 方法的结果是 **CString** 对象形式的时间：

```
szReadTS = COleDateTime(pftTimeStamps[i]).Format();
}
```

所有的结果都被转换为 **CString** 对象后，程序可调用 **COPCDA\_AsyncDlg** 类的 *DisplayData0* 方法或 *DisplayData1* 方法：

```
switch (phClientItems[i])
{
    case 0:
        m_pCDlgClass->DisplayData0(szReadVal,
                                      szReadQuality,
                                      szReadTS);
        break;
    case 1:
        m_pCDlgClass->DisplayData1(szReadVal,
                                      szReadQuality,
                                      szReadTS);
        break;
    default: ASSERT(0);
}
}

return S_OK;
};
```

### OnReadComplete()

OPC 服务器会在完成异步读取后调用此方法。*OnReadComplete()* 的参数与 *OnDataChange* 的参数相匹配：

```
STDMETHODIMP COPCDataCallback::OnReadComplete(
    DWORD dwTransid,
    OPCHANDLE hGroup,
    HRESULT hrMasterquality,
    HRESULT hrMastererror,
    DWORD dwCount,
    OPCHANDLE __RPC_FAR *phClientItems,
    VARIANT __RPC_FAR *pvValues,
    WORD __RPC_FAR *pwQualities,
    FILETIME __RPC_FAR *pftTimeStamps,
    HRESULT __RPC_FAR *pErrors)
```

参数	说明
dwTransid	如果是数值更改引起了方法调用，则此类型方法调用的代码为零，如果代码不为零，则此方法调用是由刷新触发的。
hGroup	组的客户端句柄（允许客户端标识组）
hrMasterquality	关于数据完整性的信息：如果所有数据项的 OPC_QUALITY_MASK 值都是 OPC_QUALITY_GOOD，则为 S_OK，否则为 S_FALSE。
hrMastererror	如果所有数据项的错误消息都是 S_OK，则为 S_OK，否则为 S_FALSE。
dwCount	其值存在的数据项数目。
phClientItems	含有其值已更改的数据项客户端句柄的数组。
pvValues	类型为 VARIANT 的数组，其中包含已更改数据项的值。
pwQualities	含有关于各个数据项值完整性信息的数组
pftTimeStamps	含有各个数据项时间信息的数组
pErrors	含有数据项错误消息的数组

在处理 OPC 服务器传送的数据之前，程序会对 *pErrors* 变量进行检查。

如果无错读取，则数据的评估结果将与 *OnDataChange* 程序中的评估结果相同。

然而，这种情况下，数据项的值会被转换为 *Long Integer* 类型：

```
if (pErrors[0] == S_OK)
{
    CComVariant * pvarOut;
    pvarOut = new CComVariant(pvValues[0]);
    pvarOut->ChangeType(VT_I4);
```

## 示例程序

### 6.3 C++ 中的 OPC 自定义接口 (异步通信)

```
CString readQuality = GetQualityText(pwQualities[0]);
CString readTS = COleDateTime(pftTimeStamps[0]).Format();
m_pCDlgClass->DisplayData(pvarOut->intVal, readQuality, readTS);
delete pvarOut;
}
```

如果出错，则 *GetQualityText* 方法会返回属于错误代码的消息：

```
else
{
    CString readQuality = GetQualityText(pErrors[0]);
    m_pCDlgClass->DisplayData(0, readQuality, "");
}
return S_OK;
};
```

### OnWriteComplete()

OPC 服务器会在完成异步写入后调用此方法。

```
STDMETHODIMP COPCDataCallback::OnWriteComplete(
    DWORD dwTransid,
    OPCHANDLE hGroup,
    HRESULT hrMastererr,
    DWORD dwCount,
    OPCHANDLE __RPC_FAR *pClienthandles,
    HRESULT __RPC_FAR *pErrors)
```

参数	说明
dwTransid	异步写入时的标识号
hGroup	组的客户端句柄
hrMastererr	如果所有数据项的错误消息都是 S_OK，则为 S_OK，否则为 S_FALSE。
dwCount	已写入值的数据项数目。
pClienthandles	含有已写入值的数据项客户端句柄的数组。
pErrors	含有数据项错误消息的数组

*OnWriteComplete()* 将调用 COPCDA\_AsyncDlg 类的 *DisplayData* 方法来显示 OPC 服务器的错误消息：

```
{    m_pCDlgClass->DisplayData(pErrors[0]);  
    return S_OK;  
};
```

## InformAboutDialog

InformAboutDialog 可返回对话框的引用:

```
void InformAboutDialog (COPCDA_AsyncDlg *pCDlgClass)  
{    m_pCDlgClass = pCDlgClass;  
}
```

## GetQualityText

GetQualityText 提供了一个 CString 对象作为指定错误代码的错误消息:

```
CString GetQualityText (UINT qnr)  
{    CString qstr;  
    switch(qnr)  
    {        case OPC_QUALITY_BAD:  
            qstr = "BAD";  
            break;  
        case OPC_QUALITY_UNCERTAIN:  
            qstr = "UNCERTAIN";  
            break;  
        case OPC_QUALITY_GOOD:  
            qstr = "GOOD";  
            break;  
        case OPC_QUALITY_NOT_CONNECTED:  
            qstr = "NOT_CONNECTED";  
            break;  
        case OPC_QUALITY_DEVICE_FAILURE:  
            qstr = "DEVICE_FAILURE";  
            break;  
        case OPC_QUALITY_SENSOR_FAILURE:  
            qstr = "SENSOR_FAILURE";  
            break;  
        case OPC_QUALITY_LAST_KNOWN:  
            qstr = "LAST_KNOWN";  
            break;
```

```
case OPC_QUALITY_COMM_FAILURE:  
    qstr = "COMM_FAILURE";  
    break;  
  
case OPC_QUALITY_OUT_OF_SERVICE:  
    qstr = "OUT_OF_SERVICE";  
    break;  
  
case OPC_QUALITY_LAST_USABLE:  
    qstr = "LAST_USABLE";  
    break;  
  
case OPC_QUALITY_SENSOR_CAL:  
    qstr = "SENSOR_CAL";  
    break;  
  
case OPC_QUALITY_EGU_EXCEEDED:  
    qstr = "EGU_EXCEEDED";  
    break;  
  
case OPC_QUALITY_SUB_NORMAL:  
    qstr = "SUB_NORMAL";  
    break;  
  
case OPC_QUALITY_LOCAL_OVERRIDE:  
    qstr = "LOCAL_OVERRIDE";  
    break;  
  
default:qstr = "UNKNOWN ERROR";  
}  
  
return qstr;  
}
```

### 6.3.11 有关编写您自己的程序的注意事项

必须满足几项要求，您自己的程序才能使用自定义接口。请按照下面列出的步骤进行操作：

1. 启动 Visual C++ 开发环境。
2. 使用 MFC 类向导创建所需类型的项目。
3. 从示例程序中将文件“pre\_opc.h”和“pre\_opc.cpp”复制到项目目录中，并将文件添加至该项目（“项目 > 添加至项目 > 文件...”(Project > Add to project > Files...)）。  
这些文件包含 OPC 特定的定义和 include 语句。
4. 将下列语句添加至项目的所有实现文件中（扩展名为“\*.cpp”）： `#include pre_opc.h`

5. 在使用 *AddGroup()* 或 *GetErrorString()* 方法的实现文件中，添加下列语句： `#define LOCALE_ID 0x409`
6. 在主对话框中排列所需操作员界面元素，或创建合适的应用程序窗口并对相应的事件过程进行编程。
7. 如果要在 ATL 上使用异步操作，请执行回调对象。  
最佳方案是使用所需方法定义您自己的类。

## 6.4 以 VB.NET 编写的 OPC 自定义接口 (异步通信)

### 使用示例程序的要求

安装 SIMATIC NET 软件后，您会在硬盘的下列文件夹中找到该程序：

`“<安装路径>\SIEMENS\SIMATIC.NET\opc2\samples\dotnet\vb\async.net”`

.NET-Framework 版本 4.0

或更高版本必须安装在要运行示例程序的计算机上。此外，还需激活仿真连接，以便程序中使用的演示变量可用（请参见“激活仿真连接 (页 762)”部分）。

### 运行系统可调用封装

该示例配合使用 OPC Data Access V2.0 的自定义接口和 .NET Framework。.NET 应用程序管理型代码的 COM 接口通过 OPC 基金会的运行时可调用包装 (RCW) 来访问。

运行系统可调用封装的主要功能是封送 .NET 客户端与非托管 COM 对象间的调用。封送的意思是使存储区可用。

### 6.4.1 使用示例程序

该程序包括多个元素，操作员可使用这些元素触发以下动作：

操作员元素	效果
启动示例	启动程序
读取值或写入值	读取或写入值
组激活	激活或禁用组
停止示例	停止程序

## 6.4 以 VB.NET 编写的 OPC 自定义接口（异步通信）

启动程序时，仅启用“启动示例”(Start Sample) 按钮。单击此按钮后，程序将生成所需的 OPC 对象。然后，便可使用其它按钮。

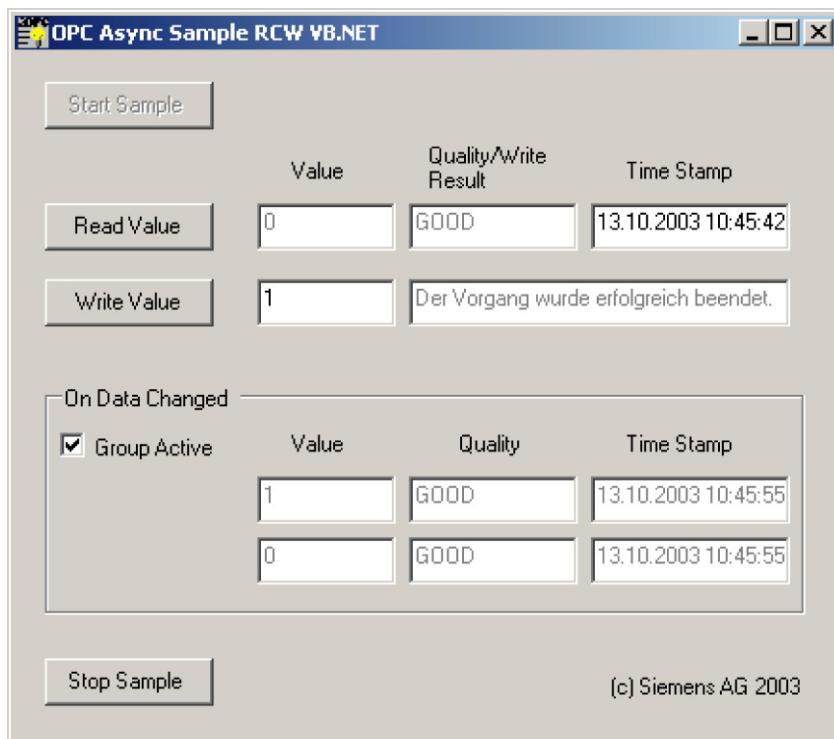


图 6-24 单击“启动示例”(Start Sample) 按钮后，以 VB.NET 编写并使用 .NET-Framework 的 OPC 自定义接口（异步通信）的示例程序的启动对话框

### 6.4.2 程序说明

#### 简介

各编程步骤的基本顺序与上一示例“C++ 中的 OPC 自定义接口（异步通信）[\(页 762\)](#)”相对应。其中包括建立与 OPC 服务器的连接、创建变量组、读取和写入数据项值。下表给出了程序要执行的步骤：

步骤	说明
1	选择 .NET 组件。
2	将 ProgID 转换为 CLSID。
3	建立与 OPC 服务器的连接。

步骤	说明
4	生成 OPC 组。
5	添加数据项。
6	请求 <b>IConnectionPointContainer</b> 接口。
7	请求 <b>IOPCAsyncIO2</b> 接口。
8	创建回调对象。
9	连接 OPC 服务器与客户端的回调对象。
10	执行所需的写入和读取操作。
11	接收 OPC 服务器的通知。
12	删除对象并释放内存。

## 步骤 1：选择 .NET 组件

要配合使用 OPC 数据访问的自定义接口和 .NET，必须在您的 Visual Basic 项目中使用上述运行时可调用包装 (RCW)。在该示例项目中，*OpcRcw.Da* 和 *OpcRcw.Comn* 组件已添加完毕。

如果它们未出现在选项中，也可使用“浏览”(*Browse*) 按钮从下列文件夹中进行选择：  
“<安装路径>\SIEMENS\SIMATIC.NET\opc2\bin”

## 6.4 以 VB.NET 编写的 OPC 自定义接口（异步通信）

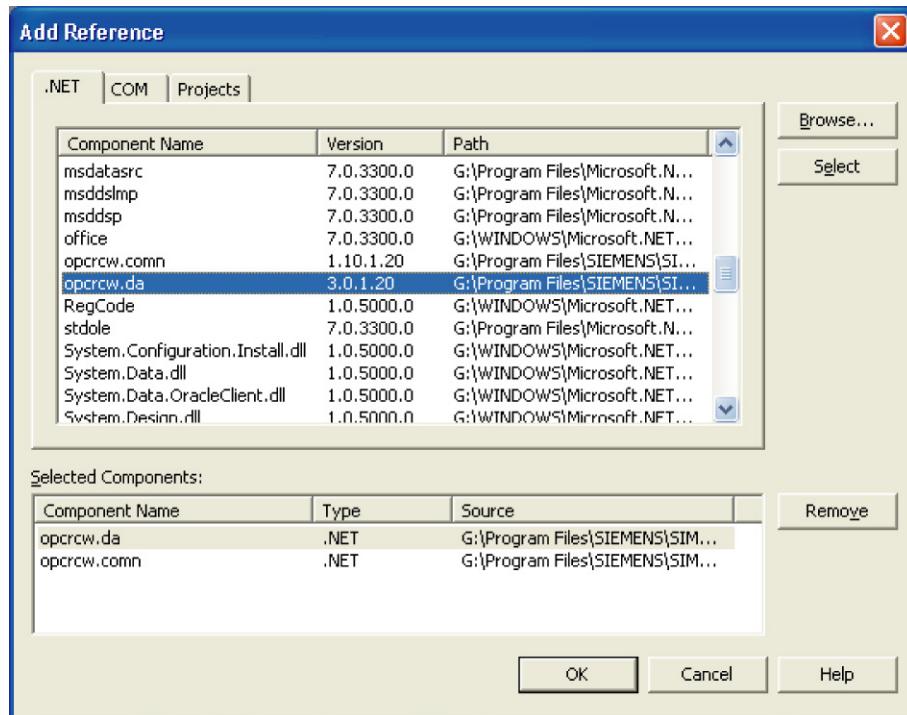


图 6-25 在 Visual Basic 项目中选择“运行时可调用包装”(RCW)

### 使用命令

```
Imports OpcRcw.Comn
Imports OpcRcw.Da
```

将 OPC 自定义接口的命名空间和方法与 .NET Framework 配合使用将十分简单。

### 步骤 2：将 ProgID 转换为类型

为了对其进行标识，每个 COM 服务器都有一个 *ProgID* 且该 *ProgID* 分配给世界上唯一的类型。使用 *GetTypeFromProgID()* 函数可实现上述操作。SIMATIC NET 的 OPC 服务器的 *ProgID* 是 L"OPC.SimaticNET":

```
Dim typeOfOPCserver As Type =
Type.GetTypeFromProgID(L"OPC.SimaticNET")
```

### 步骤 3：建立与 OPC 服务器的连接

*Activator*类的 *CreateInstance()* 函数生成一个具有之前指定类型的类的实例：

```
m_server = Activator.CreateInstance(typeOfOPCserver)
```

该程序段的结果是 IOPCServer 类型的接口变量 *m\_server*。

#### 步骤 4：创建 OPC 组

IOPCServer 接口具有用于创建组的 *AddGroup()* 方法：

```
m_server.AddGroup("MyOPCGroup",
    0,
    250,
    1,
    pTimeBias,
    pDeadband,
    LOCALE_ID,
    ServerGroup,
    RevisedUpdateRate,
    GetType(IOPCGroupStateMgt).GUID,
    m_group)
```

该程序段的结果是一个具有指定名称和所需属性的组。*AddGroup()* 还返回 *m\_group* 变量作为返回参数（组对象的接口），本例中为 *IOPCGroupStateMgt*。有了 *IOPCGroupStateMgt* 接口后，才允许使用 *SetState()* 方法。稍后需使用 *SetState()* 方法激活和禁用组。

通过简单的分配，*IOPCItemMgt* 类型的 *m\_item* 变量可源自 *m\_group* 接口变量。

```
m_item = m_group
```

#### 步骤 5：添加数据项

*IOPCItemMgt* 接口具有用于创建 OPC 数据项的 *AddItems()* 方法：

```
m_item.AddItems(2, itemdefs, Results, pErrors)
```

该程序段的结果是服务器添加两个具有 *itemdefs* 参数中所指定属性的数据项。此外，还会对结果结构 *Results* 的变量（服务器句柄、目标系统上数据项的数据类型等）进行初始化。

由于结果通过基础 COM 接口写入至非管理型内存，因此来自管理型 .NET 代码的访问必须利用封送函数：

## 6.4 以 VB.NET 编写的 OPC 自定义接口 (异步通信)

```
result = Marshal.PtrToStructure(pos, GetType(OPCITEMRESULT))
ServerHandle1 = result.hServer
pos = New IntPtr(pos.ToInt32() +
                  Marshal.SizeOf(GetType(OPCITEMRESULT)))
result = Marshal.PtrToStructure(pos, GetType(OPCITEMRESULT))
ServerHandle2 = result.hServer
```

客户端还必须确保非管理型内存已释放：

```
Marshal.FreeCoTaskMem(Results)
Marshal.FreeCoTaskMem(pErrors)
```

### 步骤 6：请求 **IConnectionPointContainer** 接口

*IConnectionPointContainer* 类型的 *m\_ConnectionContainer* 变量可源自 *m\_group* 变量。

```
m_ConnectionContainer = m_group
```

定位 *IConnectionPoint* 接口需要使用该接口。

### 步骤 7：请求 **IOPCAsyncIO2** 接口

*IOPCAsyncIO2* 类型的 *m\_asyncIO2* 变量可源自 *m\_group* 变量。

```
m_asyncIO2 = m_group
```

该接口提供用于异步读取和写入值的方法。

### 步骤 8：创建回调对象

为了允许 OPC 服务器返回异步操作的值，必须在客户端上执行 *IOPCDataCallback* 接口。

```
Implements IOPCDataCallback
```

使用 *IConnectionPointContainer* 接口的 *FindConnectionPoint()* 方法建立 OPC 服务器的 *IConnectionPoint* 和 *IOPCDataCallback* 接口间的连接。

该连接将为异步调用生成一个回调对象。

```
m_ConnectionContainer.FindConnectionPoint(
    GetType(IOPCDataCallback).GUID,
    m_ConnectionPoint)
```

### 步骤 9：连接 OPC 服务器与客户端的回调对象

OPC 服务器的回调对象与客户端应用程序间的连接通过 *Advise()* 方法来建立。返回变量 *m\_dwCookie* 是该连接的 ID。

```
m_ConnectionPoint.Advise(Me, m_dwCookie)
```

### 步骤 10：执行所需的写入或读取操作

在步骤 7 中生成的 *Read()* 和 *Write()* 方法可通过 **IOPCAsyncIO2** 接口来访问：

```
m_asyncIO2.Read(1, pServer, Transaction, CancelID, pErrors)
```

### 步骤 11：接收 OPC 服务器的通知

如果激活组中激活数据项的数据发生更改，服务器将调用回调对象的 *OnDataChange* 方法。读取操作完成后，服务器调用 *OnReadComplete()* 方法；写入操作完成后，服务器调用 *OnWriteComplete()* 方法。服务器将返回值以参数的形式传递至方法。

```
Overridable Sub OnDataChange(
    ByVal dwTransid As Integer,
    ByVal hGroup As Integer,
    ByVal hrMasterquality As Integer,
    ByVal hrMastererror As Integer,
    ByVal dwCount As Integer,
    ByVal phClientItems() As Integer,
    ByVal pvValues() As Object,
    ByVal pwQualities() As Short,
    ByVal pftTimeStamps() As OpcRcw.Da.FILETIME,
    ByVal pErrors() As Integer) Implements
    IOPCDataCallback.OnDataChange
```

## 步骤 12: 删除对象并释放内存

退出程序之前，必须删除已创建的 OPC 对象并释放为其保留的内存。

相关的函数是先前使用的接口的重要组成部分。*Unadvise()* 将终止 OPC 服务器与回调对象之间的连接：

```
m_ConnectionPoint.Unadvise(m_dwCookie)  
Marshal.ReleaseComObject(m_ConnectionPoint)  
m_item.RemoveItems(2, pItems, pErrors)  
....  
Marshal.FreeCoTaskMem(pErrors)  
m_server.RemoveGroup(ServerGroup, True)
```

### 释放内存

使用 COM 时，客户端有时需要释放服务器所请求的内存。相关规则如下：

- [out]: 将从服务器请求具有该属性的参数的内存。
- [in, out]: 将从客户端请求此类参数的内存。  
服务器可以再次释放内存并对其进行重新分配。
- [in]: 客户端将请求内存。服务器不负责释放内存。

在这所有三种情况下，客户端都应释放为非管理型 COM 对象所保留的内存。

## 6.5 以 C# 编写的 OPC 自定义接口（同步通信）

### 使用示例程序的要求

安装 SIMATIC NET 软件后，您会在硬盘的下列文件夹中找到该程序：

“<安装路径>\SIEMENS\SIMATIC.NET\opc2\samples\dotnet\C#\sync.net”

#### .NET-Framework 版本 4.0

或更高版本必须安装在要运行示例程序的计算机上。此外，还需激活仿真连接，以便程序中使用的演示变量可用（请参见“激活仿真连接（页 762）”部分）。

## 运行系统可调用封装

该示例配合使用 OPC Data Access V2.0 的自定义接口和 .NET Framework。.NET 应用程序管理型代码的 COM 接口通过 OPC 基金会的运行时可调用包装 (RCW) 来访问。

运行系统可调用封装的主要功能是封送 .NET 客户端与非托管 COM 对象间的调用。封送的意思是使存储区可用。

### 6.5.1 使用示例程序

该应用程序包含多个元素，操作员可使用这些元素触发以下动作：

操作员元素	效果
启动示例	启动应用程序
读取数据项或写入值	读取数据项或写入值
停止示例	退出应用程序

启动程序时，仅启用“启动示例”(Start Sample) 按钮。

单击此按钮后，应用程序将生成所需的 OPC 对象。随即会启用其它按钮。

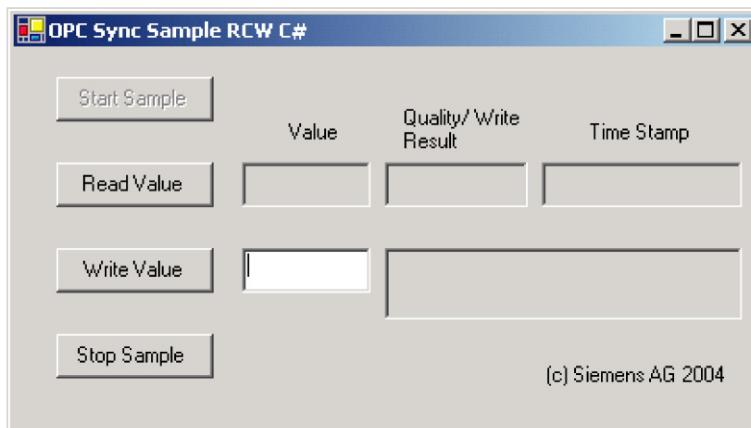


图 6-26 单击“启动示例”(Start Sample) 按钮后，以 C# 编写并使用 .NET-Framework 的 OPC 自定义接口（同步通信）的示例程序的启动对话框

## 6.5.2 程序说明

### 简介

各编程步骤的基本顺序与上一示例“C++ 中的 OPC 自定义接口（同步通信）(页 735)”相对应。其中包括建立与 OPC 服务器的连接、创建变量组、读取和写入数据项值。下表给出了程序要执行的步骤：

步骤	说明
1	选择 .NET 组件
2	将 ProgID 转换为 CLSID
3	建立与 OPC 服务器的连接
4	创建 OPC 组
5	添加数据项
6	请求 IOPCSyncIO 接口
7	执行所需的写入和读取操作
8	删除对象并释放内存

### 步骤 1：选择 .NET 组件

请参见“程序说明 (页 806)”部分中的步骤 1

### 步骤 2：将 ProgID 转换为 CLSID

为了对其进行标识，每个 COM 服务器都有一个 *ProgID* 且该 *ProgID* 分配给世界上唯一的类型。使用 *GetTypeFromProgID()* 函数可实现上述操作。SIMATIC NET 的 OPC 服务器的 *ProgID* 是 L"OPC.SimaticNET"：

```
Type svrComponenttyp = Type.GetTypeFromProgID("OPC.SimaticNET");
```

### 步骤 3：建立与 OPC 服务器的连接

Activator 类的 *CreateInstance()* 函数生成一个具有之前指定类型的类的实例：

```
pIOPCServer =
    (IOPCServer)Activator.CreateInstance(svrComponenttyp);
```

该程序段的结果是 IOPCServer 类型的接口变量 *pIOPCServer*。

## 步骤 4：创建 OPC 组

**IOPCServer** 接口具有用于创建组的 *AddGroup()* 方法：

```
    pIOPCServer.AddGroup(GroupName,
        0,
        dwRequestedUpdateRate,
        hClientGroup,
        hTimeBias.AddrOfPinnedObject(),
        hDeadband.AddrOfPinnedObject(),
        dwLCID,
        out pSvrGroupHandle,
        out pRevUpdateRate,
        ref out pobjGroup1)
```

### 注意

**hDeadband** 和 **hTimeBias** 的内存分配由 **GCHandle.Alloc()** 函数完成。

```
GCHandle hTimeBias;
hTimeBias = GCHandle.Alloc(timebias, GCHandleType.Pinned);

GCHandle hDeadband;
hDeadband = GCHandle.Alloc(deadband, GCHandleType.Pinned);
```

这些函数处理传送变量时间偏倚和死区（百分比死区）的非管理型内存。

当不再需要管理句柄时，必须按如下所示再次将其释放：

```
if (hTimeBias.IsAllocated) hTimeBias.Free();
if (hDeadband.IsAllocated) hDeadband.Free();
```

该程序段的结果是一个具有指定名称和所需属性的组。*AddGroup()* 还返回 **pobjGroup1** 变量作为返回参数，即组对象的接口。

使用类型调整调用 **IOPCGroupStateMgt**，可以只从返回的组接口 **pobjGroup1**

获取该接口。该调用简化了 COM 方法 *QueryInterface()*。

稍后需使用 **IOPCGroupStateMgt** 接口的 *SetState()* 方法来激活和禁用组。

```
pIOPCGroupStateMgt = (IOPCGroupStateMgt)pobjGroup1;
```

## 步骤 5：添加数据项

**IOPCItemMgt** 接口具有用于创建 OPC 数据项的 *AddItems()* 方法：

## 示例程序

### 6.5 以 C# 编写的 OPC 自定义接口（同步通信）

```
((IOPCItemMgt)pobjGroup1).AddItems(1, ItemDefArray, out pResults, out  
pErrors);
```

该程序段的结果是服务器添加一个具有 *itemdefs* 参数中所指定属性的数据项。

此外，还会对结果结构 *OPCITEMRESULT* 的变量（服务器句柄、目标系统上数据项的数据类型等）进行初始化。

由于结果通过基础 COM 接口写入至非管理型内存，因此来自管理型 .NET 代码的访问必须利用封送函数：

```
OPCITEMRESULT result =  
    (OPCITEMRESULT)Marshal.PtrToStructure(pResults,  
    typeof(OPCITEMRESULT));
```

客户端还必须确保非管理型内存已释放：

```
Marshal.FreeCoTaskMem(Results)  
Marshal.FreeCoTaskMem(pErrors)
```

### 步骤 6：请求 IOPCSyncIO 接口

*pIOPCSyncIO* 类型的 *pobjGroup1* 变量可源自 *pIOPCSyncIO* 变量。

```
pIOPCSyncIO2 = (IOPCSyncIO2)pobjGroup1;
```

该接口提供用于同步写入和读取值的方法。

### 步骤 7：执行所需的写入或读取操作

使用 *Read()* 和 *Write()* 方法，可通过 IOPCSyncIO 接口访问 OPC 数据项的值：

```
pIOPCSyncIO.Read(OPCDATASOURCE.OPC_DS_DEVICE, 1, nItemSvrID, out  
pItemValues, out pErrors);  
  
pIOPCSyncIO.Write(1, nItemSvrID, values, out pErrors);
```

由于结果通过基础 COM 接口写入至非管理型内存，因此来自管理型 .NET 代码的访问必须利用封送函数：

```
OPCITEMSTATE pItemState =  
    (OPCITEMSTATE)Marshal.PtrToStructure(pItemValues,  
    typeof(OPCITEMSTATE));
```

客户端还必须确保非管理型内存已释放：

```
Marshal.FreeCoTaskMem(pItemValues);  
Marshal.FreeCoTaskMem(pErrors);
```

### 步骤 8：删除对象并释放内存

退出程序前或单击“停止”(Stop) 后，必须删除已生成的 OPC 对象并释放所占用的内存。

```
Marshal.ReleaseComObject(pIOPCSyncIO);  
pIOPCServer.RemoveGroup(nSvrGroupID, 0);  
Marshal.ReleaseComObject(pobjGroup1);  
Marshal.ReleaseComObject(pIOPCServer);
```

#### 释放内存

使用 COM 时，客户端有时需要释放服务器所请求的内存。相关规则如下：

- **[out]**: 将从服务器请求具有该属性的参数的内存。
- **[in, out]**: 将从客户端请求此类参数的内存。  
服务器可以再次释放内存并对其进行重新分配。
- **[in]**: 客户端将请求内存。服务器不负责释放内存。

在这所有三种情况下，客户端都应释放为非管理型 COM 对象所保留的内存。

## 6.6 以 C# 编写的 OPC 自定义接口 (异步通信)

### 使用示例程序的要求

安装 SIMATIC NET 软件后，您会在硬盘的下列文件夹中找到该程序：

“<安装路径>\SIEMENS\SIMATIC.NET\opc2\samples\dotnet\C#\async.net”

.NET-Framework 版本 4.0

或更高版本必须安装在要运行示例程序的计算机上。此外，还需激活仿真连接，以便程序中使用的演示变量可用（请参见“激活仿真连接 (页 762)”部分）。

## 运行系统可调用封装

该示例配合使用 OPC Data Access V2.0 的自定义接口和 .NET Framework。.NET 应用程序管理型代码的 COM 接口通过 OPC 基金会的运行时可调用包装 (RCW) 来访问。

运行系统可调用封装的主要功能是封送 .NET 客户端与非托管 COM 对象间的调用。封送的意思是使存储区可用。

### 6.6.1 使用示例程序

该程序包括多个元素，操作员可使用这些元素触发以下动作：

操作员元素	效果
启动示例	启动程序
读取数据项或写入值	读取数据项或写入值
组激活	激活或禁用组
停止示例	停止程序

启动程序时，仅启用“启动示例”(Start Sample) 按钮。单击此按钮后，程序将生成所需的 OPC 对象。随即会启用其它按钮。

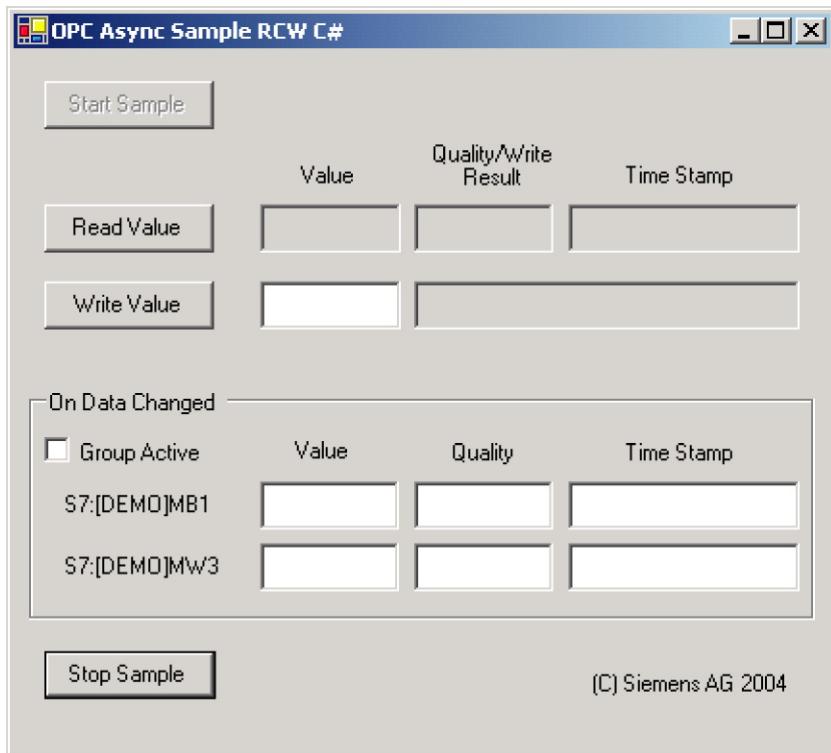


图 6-27 单击“启动示例”(Start Sample) 按钮后，以 C# 编写并使用 .NET-Framework 的 OPC 自定义接口（异步通信）的示例程序的启动对话框

## 6.6.2 程序说明

### 简介

各编程步骤的基本顺序与上一示例“以 VB.NET 编写的 OPC 自定义接口（异步通信）(页 805)”相对应。其中包括建立与 OPC

服务器的连接、创建变量组、读取和写入数据项值。下表给出了程序要执行的步骤：

步骤	说明
1	选择 .NET 组件
2	将 ProgID 转换为 CLSID
3	建立与 OPC 服务器的连接
4	创建 OPC 组
5	添加数据项
6	请求 IConnectionPointContainer 接口

步骤	说明
7	请求 IOPCAsynclO2 接口
8	创建回调对象
9	连接 OPC 服务器与客户端的回调对象
10	执行所需的写入和读取操作
11	接收 OPC 服务器的通知
12	删除对象并释放内存

### 步骤 1：选择 .NET 组件

请参见“程序说明 (页 806)”部分中的步骤 1

### 步骤 2：将 ProgID 转换为 CLSID

为了对其进行标识，每个 COM 服务器都有一个 *ProgID* 且该 *ProgID* 分配给世界上唯一的类型。使用 *GetTypeFromProgID()* 函数可实现上述操作。SIMATIC NET 的 OPC 服务器的 *ProgID* 是 L"OPC.SimaticNET":

```
Type svrComponenttyp = Type.GetTypeFromProgID(L"OPC.SimaticNET");
```

### 步骤 3：建立与 OPC 服务器的连接

Activator 类的 *CreateInstance()* 函数生成一个具有之前指定类型的类的实例：

```
pIOPCServer =
    (IOPCServer)Activator.CreateInstance(svrComponenttyp);
```

该程序段的结果是 *IOPCServer* 类型的接口变量 *pIOPCServer*。

### 步骤 4：创建 OPC 组

*IOPCServer* 接口具有用于创建组的 *AddGroup()* 方法：

```
pIOPCServer.AddGroup(GROUP_NAME,
    0,
    dwRequestedUpdateRate,
    hClientGroup,
    hTimeBias..AddrOfPinnedObject(),
    hDeadband.AddrOfPinnedObject(),
```

```

        dwLCID,
        out pSvrGroupHandle,
        out pRevUpdateRate,
        ref out pobjGroup1)

```

### 注意

`hDeadband` 和 `hTimeBias` 的内存分配由 `GCHandle.Alloc()` 函数完成。

```

GCHandle hTimeBias;
hTimeBias = GCHandle.Alloc(timebias,GCHandleType.Pinned);

GCHandle hDeadband;
hDeadband = GCHandle.Alloc(deadband,GCHandleType.Pinned);

```

这些函数处理传送变量时间偏倚和死区（百分比死区）的非管理型内存。

当不再需要管理句柄时，必须按如下所示再次将其释放：

```

if (hTimeBias.IsAllocated) hTimeBias.Free();
if (hDeadband.IsAllocated) hDeadband.Free();

```

该程序段的结果是一个具有指定名称和所需属性的组。 `AddGroup()` 还返回 `pobjGroup1` 变量作为返回参数，即组对象的接口。

使用类型调整调用 `IOPCGroupStateMgt`，可以只从返回的组接口 `pobjGroup1` 获取该接口。该调用的简化形式对应于 COM 方法 `QueryInterface()`。稍后需使用 `IOPCGroupStateMgt` 接口的 `SetState()` 方法激活和禁用组。

```
piOPCGroupStateMgt = (IOPCGroupStateMgt)pobjGroup1;
```

### 步骤 5：添加数据项

`IOPCItemMgt` 接口具有用于创建 OPC 数据项的 `AddItems()` 方法：

```
((IOPCItemMgt)pobjGroup1).AddItems(2,ItemDefArray,out
                                         pResults,out pErrors);
```

该程序段的结果是服务器添加两个具有 `itemdefs` 参数中所指定属性的数据项。

此外，还会对结果结构 `OPCITEMRESULT` 的变量（服务器句柄、目标系统上数据项的数据类型等）进行初始化。

由于结果通过基础 COM 接口写入至非管理型内存，因此来自管理型 .NET 代码的访问必须利用封送函数：

## 示例程序

### 6.6 以 C# 编写的 OPC 自定义接口（异步通信）

```
OPCITEMRESULT result = (OPCITEMRESULT)Marshal.PtrToStructure(pos,
    typeof(OPCITEMRESULT));
ItemSvrHandleArray[0] = result.hServer;

pos = new IntPtr(pos.ToInt32() +
    Marshal.SizeOf(typeof(OPCITEMRESULT)));

OPCITEMRESULT result = (OPCITEMRESULT)Marshal.PtrToStructure
    (pos, typeof(OPCITEMRESULT));
ItemSvrHandleArray[1] = result.hServer;
```

客户端还必须确保非管理型内存已释放：

```
Marshal.FreeCoTaskMem(Results);
Marshal.FreeCoTaskMem(pErrors);
```

### 步骤 6：请求 **IConnectionPointContainer** 接口

*IConnectionPointContainer* 类型的 *pIConnectionPointContainer* 变量可源自 *m\_group\_1* 变量。

```
pIConnectionPointContainer =
    (IConnectionPointContainer)pobjGroup1;
```

定位 *IConnectionPoint* 接口需要使用该接口。

### 步骤 7：请求 **IOPCAsyncIO2** 接口

*pIOPCAsyncIO2* 类型的 *pIOPCAsyncIO2* 变量可源自 *pobjGroup1* 变量。

```
pIOPCAsyncIO2 = (IOPCAsyncIO2)pobjGroup1;
```

该接口提供用于异步读取和写入值的方法。

### 步骤 8：创建回调对象

为了允许 OPC 服务器返回异步操作的值，必须在客户端上执行 *IOPCDataCallback* 接口。

```
public class OPCAsync : System.Windows.Forms.Form ,
    IOPCDataCallback
```

使用 `IConnectionPointContainer` 接口的 `FindConnectionPoint()` 方法建立 OPC 服务器的 `IConnectionPoint` 和 `IOPCDataCallback` 接口间的连接。

该连接将为异步调用生成一个回调对象。

```
Guid iid = typeof(IOPCDataCallback).GUID;
pIConnectionPointContainer.FindConnectionPoint(ref iid,
                                                out pIConnectionPoint);
```

### 步骤 9：连接 OPC 服务器与客户端的回调对象

OPC 服务器的回调对象与客户端应用程序间的连接通过 `Advise()` 方法来建立。返回变量 `dwCookie` 是该连接的 ID。

```
pIConnectionPoint.Advise(this, out dwCookie);
```

### 步骤 10：执行所需的写入或读取操作

在步骤 7 中生成的 `Read()` 和 `Write()` 方法可通过 `IOPCAsyncIO2` 接口来访问：

```
pIOPCAsyncIO2.Read(1, ItemSvrHandleArray, nTransactionID+1,
                    out nCancelid, out pErrors);
```

### 步骤 11：接收 OPC 服务器的通知

如果激活组中激活数据项的数据发生更改，服务器将调用回调对象的 `OnDataChange` 方法。读取操作完成后，服务器调用 `OnReadComplete()` 方法；写入操作完成后，服务器调用 `OnWriteComplete()` 方法。  
服务器将返回值以参数的形式传递至方法。

```
virtual void OnDataChange(
    Int32 dwTransid,
    Int32 hGroup,
    Int32 hrMasterquality,
    Int32 hrMastererror,
    Int32 dwCount,
    int[] phClientItems,
    object[] pvValues,
    short[] pwQualities,
    FILETIME[] pftTimeStamps,
    int[] pErrors,
)
```

## 步骤 12: 删除对象并释放内存

退出程序前或单击“停止”(Stop) 后，必须删除已生成的 OPC 对象并释放所占用的内存。

```
pIConnectionPoint.Unadvise(dwCookie);  
Marshal.ReleaseComObject(pIConnectionPoint);  
Marshal.ReleaseComObject(pIConnectionPointContainer);  
Marshal.ReleaseComObject(pIOPCAsyncIO2);  
Marshal.ReleaseComObject(pIOPCGroupStateMgt);  
Marshal.ReleaseComObject(pobjGroup1);  
Marshal.ReleaseComObject(pIOPCServer);
```

### 释放内存

使用 COM 时，客户端有时需要释放服务器所请求的内存。相关规则如下：

- [out]: 将从服务器请求具有该属性的参数的内存。
- [in, out]: 将从客户端请求此类参数的内存。  
服务器可以再次释放内存并对其进行重新分配。
- [in]: 客户端将请求内存。服务器不负责释放内存。

在这所有三种情况下，客户端都应释放为非管理型 COM 对象所保留的内存。

## 6.7 以 C# 编写的 OPC XML 接口

该示例以 C# 编程语言编写，其使用 OPC 数据访问的 XML 接口。其中包含与 Web 服务建立连接及执行读取和写入作业的方法。

### 使用示例程序的要求

首先，必须在要运行示例程序的计算机上激活 S7 仿真模块和 S7 协议。

此外，还必须建立 Web 服务。

### 6.7.1 使用示例程序

#### 启动程序

启动程序后，除“启动示例”(Start Sample) 外，其它所有按钮均禁用。

单击此按钮可建立与 OPC XML Web 服务的连接。如果 Web 服务可用，对话框将显示服务器的相关信息。否则，将显示错误消息。

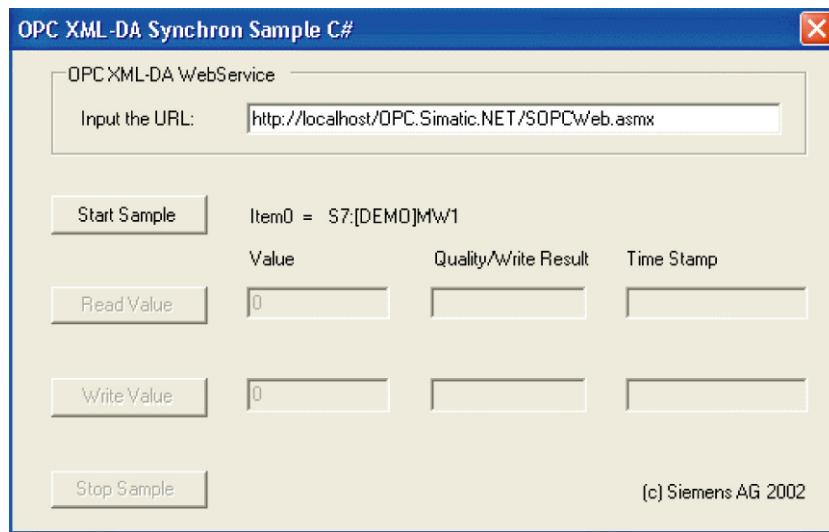


图 6-28 单击“启动示例”(Start Sample) 按钮前，通过 XML 接口进行 OPC 数据访问的示例程序 (C#) 的启动对话框

## 读取值

单击“读取值”(Read Value) 按钮可读取 *S7:[DEMO]MW1* 数据项的值。

如果读取作业执行成功，程序将在“读取值”(Read Value)

按钮旁的文本框中显示数据项的值、质量相关信息和时间戳：

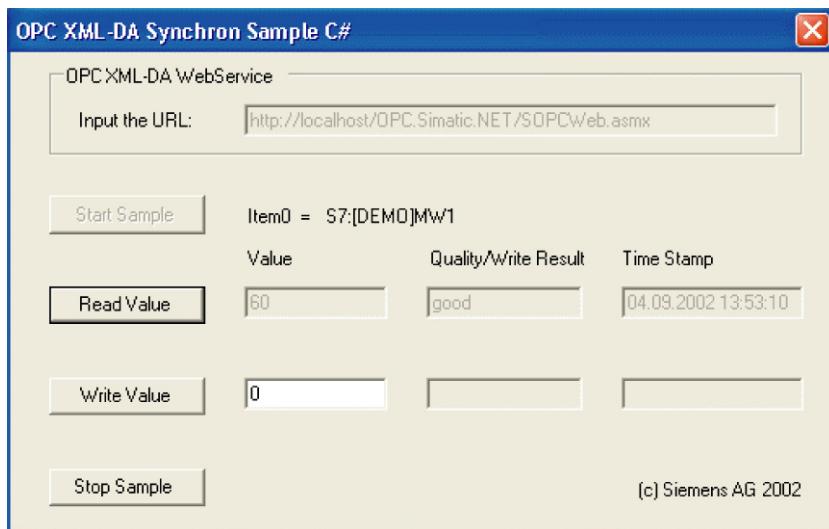


图 6-29 在单击“读取值”(Read Value) 按钮后显示读取值

## 写入值

在“写入值”(Write Value) 按钮旁的文本框中输入要写入的值，然后单击“写入值”(Write Value) 按钮。如果写入作业执行无误，程序将在“质量/写入结果”(Quality/Write Result) 文本框中显示值质量的相关信息、在“时间戳”(TimeStamp) 文本框中显示时间戳。

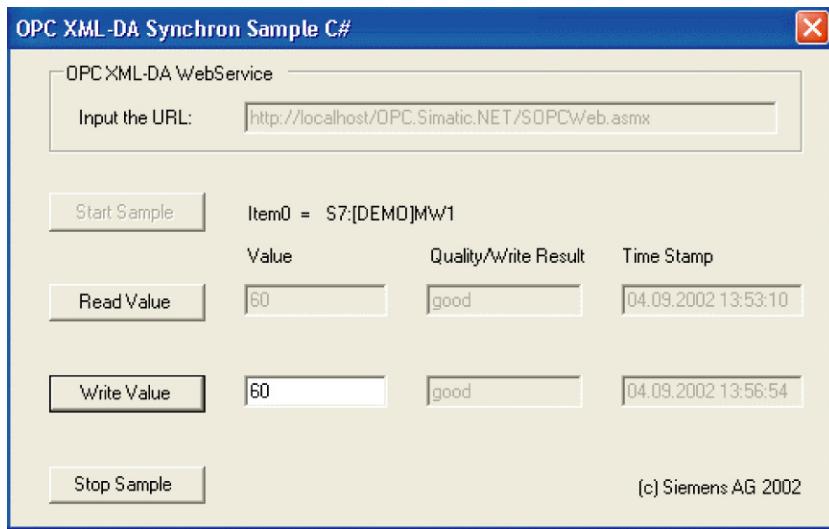


图 6-30 单击“写入值”(Write Value) 按钮后写入值的值质量和时间戳的显示

## 停止程序

单击“停止示例”(Stop Sample) 按钮，关闭程序。这会禁用“读取值”(Read Value)、“写入值”(Write Value)、和“停止示例”(Stop Sample) 按钮。“启动示例”(Start Sample) 按钮和 URL 文本框会再次启用。

## 6.7.2 向项目添加 Web 服务

### Visual Studio .NET 中的设置

为使示例程序使用 OPC XML 接口的 Web 服务，Visual Studio .NET 项目中应包含“Web 引用”。

如果要使用 OPC XML 接口自行创建程序，需完成以下步骤：

1. 调用“Microsoft Visual Studio”中的“项目 > 添加 Web 引用”(Project > Add Web Reference) 菜单。
2. 单击“添加 Web 引用”(Add Web Reference) 对话框中的“高级”(Advanced) 按钮。
3. 单击“服务引用设置”(Service Reference Settings) 对话框中的“添加 Web 引用”(Add Web Reference) 按钮。

## 6.7 以 C# 编写的 OPC XML 接口

4. 在“URL”输入框中输入 SIMATIC NET OPC XML Web 服务的 Web 地址。该 URL 如下：

“<http://<计算机地址>/<OPC SIMATIC NET Web 服务的名称>/SOPCWeb.asmx?wsdl>”

5. 单击“添加引用”(Add Reference) 按钮。

在“OPCXML\_DataAccess Description”窗口中找到可用的方法。

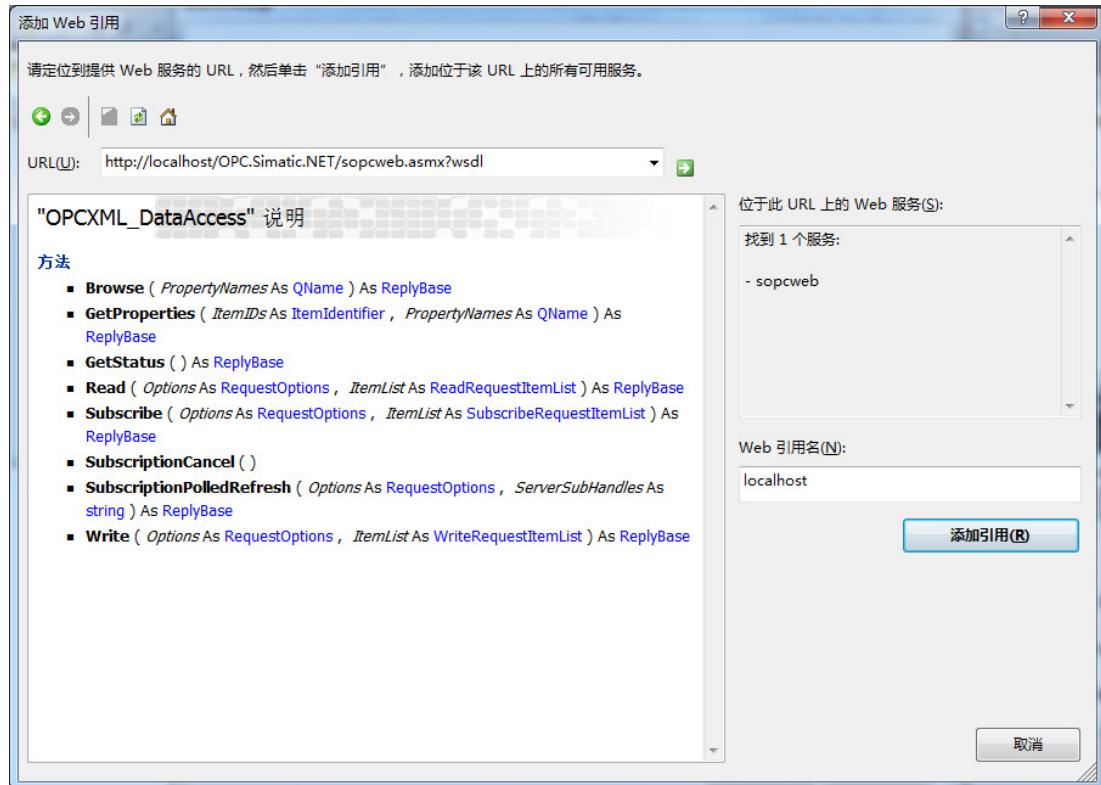


图 6-31 列出 WSDL 中所定义方法的“添加 Web 引用”(Add Web Reference) 窗口

### 6.7.3 MainForm 类

#### 程序代码

示例程序的所有方法均在 *MainForm* 类中定义。首先，声明对话框的元素。

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
```

```
using System.Data;
using System.Net;
namespace opcxml_da_sync
{
    using localhost;
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class MainForm : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Button Button_Start_Sample;
        private System.Windows.Forms.Button Button_Stop_Sample;
        private System.Windows.Forms.Button Button_Read_Value;
        private System.Windows.Forms.Button Button_Write_Value;
        private System.Windows.Forms.TextBox Edit_URL;
        private System.Windows.Forms.Label Label_URL;
        private System.Windows.Forms.Label Label_Item0;
        private System.Windows.Forms.Label Label_Item_Value;
        private System.Windows.Forms.Label Label_Siemens;
        private System.Windows.Forms.Label Label_Value;
        private System.Windows.Forms.Label Label_Quality;
        private System.Windows.Forms.Label Label_Timestamp;
        private System.Windows.Forms.TextBox Edit_Read_Value;
        private System.Windows.Forms.TextBox Edit_Write_Value;
        private System.Windows.Forms.TextBox Edit_Read_Quality;
        private System.Windows.Forms.TextBox Edit_Write_Quality;
        private System.Windows.Forms.TextBox Edit_Read_TimeStamp;
        private System.Windows.Forms.TextBox Edit_Write_TimeStamp;
        private System.Windows.Forms.GroupBox GroupBox_URL;
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components = null;
```

Web 服务是 OPCXML\_DataAccess 类的实例:

```
// proxy class for the webservice
private OPCXML_DataAccess m_OPCTML_DataAccess;
```

数据项的名称存储在 **m\_strItemName** 变量中：

```
// ItemID used in this sample
private string m_strItemName = "S7:[DEMO]MW1";
```

#### 6.7.4 MainForm 的构造函数和 Dispose 方法

##### 程序代码

对话框类的构造函数调用 *InitializeComponent()* 方法。

对话框的所有元素均在此处创建。使用窗体设计器创建对话框时，Visual Studio .NET 会自动创建 *InitializeComponent()* 的内容。

```
public MainForm()
{
    //
    // Required for Windows Form Designer support
    //
    InitializeComponent();
    //
    // TODO: Add any constructor code after
    // InitializeComponent call
```

数据项名称将出现在对话框中，因为 *System.Windows.Forms.Label* 类的实例通过 **m\_strItemName** 变量的内容进行初始化：

```
Label_Item_Value.Text = m_strItemName;
}
```

*Dispose()* 方法会在程序结束时释放所有已用资源：

```
/// <summary>
/// Clean up any resources being used.
/// </summary>

protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
```

```
        }

    }

    base.Dispose( disposing );
}

}
```

## 6.7.5 创建对话框元素

### 程序代码

用户使用窗体设计器创建应用程序的主对话框时，Visual Studio .NET 会自动创建以下部分。

这包括对话框中所有元素的相关信息，如大小、位置和按钮的标签设定。此部分还指定单击按钮后所执行的事件过程。

```
#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.Button_Start_Sample = new System.Windows.Forms.Button();
    this.Button_Stop_Sample = new System.Windows.Forms.Button();
    this.Button_Read_Value = new System.Windows.Forms.Button();
    this.Button_Write_Value = new System.Windows.Forms.Button();
    this.Edit_URL = new System.Windows.Forms.TextBox();
    this.Label_URL = new System.Windows.Forms.Label();
    this.Label_Item0 = new System.Windows.Forms.Label();
    this.Label_Item_Value = new System.Windows.Forms.Label();
    this.Label_Siemens = new System.Windows.Forms.Label();
    this.Label_Value = new System.Windows.Forms.Label();
    this.Label_Quality = new System.Windows.Forms.Label();
    this.Label_Timestamp = new System.Windows.Forms.Label();
    this.Edit_Read_Value = new System.Windows.Forms.TextBox();
    this.Edit_Write_Value = new System.Windows.Forms.TextBox();
    this.Edit_Read_Quality = new System.Windows.Forms.TextBox();
    this.Edit_Write_Quality = new System.Windows.Forms.TextBox();
}
```

## 6.7 以 C# 编写的 OPC XML 接口

```
this.Edit_Read_TimeStamp = new System.Windows.Forms.TextBox();
this.GroupBox_URL = new System.Windows.Forms.GroupBox();
this.Edit_Write_TimeStamp = new System.Windows.Forms.TextBox();
this.SuspendLayout();
//
// Button_Start_Sample
//
this.Button_Start_Sample.Location = new System.Drawing.
    Point(24, 88);
this.Button_Start_Sample.Name = "Button_Start_Sample";
this.Button_Start_Sample.Size =
    new System.Drawing. Size(96, 24);
this.Button_Start_Sample.TabIndex = 0;
this.Button_Start_Sample.Text = "Start Sample";
this.Button_Start_Sample.Click += new System.EventHandler(
    this.Button_Start_Sample_Click);
//
// Button_Stop_Sample
//
this.Button_Stop_Sample.Enabled = false;
this.Button_Stop_Sample.Location = new System.Drawing.
    Point(24, 256);
this.Button_Stop_Sample.Name = "Button_Stop_Sample";
this.Button_Stop_Sample.Size = new System.Drawing. Size(96, 23);
this.Button_Stop_Sample.TabIndex = 1;
this.Button_Stop_Sample.Text = "Stop Sample";
this.Button_Stop_Sample.Click += new System.EventHandler(
    this.Button_Stop_Sample_Click);
//
// Button_Read_Value
//
this.Button_Read_Value.Enabled = false;
this.Button_Read_Value.Location = new System.Drawing.
    Point(24, 144);
this.Button_Read_Value.Name = "Button_Read_Value";
this.Button_Read_Value.Size = new System.Drawing. Size(96, 23);
this.Button_Read_Value.TabIndex = 2;
this.Button_Read_Value.Text = "Read Value";
this.Button_Read_Value.Click += new System.EventHandler(
```

```
this.Button_Read_Value_Click);  
//  
// Button_Write_Value  
//  
this.Button_Write_Value.Enabled = false;  
this.Button_Write_Value.Location = new System.Drawing.  
    Point(24, 200);  
this.Button_Write_Value.Name = "Button_Write_Value";  
this.Button_Write_Value.Size = new System.Drawing. Size(96, 23);  
this.Button_Write_Value.TabIndex = 3;  
this.Button_Write_Value.Text = "Write Value";  
this.Button_Write_Value.Click += new System.EventHandler(  
    this.Button_Write_Value_Click);  
//  
// Edit_URL  
//  
this.Edit_URL.Location = new System.Drawing.Point(144, 32);  
this.Edit_URL.Name = "Edit_URL";  
this.Edit_URL.Size = new System.Drawing.Size(328, 20);  
this.Edit_URL.TabIndex = 4;  
this.Edit_URL.Text =  
    "http://localhost/OPC.Simatic.NET/SOPCWeb.asmx";  
//  
// Label_URL  
//  
this.Label_URL.Location = new System.Drawing.Point(40, 35);  
this.Label_URL.Name = "Label_URL";  
this.Label_URL.Size = new System.Drawing.Size(96, 24);  
this.Label_URL.TabIndex = 5;  
this.Label_URL.Text = "Input the URL:";  
//  
// Label_Item0  
//  
this.Label_Item0.Location = new System.Drawing. Point(144, 94);  
this.Label_Item0.Name = "Label_Item0";  
this.Label_Item0.Size = new System.Drawing.Size(48, 23);  
this.Label_Item0.TabIndex = 6;  
this.Label_Item0.Text = "Item0 = ";  
//
```

## 6.7 以 C# 编写的 OPC XML 接口

```
// Label_Item_Value
//
this.Label_Item_Value.Location = new System.Drawing.
    Point(192, 94);
this.Label_Item_Value.Name = "Label_Item_Value";
this.Label_Item_Value.Size = new System.Drawing. Size(296, 23);
this.Label_Item_Value.TabIndex = 7;
//
// Label_Siemens
//
this.Label_Siemens.Location = new System.Drawing.
    Point(384, 262);
this.Label_Siemens.Name = "Label_Siemens";
this.Label_Siemens.Size = new System.Drawing.Size(120, 16);
this.Label_Siemens.TabIndex = 8;
this.Label_Siemens.Text = "(c) Siemens AG 2003";
//
// Label_Value
//
this.Label_Value.Location = new System.Drawing. Point(144, 120);
this.Label_Value.Name = "Label_Value";
this.Label_Value.Size = new System.Drawing.Size(40, 16);
this.Label_Value.TabIndex = 9;
this.Label_Value.Text = "Value";
//
// Label_Quality
//
this.Label_Quality.Location = new System.Drawing.
    Point(256, 120);
this.Label_Quality.Name = "Label_Quality";
this.Label_Quality.Size = new System.Drawing.Size(104, 23);
this.Label_Quality.TabIndex = 10;
this.Label_Quality.Text = "Quality/Write Result";
//
// Label_Timestamp
//
this.Label_Timestamp.Location = new System.Drawing.
    Point(374, 120);
this.Label_Timestamp.Name = "Label_Timestamp";
```

```
this.Label_Timestamp.TabIndex = 11;
this.Label_Timestamp.Text = "Time Stamp";
//
// Edit_Read_Value
//
this.Edit_Read_Value.Enabled = false;
this.Edit_Read_Value.Location = new System.Drawing.
    Point(144, 144);
this.Edit_Read_Value.Name = "Edit_Read_Value";
this.Edit_Read_Value.Size = new System.Drawing. Size(88, 20);
this.Edit_Read_Value.TabIndex = 12;
this.Edit_Read_Value.Text = "0";
//
// Edit_Write_Value
//
this.Edit_Write_Value.Enabled = false;
this.Edit_Write_Value.Location = new System.Drawing.
    Point(144, 200);
this.Edit_Write_Value.Name = "Edit_Write_Value";
this.Edit_Write_Value.Size = new System.Drawing. Size(88, 20);
this.Edit_Write_Value.TabIndex = 13;
this.Edit_Write_Value.Text = "0";
//
// Edit_Read_Quality
//
this.Edit_Read_Quality.Enabled = false;
this.Edit_Read_Quality.Location = new System.Drawing.
    Point(256, 144);
this.Edit_Read_Quality.Name = "Edit_Read_Quality";
this.Edit_Read_Quality.Size = new System.Drawing. Size(96, 20);
this.Edit_Read_Quality.TabIndex = 14;
this.Edit_Read_Quality.Text = "";
//
// Edit_Write_Quality
//
this.Edit_Write_Quality.Enabled = false;
this.Edit_Write_Quality.Location = new System.Drawing.
    Point(256, 200);
this.Edit_Write_Quality.Name = "Edit_Write_Quality";
```

## 6.7 以 C# 编写的 OPC XML 接口

```
this>Edit_Write_Quality.Size = new System.Drawing. Size(96, 20);
this>Edit_Write_Quality.TabIndex = 15;
this>Edit_Write_Quality.Text = "";
//
// Edit_Read_TimeStamp
//
this>Edit_Read_TimeStamp.Enabled = false;
this>Edit_Read_TimeStamp.Location = new System.Drawing.
    Point(376, 144);
this>Edit_Read_TimeStamp.Name = "Edit_Read_TimeStamp";
this>Edit_Read_TimeStamp.Size = new System.Drawing.
    Size(112, 20);
this>Edit_Read_TimeStamp.TabIndex = 16;
this>Edit_Read_TimeStamp.Text = "";
//
// GroupBox_URL
//
this.GroupBox_URL.Location = new System.Drawing. Point(24, 8);
this.GroupBox_URL.Name = "GroupBox_URL";
this.GroupBox_URL.Size = new System.Drawing.Size(464, 56);
this.GroupBox_URL.TabIndex = 17;
this.GroupBox_URL.TabStop = false;
this.GroupBox_URL.Text = "OPC XML-DA WebService";
//
// Edit_Write_TimeStamp
//
this>Edit_Write_TimeStamp.Enabled = false;
this>Edit_Write_TimeStamp.Location = new System.Drawing.
    Point(376, 200);
this>Edit_Write_TimeStamp.Name = "Edit_Write_TimeStamp";
this>Edit_Write_TimeStamp.Size = new System.
    Drawing.Size(112, 20);
this>Edit_Write_TimeStamp.TabIndex = 18;
this>Edit_Write_TimeStamp.Text = "";
//
// MainForm
//
this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
this.ClientSize = new System.Drawing.Size(506, 293);
```

```
this.Controls.Add(this.Edit_Write_TimeStamp);
this.Controls.Add(this.Edit_Read_TimeStamp);
this.Controls.Add(this.Edit_Write_Quality);
this.Controls.Add(this.Edit_Read_Quality);
this.Controls.Add(this.Edit_Write_Value);
this.Controls.Add(this.Edit_Read_Value);
this.Controls.Add(this.Edit_URL);
this.Controls.Add(this.Label_Timestamp);
this.Controls.Add(this.Label_Quality);
this.Controls.Add(this.Label_Value);
this.Controls.Add(this.Label_Siemens);
this.Controls.Add(this.Label_Item_Value);
this.Controls.Add(this.Label_Item0);
this.Controls.Add(this.Label_URL);
this.Controls.Add(this.Button_Write_Value);
this.Controls.Add(this.Button_Read_Value);
this.Controls.Add(this.Button_Stop_Sample);
this.Controls.Add(this.Button_Start_Sample);
this.Controls.Add(this.GroupBox_URL);
this.FormBorderStyle =
    System.Windows.Forms.FormBorderStyle.FixedDialog;
this.MaximizeBox = false;
this.MinimizeBox = false;
this.Name = " MainForm";
this.Text = "OPC XML-DA Synchron Sample C#";
this.ResumeLayout(false);
}
#endregion
```

## 6.7.6 Main 方法

### 程序代码

Main 方法是示例程序的起始点。对话框类的实例通过调用 Application.Run 来生成和启动。

```

/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new MainForm());
}

```

### 6.7.7 Button\_Start\_Sample\_Click 方法

#### 程序代码

该方法在单击“启动示例”(Start Sample) 按钮后执行。OPC XML Web 服务由 OPCXML\_DataAccess 类的实例表示。

```

/*
 *-----*
 | Name:  Button_Start_Sample_Click
 | Desc:  Handler is called, when button "Start Sample"
 |         is pressed
 | Notes: Create an instance of the proxy class of the
 |         webservice
 *-----*/
private void Button_Start_Sample_Click(object sender, System.EventArgs e)
{
    try
    {
        if (m_OPCTML_DataAcess == null)
        {
            m_OPCTML_DataAcess = new OPCXML_DataAccess ();
            m_OPCTML_DataAcess.Timeout = 10000; //Timeout 10 sec.
        }
    }
}

```

该程序通过调用 *WebRequest* 类的方法来检查与 Web 服务的连接。如果 Web 服务不可用，将触发异常。

```

// Checking the connection to the WebService
// The WebRequest class throws a WebException when
// errors occur while accessing an Internet resource

```

```
WebRequest myRequest = WebRequest.Create(Edit_URL.Text);
```

```
WebResponse myResponse = myRequest.GetResponse();
myResponse.Close();
```

程序使用 *GetStatus* 方法查询服务器状态。这是第一次使用 XML 接口的方法。

```
// Assigning the url to the proxy class for the
// web service
m_OPCTXML_DataAccess.Url = Edit_URL.Text;

//Checking the webservice status
ServerStatus Status;
ReplyBase replay = m_OPCTXML_DataAccess.GetStatus ("en", "1", out
Status);
```

如果服务器运行正常，程序将显示含有各种信息的对话框。

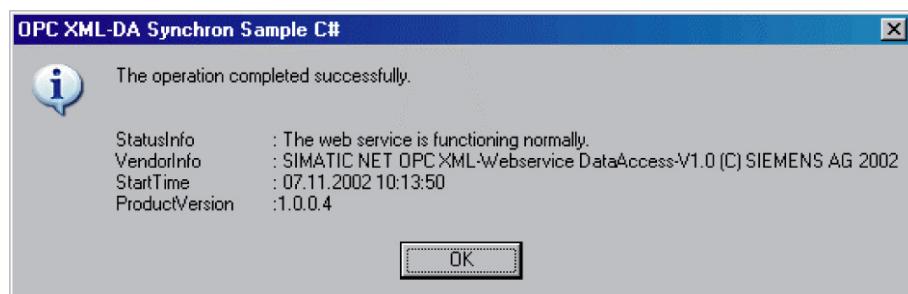


图 6-32 成功启动 Web 服务后的信息对话框

```
if (replay.ServerState == serverState.running)
{
    string strText =
        """The operation completed successfully.\n\n";
    if (Status.StatusInfo != null)
    {
        strText+= "\nStatusInfo\t:t: " + Status.StatusInfo;
    }
    if (Status.VendorInfo != null)
    {
        strText+= "\nVendorInfo\t:t: " + Status.VendorInfo;
    }
    strText+= "\nStartTime\t:t: " +
        Status.StartTime.ToString();
    if (Status.ProductVersion != null)
    {
        strText+= "\nProductVersion\t:t:"
```

## 6.7 以 C# 编写的 OPC XML 接口

```
+ Status.ProductVersion;  
}  
  
MessageBox.Show (this, strText, this.Text,  
MessageBoxButtons.OK, MessageBoxIcon.Information);
```

用于输入 URL 的文本框和“启动示例”(Start Sample) 按钮将禁用。  
其余所有按钮均会启用。

```
Edit_URL.Enabled = false;  
Button_Start_Sample.Enabled = false;  
Edit_Write_Value.Enabled = true;  
Button_Read_Value.Enabled = true;  
Button_Write_Value.Enabled = true;  
Button_Stop_Sample.Enabled = true;  
}
```

如果 Web 服务发生错误，这将在对话框中显示出来：

```
else  
{    string strText =  
    "The operation is not completed successfully.\n\n";  
    MessageBox.Show (this, strText, this.Text,  
    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);  
}  
}
```

如果程序执行期间发生错误，将在捕捉块中对此进行处理。

```
catch (Exception excep)  
{    MessageBox.Show (this, excep.Message, this.Text,  
    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);  
}  
}
```

## 6.7.8 Button\_Stop\_Sample\_Click 方法

### 程序代码

该方法在单击“停止示例”(Stop Sample) 按钮后执行。

仅对话框的外观会发生更改，可再次编辑 URL 的文本框，而且可单击“启动示例”(Start Sample) 按钮。所有其它按钮都将被禁用。

```
/*
| Name: Button_Stop_Sample_Click
| Desc: Handler is being called, when button "Stop Sample"
|       has been pressed
-----
private void Button_Stop_Sample_Click(object sender, System.EventArgs e)
{
    Edit_URL.Enabled      = true;
    Button_Start_Sample.Enabled = true;
    Edit_Write_Value.Enabled = false;
    Button_Read_Value.Enabled = false;
    Button_Write_Value.Enabled = false;
    Button_Stop_Sample.Enabled = false;
}
```

## 6.7.9 Button\_Read\_Value\_Click 方法

### 程序代码

该方法在单击“读取值”(Read Value) 按钮后执行。

程序首先生成读取作业所需的全部对象、*ReadRequestItemLists* 类型的数组和 *ReadRequestItem* 类型的数组（每个数组都有一个元素）。除此之外，还需要使用 *RequestOptions* 类和 *OPCError* 类的实例作为返回参数。

```
/*
| Name: Button_Read_Value_Click
| Desc: Handler is called, when button "Read Value"
|       is pressed
| Notes: initiates a sync read request
-----
private void Button_Read_Value_Click(object sender,
                                    System.EventArgs e)
```

```

{    try
{
    Edit_Read_Value.Text = "0";
    Edit_Read_Quality.Text = "";
    Edit_Read_TimeStamp.Text = "";
    /// <summary>
    /// Make a new ItemList for a ReadRequest
    /// </summary>
    ReadRequestItemList ItemLists = new ReadRequestItemList();
    ItemLists.Items = new ReadRequestItem[1];
    ItemLists.Items[0] = new ReadRequestItem();
    ItemLists.Items[0].ItemPath = "";
    ItemLists.Items[0].ItemName = m_strItemName;
    RequestOptions opt = new RequestOptions();
    ReplyItemList ItemValues;
    OPCError[] Errors;
}

```

XML 接口的 *Read* 方法使用之前定义的参数来调用。

如果可以读取值，对话框的“值”(Value) 文本框中将显示该值。否则，将显示“0”。

```

m_OPCTXML_DataAccess.Read (opt, ItemLists, out ItemValues,
                           out Errors);
/// <summary>
/// Assign the returned values to the TextBoxes
/// </summary>
if (ItemValues.Items[0].Value != null)
{
    Edit_Read_Value.Text = ItemValues.Items[0].Value.ToString();
}

else
{
    Edit_Read_Value.Text = "0";
}

```

时间戳值和值质量也会显示在相应的文本框中。

```

if (ItemValues.Items[0].TimestampSpecified)
{
    Edit_Read_TimeStamp.Text =

```

```
    ItemValues.Items[0].Timestamp.ToString();  
}  
  
else  
{    Edit_Read_TimeStamp.Text = "";  
}  
  
  
if(ItemValues.Items[0].Quality!=null)  
{    Edit_Read_Quality.Text =  
    ItemValues.Items[0].Quality.QualityField.ToString();  
}  
  
else  
{    Edit_Read_Quality.Text = "";  
}  
  
/// <summary>  
/// Show Errors in a Message Box  
/// </summary>  
if(Errors.Length>0)  
{    MessageBox.Show (this,  
    Errors[0].Text,  
    this.Text,  
    MessageBoxButtons.OK,  
    MessageBoxIcon.Exclamation);  
}
```

程序错误在捕捉块中进行处理。

```
catch(Exception excep)  
{    MessageBox.Show (this, excep.Message, this.Text,  
    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);  
}  
}
```

### 6.7.10 Button\_Write\_Value\_Click 方法

#### 程序代码

该方法在单击“写入值”(Write Value) 按钮后执行。

程序首先生成写入作业所需的全部对象、*WriteRequestItemList* 类型的数组和 *ItemValue* 类型的数组（每个数组都有一个元素）。读取值以 *ReplyItemList* 类型的参数返回。除此之外，还需要使用 *RequestOptions* 类和 *OPCError* 类的实例作为返回参数。

```
/*
| Name: Button_Write_Value_Click
| Desc: Handler is called, when button "Write Value"
|       is pressed
| Notes: initiates a sync read request
-----
private void Button_Write_Value_Click(object sender,
                                      System.EventArgs e)
{
    try
    {
        Edit_Write_Quality.Text = "";
        /// <summary>
        /// Make a new ItemList for a WriteRequest
        /// </summary>
        WriteRequestItemList ItemLists =
            new WriteRequestItemList();
        ItemLists.Items = new ItemValue[1];
        ItemLists.Items[0] = new ItemValue();
        ItemLists.Items[0].ItemPath = "";
        ItemLists.Items[0].ItemName = m_strItemName;
        ItemLists.Items[0].Value = System.Convert.ToInt32(Edit_Write_
            Value.Text);
        ItemLists.Items[0].TimestampSpecified = false;
        RequestOptions opt = new RequestOptions();
        ReplyItemList ItemValues;
        OPCError[] Errors;
```

XML 接口的 *Write* 方法使用之前定义的参数来调用。

如果可以写入值，对话框的“值”(Value) 文本框中将显示该值。否则，将显示“0”。

```
ReplyBase replay = m_OPCTXML_DataAccess.Write(
    opt,
```

```
        ItemLists,
        true,
        out ItemValues,
        out s);

    /// <summary>
    /// Assign the returned values to the TextBoxes
    /// </summary>
    if (ItemValues.Items[0].Value != null)
    {
        Edit_Write_Value.Text =
            ItemValues.Items[0].Value.ToString();
    }
    else
    {
        Edit_Write_Value.Text = "0";
    }
```

返回的时间戳值和值质量也会显示在相应的文本框中。

```
if (ItemValues.Items[0].TimestampSpecified)
{
    Edit_Write_TimeStamp.Text =
        ItemValues.Items[0].Timestamp.ToString();
}
else
{
    Edit_Write_TimeStamp.Text = "";
}
if (ItemValues.Items[0].Quality!=null)
{
    Edit_Write_Quality.Text =
        ItemValues.Items[0].Quality.QualityField.ToString();
}
else
{
    Edit_Write_Quality.Text = "";
}

    /// <summary>
    /// Show Errors in a Message Box
    /// </summary>
    if (Errors.Length>0)
    {
        MessageBox.Show (this,
                        Errors[0].Text,
                        this.Text,
                        MessageBoxButtons.OK,
```

## 6.8 以 C++ 编写的 OPC 报警和事件自定义接口

```

        MessageBoxIcon.Exclamation);
    }
}

```

程序错误在捕捉块中进行处理。

```

catch (Exception excep)
{
    MessageBox.Show (this, excep.Message, this.Text,
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
}
}

```

## 6.8 以 C++ 编写的 OPC 报警和事件自定义接口

### 简介

本部分介绍客户端要处理事件必须执行的基本步骤。

#### 步骤 1：初始化 COM 库

要使用 COM 库的每个程序必须首先初始化 COM 库。此操作可通过 *CoInitializeEx* 函数来完成，该函数比所需线程模型的 *CoInitialize* 多一个参数。这允许您确定是以单线程模式还是多线程模式创建 COM 对象。

```
HRESULT hr = CoInitializeEx(NULL, COINIT_MULTITHREADED);
```

#### 步骤 2：将 ProgID 转换为 CLSID

为了对其进行标识，每个 COM 服务器都有一个分配给全球唯一 *CLSID* 的 *ProgID*。使用 *CLSIDFromProgID()* 函数可实现上述操作。SIMATIC NET 的 OPC 报警和事件服务器的 ProgID 是 L"OPC.SimaticNETAlarms":

```
hr = CLSIDFromProgID(L"OPC.SimaticNETAlarms",
                      &clsidOPCEventServer);
```

#### 步骤 3：创建服务器对象

*CoCreateInstance()* 函数将创建已指定 *CLSID* 的类的实例。

```

hr = CoCreateInstance (clsidOPCEventServer,
                      NULL,
                      CLSCTX_LOCAL_SERVER,
                      IID_IOPCEventServer,
                      (void**)&gpIOPCEventServer);

```

此程序段的结果是 OPC 服务器类的对象。*CoCreateInstance* 还会提供指向服务器对象的 IOPCServer 接口的指针 (*gpIOPCEventServer* 参数)：

#### 步骤 4：在事件服务器注册

客户端必须在服务器注册以获取事件通知。OPCEventServer 类的 *CreateEventSubscription* 方法作为 *OPCEventSubscription* 类到 ES 服务器的对象，返回一个指向 *IOPCEventSubscriptionMgt* 接口的指针 (*pgIOPCEventSubscrMgt* 参数)。

```

hr = gpIOPCEventServer->CreateEventSubscription
    (TRUE,
     dwBufferTime,
     dwMaxSize,
     hClientSubscription,
     IID_IOPCEventSubscriptionMgt,
     (LPUNKNOWN*) &pgIOPCEventSubscrMgt,
     &dwRevisedBufferTime,
     &dwRevisedMaxSize );

```

#### 步骤 5：创建回调对象

为了允许发生事件时事件服务器向客户端发送通知，必须在客户端上执行 *IOPCEventCallback* 接口。示例程序中有一个用于此目的的 *COPCEventCallback* 类的对象，该类源于 *CObjectRoot* 和 *IOPCEventSink*。

```

CComObject<COPCEventCallback>::CreateInstance
    (&pgCOPCEventCallback);

```

*pgCOPCEventCallback* 参数是指向回调对象的指针。

## 步骤 6：连接 OPC 事件服务器与回调对象

*AtAdvise()* 方法将创建 OPC 服务器与回调对象之间的连接。

第一个参数是指向客户端要连接到的对象的 **IUnknown** 接口的指针。

第二个参数是指向回调对象的 **IUnknown** 接口的指针：

```
hr = AtlAdvise(pgiOPCEventSubscrMgt,
                pgCOPCEventCallback->GetUnknown(),
                IID_IOPCEventSink,
                &dwAdviseEvent);
```

## 步骤 7：接收来自报警和事件服务器的事件

如果发生报警，服务器将调用回调对象的 *OnEvent* 方法。在示例程序中已执行 *OnEvent*，这样 *CReceiveAnAlarm* 对话框类的 *displayEvent* 方法即显示事件的详细信息。

```
STDMETHODIMP COPCEventCallback::OnEvent(
    OPCHANDLE hClientSubscription,
    BOOL bRefresh,
    BOOL bLastRefresh,
    DWORD dwCount,
    ONEVENTSTRUCT __RPC_FAR *pEvents)
```

*pEvents* 参数是指向存储事件相关信息的结构的指针。

## 步骤 8：确认接收的事件

客户端使用 *AckCondition* 方法确认服务器通知。*pszSource*

参数包含事件的源，*pszConditionName* 表示必须确认状态更改的条件。

两个参数都是数组地址。在该示例中，仅确认了一个事件。然而，调用 *AckCondition* 可以确认多个事件。

报警和事件服务器调用 *OnEvent* 作为 **ONEVENTSTRUCT** 结构的 *szSource* 及 *szConditionName* 组件时，其提供 *pszSource* 和 *pszConditionName* 的值。

```
hr = gpIOPCEventServer->AckCondition(1,
                                         L"Me",
                                         L"NoComment",
                                         &pszSource,
                                         &pszConditionName,
                                         &ftActiveTime,
```

```
&dwCookie,
&pErrors);
```

### 步骤 9：删除对象并释放内存

退出程序之前，必须删除已创建的 OPC 对象并释放为其保留的内存。*AtlUnadvise* 将终止 OPC 服务器与回调对象之间的连接。

```
hr = AtlUnadvise(pgIOPCEventSubscrMgt,
                  IID_IOPCEventSink,
                  dwAdviseEvent);
```

每个 COM 接口都具有能够递减接口的参考计数器的 **Release** 方法。不再参考接口时，将释放所占用的资源。

```
hr=pgIOPCEventSubscrMgt->Release();
```

*CoUninitialize* 方法可关闭相关线程的 COM 库并释放其所使用的资源。对于每个成功的 *CoInitializeEx* 调用，必须有一个相应的 *CoUninitialize* 调用。

```
CoUninitialize();
```

## 6.9 以 C 编写的 OPC UA 接口

这些示例使用以简单的 C 语言编写的 OPC UA 接口。

您将在以下位置找到它们：

- “<安装路径>\SIEMENS\SIMATIC.NET\opc2\samples\ua\c\read.c”
- “<安装路径>\SIEMENS\SIMATIC.NET\opc2\samples\ua\c\alarm.c”
- “<安装路径>\SIEMENS\SIMATIC.NET\opc2\samples\ua\c\publish.c”

“Read.exe”程序显示“同步读取”功能。仅使用未加密通信。

“Alarm.exe”程序显示“异步监控数据变量和 S7 报警”功能。使用的是加密通信。

“Publish.exe”程序显示“读取、写入及异步监控数据变量和 S7 报警”功能。未加密和加密通信均可使用。可能还会进行用户验证。

后文的说明参考了较完整的示例“Publish”，该示例还包含更为简单的示例“Read”和“Alarm”的功能。

### 6.9.1 激活仿真连接

要使程序正常运行，需要激活使程序所用仿真变量可用的仿真连接。

请按下列步骤操作：

1. 从“开始”(Start) 菜单中启动“通信设置”(Communication Setting) 程序。

“开始 > Siemens 自动化 > SIMATIC > SIMATIC NET > 通信设置”(Start > Siemens Automation > SIMATIC > SIMATIC NET > Communication Settings)

2. 在左侧导航窗口中，通过逐级打开“SIMATIC NET 组态 > OPC 协议选择”(SIMATIC NET Configuration > OPC Protocol Selection) 打开 OPC 协议选择的属性页。
3. 启用待仿真协议的复选框。

本例使用 S7 协议。

因此要为 S7 协议激活演示连接。

4. 关闭“通信设置”(Communication Settings) 程序。

---

#### 说明

为使更改生效，必须先关闭所有 OPC 客户端，然后重新启动 OPC 服务器！

---

### 6.9.2 导入客户端证书

为了能够与服务器建立安全连接，服务器必须在交换证书期间接受客户端传来的证书。

为确保这一点，证书导入服务器的证书存储库。

按以下步骤导入客户端证书：

1. 从开始菜单中启动“通信设置”(Communication Settings) 组态程序。

“开始 > SIMATIC > SIMATIC NET > 设置 > 通信设置”(Start > SIMATIC > SIMATIC NET > Settings > Communication Settings)

2. 在左侧导航窗口中，通过逐级打开“SIMATIC NET 组态 > 应用程序 > OPC 设置”(SIMATIC NET Configuration > Applications > OPC Settings) 并单击“OPC UA 证书”(OPC UA Certificates) 图标，打开证书的属性页。
3. 在快捷菜单中，选择菜单命令“导入客户端证书...”。  
其位于硬盘的  
<安装路径>\SIEMENS\SIMATIC.NET\opc2\samples\ua\c\pki\ca\certs\Opc.Publish.C. Sample.der 中，然后单击“打开”(Open)。
4. 关闭“通信设置”(Communication Settings) 程序。

### 6.9.3 使用示例程序

该程序包含两个菜单。第一个是主菜单，会在您启动程序时出现。

如果与服务器之间建立了连接，则会出现连接菜单。下列操作均由菜单触发：

菜单命令	操作
<b>主菜单</b>	
c (连接)	与服务器之间建立了一个不安全的连接。
s (安全连接)	与服务器之间建立了一个安全的连接。
q (退出)	退出程序。
<b>连接菜单</b>	
r (读取)	读取变量的值。
w (写入)	写入变量的值。
m (监控)	监控变量的值更改和报警。
d (断开)	终止与服务器的连接。
q (退出)	终止与服务器的连接并退出程序。

### 6.9.4 启动程序

该程序位于硬盘中的以下位置：

“<安装路径>\SIEMENS\SIMATIC.NET\opc2\samples\ua\c\publish.c”

程序启动后，主菜单出现。通过按键盘上的“c”键，可建立与服务器之间的不安全连接。

相反，如果您按下“s”键，则会在客户端与服务器之间交换证书，此时两端都必须确认对方的证书。之后，将会与服务器之间建立安全连接。

在连接建立后，将显示连接菜单以提供用于读取、写入和监控变量的选项。

### 6.9.5 读取和写入值

通过在键盘上按“r”键选择“读取”操作后，会在控制台上读取并显示变量值。

然后，连接菜单将出现。

如果按键盘上的“w”键，接着输入一个介于 0 和 255

之间的整数，则会将该数值写入变量。连接菜单会在操作完成时再次显示。

按照实际情况来说，示例程序专用于通过演示连接进行操作。

如果要在真实环境下运行示例程序，需要调整实际变量的代码（请参见“有关转换为实际变量的注意事项（页 856）”）。

### 6.9.6

### 监控变量

如果按键盘上的“m”键，则会在控制台连续输出值更改和变量报警。

通过按“q”键将监控停止，并且连接菜单会再次出现。

按照实际情况来说，示例程序专用于通过演示连接进行操作。

要在控制台上查看值更改，则其需要手动或使用生成模式进行生成。

请按下列步骤操作：

1. 从开始菜单中启动 OPC Scout V10 程序。

“开始 > Siemens Automation > SIMATIC > SIMATIC NET > OPC Scout V10”(Start > Siemens Automation > SIMATIC > SIMATIC NET > OPC Scout V10)

2. 在左侧的导航窗口中，逐级打开

“UA 服务器 > opc.tcp://<hostname>:55101 > 对象 > 服务器 >  
S7: > DEMO > blocks > db > db20)

3. 将变量“db20.0,b”拖动到 DA 视图 1 中。

4. 在“新值”(New value) 框中输入新值以手动更改值，然后单击“写入”(Write) 按钮。

单击“读取或监控打开”(Reading or Monitoring ON)  
按钮时，当前变量值会出现在“值”(Value) 框中。

5. 在“生成模式”(Generation Mode) 框中输入所需的生成来生成用于设置变量的值（例如  
[0...255]+1），以将值从 0 到 255 连续增加。

6. 使用“生成值开启”(Generate Values ON) 按钮激活值的生成。

---

### 说明

如果要在真实环境下运行示例程序，需要调整实际变量的代码（请参见“有关转换为实际变量的注意事项（页 856）”）。

---

### 6.9.7

### 停止程序

通过按主菜单或连接菜单中的“q”键，可关闭示例程序。

如果该程序位于连接菜单中，则会首先终止与服务器的连接。

## 6.9.8 程序顺序的说明

### 6.9.8.1 连接建立

与服务器建立连接的方法有两种。

第一种是通过建立安全连接及交换证书和安全端点进行建立，第二种是通过不安全端点进行建立。

#### 安全连接建立的步骤

##### 1. 创建通道。

要建立与服务器之间的连接，必须先创建通道。为此，可使用“`OpcUa_Channel_Create()`”函数。

##### 2. 在安全模式为“无”的情况下打开一个端点的通道。

要建立与端点的连接，可使用函数“`OpcUa_Channel_Connect()`”。这将传送端点的安全要求，在这种情况下，安全模式为“无”。这将指定服务器不需要任何安全功能。因此，该连接不安全。

##### 3. 读出服务器的端点。

“`OpcUa_ClientApi_GetEndpoints()`”函数将服务器的端点告知客户端。

在安全模式为“Sign&Encrypt”的情况下，服务器证书也会被传送到通过端点建立连接时所需的客户端。

##### 4. 接受服务器证书。

服务器证书必须被客户端接受。

之后，“`ValidateCertificate()`”函数会检查证书是否已经位于客户端的证书存储库中。

如果不是这种情况，则将使用“`SaveCertificate()`”函数将其复制到客户端存储库中。

##### 5. 关闭不安全通道。

要关闭与端点之间的连接，应使用“`OpcUa_ClientApi_Disconnect()`”函数。

##### 6. 打开安全通道。

如第 2 步中所示，再次使用“`OpcUa_Channel_Connect()`”函数来建立与端点的连接。

此处所使用的端点具有安全模式“Sign&Encrypt”，这意味着客户端与服务器之间的消息需要进行加密和签名。

##### 7. 创建会话。

要创建一个会话，请使用“`OpcUa_ClientApi_CreateSession()`”函数。服务器的回复由一个唯一的数字组成（称为“`ServerNonce`”），其在步骤 8 中激活会话时必不可少。

##### 8. 激活会话。

要激活在步骤 7

中所创建的会话，请使用“`OpcUa_ClientApi_ActivateSession()`”函数。它将传送一个由在步骤 7 中接收的“`ServerNonce`”和服务器证书所组成的签名。

## 不安全连接建立的步骤

### 1. 创建通道。

要建立与服务器之间的连接，必须先创建通道。为此，可使用“`OpcUa_Channel_Create()`”函数。

### 2. 在安全模式为“无”的情况下打开一个端点的通道。

要建立与端点的连接，可使用函数“`OpcUa_Channel_Connect()`”。这将传送端点的安全要求，在这种情况下，安全模式为“无”。这将指定服务器不需要任何安全功能。因此，该连接不安全。

### 3. 创建会话。

要创建一个会话，请使用“`OpcUa_ClientApi_CreateSession()`”函数。服务器的回复由一个唯一的数字组成（称为“`ServerNonce`”），其在步骤 8 中激活会话时必不可少。

### 4. 激活会话。

要激活在步骤 7

中所创建的会话，请使用“`OpcUa_ClientApi_ActivateSession()`”函数。它将传送一个由在步骤 7 中接收的“`ServerNonce`”和服务器证书所组成的签名。

## 6.9.8.2 读取和写入变量

### 读取变量的值

要读取一个变量的值，请使用“`OpcUa_ClientApi_Read()`”函数。

其不仅会返回值，而且会返回一个用于指定函数能否成功完成的状态代码。

### 将值写入变量

要将值写入变量，请使用“`OpcUa_ClientApi_Write()`”函数。

输出是一个指定函数是否成功的状态代码。

### 6.9.8.3 监控变量和报警

请按以下步骤监控变量和报警：

1. 创建订阅。

要创建一个订阅，请使用“`OpcUa_CreateSubscription()`”函数。

作为传送值，该函数还会传送发布间隔。

2. 创建用于监控值和报警的已监控数据项。

用于监控数据更改和报警的已监控数据项是用“`OpcUa_ClientApi_CreateMonitoredItems()`”函数创建的。对于用于监控报警的已监控数据项，需要设置一个事件过滤器。

3. 向服务器发送一个发布请求。

要接收来自服务器的报警或数据更改，客户端需要调用“`OpcUa_ClientApi_BeginPublish()`”函数。这将传递为服务器调用以发送其响应的回调函数。

4. 评估发布请求的响应。

“`pfnUaServer_ClientChannelRequestComplete()`”回调函数会接收服务器对于发布请求做出的响应。在此，所包含的报警会被存储，并且相关的屏幕显示也会生成。

之后，系统会向服务器发送进一步的发布请求。

5. 删除监控数据项。

监控数据项可使用“`OpcUa_ClientApi_DeleteMonitoredItems()`”函数进行删除。

6. 删除订阅。

“`OpcUa_ClientApi_DeleteSubscription()`”函数可删除订阅。

在此，如果仅在已删除订阅后才会出现未决的发布响应，则可调用回调函数并显示错误。

### 6.9.8.4 终止连接

请按照以下步骤终止连接：

1. 关闭会话。

会话可使用“`OpcUa_ClientApi_CloseSession()`”函数进行关闭。

2. 关闭通道。

“`OpcUa_Channel_Close()`”函数可关闭通道。

3. 删除通道。

通道可使用“`OpcUa_Channel_Delete()`”函数进行删除。

### 6.9.9 有关转换为实际变量的注意事项

要将示例程序转换为实际变量，必须调整以下几行才能适合变量：

- g\_NodeId\_Variable.NamespaceIndex = NAMESPACE\_S7;
- g\_NodeId\_Variable.IdentifierType = OpcUa\_IdentifierType\_String;
- g\_NodeId\_Variable.Identifier.String.strContent = NODE\_IDENTIFIER\_STRING;
- g\_NodeId\_Variable.Identifier.String.uLength =  
OpcUa\_StrLenA(NODE\_IDENTIFIER\_STRING);

---

#### 说明

有关 **NodeId** 和服务器命名空间的更多详细信息，请参见“通过 OPC UA 进行的 S7 通信 (页 184)”部分。

---

## 6.10 C# 中的 OPC UA 接口（异步通信）

您可从我们的 Internet 页面中，下载有关编程 C# 中 OPC UA 接口的充分描述的示例：

用 C# 为 SIMATIC NET OPC UA 服务器编程一个 OPC UA .NET 客户端  
(<http://support.automation.siemens.com/WW/view/zh/42014088>)

在此，通过两个复杂性阶段实施 PC 站中的 OPC UA 客户端。

一个非常简单的客户端（简单 OPC UA

客户端）将为您显示所有基本功能来帮助您快速掌握 OPC UA。

一个带有方便的用户界面的更复杂的客户端（OPC UA .NET 客户端）演示了在 .NET 下以 C# 编程语言通过可重复使用类别所实施的 OPC UA 的专业用法。

在两个示例客户端中依据程序对下列情况进行了解释：

- 使用 OPC UA 服务器进行登录、注销和身份验证
- 浏览变量的命名空间
- 读取、写入和监控变量
- 使用 S7 块服务时进行读取和写入
- 使用绝对和符号寻址
- 简单错误处理

# 参考自动化接口

## 简介

本部分包含了 OPC 自动化接口的规范。

它描述了属性、方法和各对象的事件（适用时）。

## 7.1 常规信息

OPC 自动化接口简介解释了规范中常用的最重要术语。 必须了解自动化接口。

以下术语将被详细解释：

- 接口
- 接口类型
- COM/OLE 对象
- 集合对象
- 对象模型
- 数据同步
- 例外
- 事件
- 数组
- 参数
- 类型库

该说明与 OPC 自动化接口规范的版本 2.02 相关。

### 7.1.1 什么是接口？

对象可具有多个接口并会通过接口进行充分描述。

此方法（对象功能）只能通过接口进行执行，并且只能通过接口来访问对象。

对象可以通过数个应用程序进行使用。 接口由一个指向实际函数的指针的表格组成。

#### 接口的结构

在 OLE 2.0 中，一个指向接口的指针是一个指向带有函数指针的跳转表的指针。

调用程序使用了一个指向所需接口的指针。

而其中包含了一个指向函数指针列表的指针。 现在，它们参考 COM/OLE 对象中的实际方法。

### 7.1.2 OPC 的两种接口类型

#### COM/OLE 接口

COM/OLE 组件为其它组件或应用程序提供对象及其方法。 这些对象将通过 COM/OLE 接口进行访问。 COM/OLE 意义上的接口是一组逻辑相关函数。

SIMATIC NET 的 OPC 服务器支持两种不同类型的 COM/OLE 接口：

- 自动化和
- 自定义接口。

两个接口都用于对象之间的通信。

自动化接口与自定义接口在以下方面彼此不同：在内部调用接口的方法。

这意味着，OPC 服务器有两种不同的接口规范。

#### 何时使用何种接口？

基于脚本语言（例如 Visual Basic 或 VBA）的客户端应用程序必须使用自动化接口。

使用 C/C++ 编程语言的应用程序应使用自定义接口来达到最佳性能。 然而，在 C/C++ 中也可以使用自动化接口。

### 7.1.3 COM/OLE 对象

COM/OLE 对象是 Windows 中的单元，它们通过其界面为其它对象提供已定义的功能。

COM/OLE 对象通过已定义的接口提供服务。

对象的内容、数据和代码对于对象用户仍然不可见。 COM/OLE

对象通过其接口进行定义。 OLE

意义上的术语对象与面向对象的编程语言中的对象定义不同。例如，COM/OLE 对象不支持继承功能。

#### COM/OLE 对象的结构

对象仅通过接口之一进行访问。

无法将实际对象、其包含的数据或代码作为一个整体进行访问。

接口隐藏了已为其分配的方法。

### 7.1.4 集合对象

集合对象用于生成和管理子单元。仅当已经存在集合对象时，才能创建相应的子单元。

集合对象为相关子单元指定默认值。

COM/OLE 自动化集合是支持 *Item* 方法、*Count* 属性和隐藏属性 *\_NewEnum* 的对象。

每一个具有这些属性并作为接口的一部分的对象都是集合对象。

在 VB 中，有两种方法可以依次运行这些集合的元素。

第一种方法将显式使用 *Count* 和 *Item* 来标识集合元素。

对于 I = 1 到 object.Count

元素 = object.Item ( I )

' 或...

元素 = object( I )

下一个 I

在第二个选项中，可用数据项将通过使用隐藏功能 *\_NewEnum* 依次运行：

对于对象中的每一个元素

...

下一个元素

使用 *For Each* 的方法依次运行一个集合的速度比使用 *Item* 属性时要快。

## 7.1 常规信息

使用 *Item*, 也可访问特定目录, 如 *Item(3)*。因此, 该属性的使用不会被限制为循环。

### 集合对象的两种类型

在 OPC 中, 有两种不同的集合对象: **OPCGroups** 集合对象可用于创建和/或编辑 **OPCGroup** 类的对象。 **OPCGroups** 集合对象具有七种属性、七种方法和一个事件。

**OPCItems** 集合对象可用于创建和/或编辑 **OPCItem** 类的对象。 **OPCItems** 集合对象具有五种属性和九种方法, 只是没有事件。

## 7.1.5 对象模型

### OPCServer 类的对象

**OPCServer** 类的对象由客户端创建。OPC 服务器的属性包含有关服务器的常规信息。在创建一个 **OPCServer** 对象时, 也会将一个 **OPCGroup** 集合创建为 **OPCServer** 对象的属性。

### OPCGroups 类的对象

**OPCGroups** 对象就是一个用于创建和管理 **OPCGroup** 对象的集合对象。 **OPCGroups** 的默认属性指定了创建所有 **OPCGroup** 对象的默认值。

---

#### 说明

SIMATIC NET 的 OPC 服务器不支持公共组。

---

### OPCGroup 类的对象

**OPCGroup** 类可管理各个过程变量, 即 OPC 数据项。利用 **OPCGroup** 对象, 客户端可以形成有意义的 **OPCItem** 对象单元并使用它们执行操作。

### OPCItems 类的对象

**OPCItems** 对象是一个创建和管理 **OPCItem** 类的对象的集合对象。 **OPCItems** 的默认属性指定了要创建的所有 **OPCItem** 对象的默认值。

## OPCItem 类的对象

OPCItem 类的对象表示与过程变量的连接，例如与可编程控制器输入模块的连接。

过程变量是过程 I/O 的可写入和/或可读取数据项，如罐温度。

每个过程变量均与某个值（数据类型 VARIANT）、质量和时间戳相关联。

下图说明了对象模型所基于的层级：

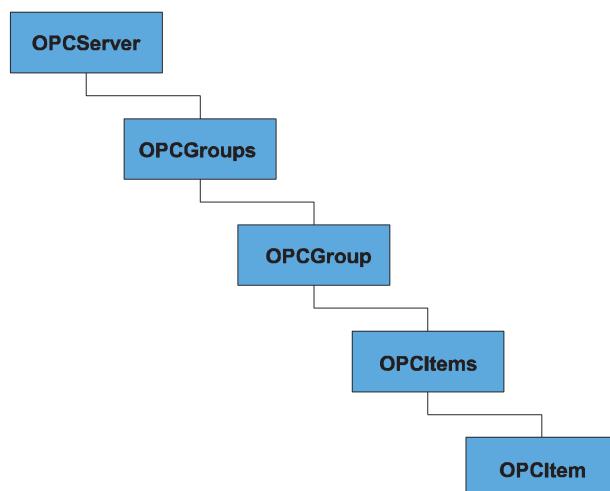


图 7-1 对象模型中的基础层级

## 7.1.6 数据同步

VB 客户端必须能够读取和接收数据以便值、质量和时间戳保持同步。

客户端必须确定数据质量和时间戳与值相匹配。

如果客户端通过读取方法获得值，则值、质量和时间戳同步。

如果客户端使用 *DataChange*

事件获取数据，则值、时间戳和质量在事件处理例程范围内同步。

如果这两种获取数据的方法无法保持严格独立，则客户端将无法辨别数据项属性是否完全同步；例如，如果当客户端正在访问属性时数据属性发生更改，则可能会出现这种情况。

## 7.1.7 例外

此处描述的大多数属性和事件与 OPC 自定义服务器进行通信。当访问属性时，难以利用 OLE 自动化返回错误。当相应的数据源中出现错误时，自动化服务器会生成一个例外。

这意味着，户端必须具有例外处理例程以便能够处理错误。

在通过 Visual Basic 的 *Err* 对象指示属性被写入时发生错误。

### 7.1.8 事件

自动化接口支持 Visual Basic 5.0 事件的通知机制。

自动化服务器根据方法调用 *AsyncRefresh*、*AsyncRead* 和 *AsyncWrite* 的要求触发事件，并且还会在数据发生更改时，根据客户端的规范触发事件。

该执行过程假定客户端具有适合的事件处理例程。

### 7.1.9 数组

数组编号从 1 开始。如果在函数中使用一个比计数或 **NumItems** 参数长的数组，则将仅使用由参数指定的数组元素数（索引从 1 开始）。

这仅与自动化接口内的函数和事件的参数相关；它不会影响数据类型本身是数组的数据项值。

为了避免错误，应使用 VB 代码 *Option Base 1*。

### 7.1.10 参数

#### 可选参数

可选参数由 *Optional* 指示。如果默认行为适当，则不必将可选参数包括在方法调用中。在 OLE 自动化中，即使可选参数包含字符串或数组等，也必须作为 *Variant* 进行声明。

#### 方法参数

方法参数将作为值进行传递，除非它们被指定为具有 *ByRef* 补充的参考。*ByRef* 参数将通过方法进行赋值和返回。

### 7.1.11 类型库

在 VB 中，“OPC 自动化类型库”用于定义下列接口。确保“OPC Automation 2.0”已激活。

## 7.2 OPCServer 对象

### 说明

客户端生成 OPCServer 自动化对象，然后通过 OPC 数据访问自定义接口与其连接（请参见 *Connect* 方法）。利用 OPCServer 对象，可以调用有关 OPC 服务器的常规信息，并可创建和处理 OPCGroups 集合对象。

### 语法

```
OPCServer
```

### 备注

利用 *WithEvents* 进行声明的影响是对象可以支持由该对象类型所指定的事件。OPC 服务器对象仅有一个事件，即 *ServerShutdown* 事件。OPCGroup 对象拥有与 *DataChange* 和对 OPCGroup 对象的异步方法的支持相关的所有事件。

### 示例

```
Dim WithEvents AnOPCServer As OPCServer  
Set AnOPCServer = New OPCServer
```

### 7.2.1 OPCServer 对象的属性

StartTime  
CurrentTime  
LastUpdateTime  
MajorVersion  
MinorVersion  
BuildNumber  
VendorInfo  
ServerState  
LocaleID  
Bandwidth  
OPCGroups  
PublicGroupNames  
ServerName  
ServerNode  
ClientName

#### 7.2.1.1 StartTime

##### 说明

(只读) 此属性指示由客户端选定的服务器的开始时间。已连接到相同 OPC 服务器的多个客户端都读取该属性的相同时间。

##### 语法

```
StartTime As Date
```

##### 备注

自动化服务器使用自定义接口 *GetStatus ()* 来获得 OPCServer 对象的该属性以及许多其它属性的值。当客户端没有通过 *Connect* 方法与 OPC 服务器连接时，会发生错误。

##### 示例

```
Dim AnOPCServerTime As Date  
AnOPCServerTime = AnOPCServer.StartTime
```

### 7.2.1.2 CurrentTime

#### 说明

(只读) 该属性会从服务器返回当前时间。如果您访问 *CurrentTime* 属性，可以获得自动化服务器通过 **GetStatus** 接口接收来自自定义服务器的值。

#### 语法

```
currentTime As Date
```

#### 备注

当客户端没有通过 *Connect* 方法与 OPC 服务器连接时，会发生错误。

#### 示例

```
Dim AnOPCServerTime As Date  
AnOPCServerTime = AnOPCServer.CurrentTime
```

### 7.2.1.3 LastUpdateTime

#### 说明

(只读) 该属性指示服务器发送最后一次数据更新的时间。如果您访问 *LastUpdateTime* 属性，则可获得自动化服务器通过 **GetStatus** 接口接收来自自定义服务器的值。

#### 语法

```
LastUpdateTime As Date
```

#### 备注

该属性指示服务器最后一次向应用程序客户端发送数据的时间。当客户端没有通过 *Connect* 方法与 OPC 服务器连接时，会发生错误。

#### 示例

```
Dim AnOPCServerTime As Date  
AnOPCServerTime = AnOPCServer.LastUpdateTime
```

### 7.2.1.4 MajorVersion

#### 说明

(只读) 该属性指示服务器的主要版本号 (例如 1 表示版本 1.32)。如果您访问 *MajorVersion* 属性，可以获得自动化服务器通过 **GetStatus** 接口接收来自自定义服务器的值。

#### 语法

```
MajorVersion As Integer
```

#### 备注

当客户端没有通过 **Connect** 方法与 OPC 服务器连接时，会发生错误。

#### 示例

```
Dim AnOPCServerMajorVersion As String  
AnOPCServerMajorVersion = Str(AnOPCServer.MajorVersion)
```

### 7.2.1.5 MinorVersion

#### 说明

(只读) 该属性指示服务器的次要版本号 (例如 32 表示版本 1.32)。如果您访问 *MinorVersion* 属性，可以获得自动化服务器通过 **GetStatus** 接口接收来自自定义服务器的值。

#### 语法

```
MinorVersion As Integer
```

#### 备注

当客户端没有通过 **Connect** 方法与 OPC 服务器连接时，会发生错误。

## 示例

```
Dim AnOPCServerMinorVersion As String  
AnOPCServerMinorVersion = Str(AnOPCServer.MinorVersion)
```

### 7.2.1.6 BuildNumber

#### 说明

(只读) 该属性会指定服务器的版本号。如果您访问 *BuildNumber* 属性，可以获得自动化服务器通过 **GetStatus** 接口接收来自自定义服务器的值。

#### 语法

```
BuildNumber As Integer
```

#### 备注

当客户端没有通过 *Connect* 方法与 OPC 服务器连接时，会发生错误。

#### 示例

```
Dim BuildNumber as Integer  
BuildNumber = AnOPCServer.BuildNumber
```

### 7.2.1.7 VendorInfo

#### 说明

(只读) 该属性提供了有关服务器供应商的信息（以字符串形式）。如果您访问 *VendorInfo* 属性，可以获得自动化服务器通过 **GetStatus** 接口接收来自自定义服务器的值。

#### 语法

```
VendorInfo As String
```

#### 备注

当客户端没有通过 *Connect* 方法与 OPC 服务器连接时，会发生错误。

## 示例

```
Dim info As String  
info = AnOPCServer.VendorInfo
```

### 7.2.1.8 ServerState

#### 说明

(只读) 该属性指示作为 OPCServerState 类型常量的服务器的状态。

#### 语法

```
ServerState As Long
```

#### 设置

##### OPC\_STATUS\_RUNNING

服务器正在正常运行。这是服务器的正常状态。

##### OPC\_STATUS\_FAILED

服务器上发生了供应商特定的错误，并且服务器无法正确运行。

消除错误的过程是供应商特定的。服务器的每一个其它方法都会输出错误代码 E\_FAIL。

##### OPC\_STATUS\_NOCONFIG

尽管服务器正在运行，但其不包含有关组态的信息，因此无法正确运行。

注意：该服务器需要有关组态的信息才能运行！

不需要组态信息的服务器不会显示此状态。

##### OPC\_STATUS\_SUSPENDED

有时，服务器没有通过供应商特定的方法所进行的连接，也不会发送或接收数据。

有关质量的信息如下所示：OPC\_QUALITY\_OUT\_OF\_SERVICE。

##### OPC\_STATUS\_TEST

服务器处于测试模式。数据输出将与硬件断开连接，否则服务器表现正常。

根据特定供应商的执行情况，输入数据或者实际存在，或者被仿真。

正常输出质量信息。

## 备注

以上状态信息在数据访问自定义接口的规范中进行了说明，并由 OPC 服务器通过该自定义接口返回；更多详细信息，请参见 *OPC 数据访问自定义接口规范*中的 *IOPCServer::GetStatus()*。自动化服务器已采用来自自定义服务器的该信息，它们通过 *GetStatus()* 接口连接在一起。当客户端没有通过 *Connect* 方法与 OPC 服务器连接时，会发生错误。

## 示例

```
Dim ServerState As Long  
ServerState = AnOPCServer.ServerState
```

### 7.2.1.9 LocaleID

#### 说明

(读取和写入) 此属性将标识位置；使用 *LocaleID*，可以根据特定语言对服务器所返回的字符串进行调整。该 *LocaleID* 将由接口的 *GetErrorString* 方法使用。

#### 语法

```
LocaleID As Long
```

#### 备注

在受 *LocaleID* 影响的服务器的所有功能中，都应将 *LocaleID* 设为默认值。  
当客户端没有通过 *Connect* 方法与 OPC 服务器连接时，会发生错误。

## 示例

```
' Get the property:  
Dim LocaleID As Long  
LocaleID = AnOPCServer.LocaleID  
  
' Specify the property:  
AnOPCServer.LocaleID = LocaleID
```

### 7.2.1.10 Bandwidth

#### 说明

(只读) **Bandwidth** 指定服务器的带宽。该属性是服务器特定的属性。

最好以可用带宽的百分比形式指定服务器的带宽。如果服务器无法计算带宽，该值将为 hFFFFFF。如果您访问该属性，则可获得自动化服务器通过 **GetStatus** 接口接收来自自定义服务器的值。

#### 语法

```
Bandwidth As Long
```

#### 备注

当客户端没有通过 **Connect** 方法与 OPC 服务器连接时，会发生错误。

#### 示例

```
Dim Bandwidth As Long  
Bandwidth = AnOPCServer.Bandwidth
```

### 7.2.1.11 OPCGroups

#### 说明

(只读) **OPCGroups** 是一个 **OPCGroup** 类的对象的集合。它是 **OPCServer** 对象的默认属性。

#### 语法

```
OPCGroups As OPCGroups
```

## 示例

```
' To specify the property precisely:  
Dim groups As OPCGroups  
Set groups = AnOPCServer.OPCGroups  
  
' Default specification:  
Dim groups As OPCGroups  
Set groups = AnOPCServer
```

### 7.2.1.12 PublicGroupNames

#### 说明

(只读) 该属性指定了由服务器提供的 **PublicGroups** 的名称。这些名称可以在 **ConnectPublicGroup** 中使用。该名称将以字符串数组的形式进行输出。

#### 语法

```
PublicGroupNames As Variant
```

#### 备注

当客户端没有通过 **Connect** 方法与 OPC 服务器连接时，会发生错误。如果 OPC 服务器不支持任何公共组或没有定义任何公共组，则将输出一个空列表。

#### 示例

```
Dim AllPublicGroupNames As Variant  
AllPublicGroupNames = AnOPCServer.PublicGroupNames
```

### 7.2.1.13 ServerName

#### 说明

(只读) 该属性会输出客户端通过 **Connect** 方法连接到的服务器的名称。

## 语法

ServerName As String

## 备注

自动化服务器返回在缓存中以本地方式找到的值。

如果未输出任何值，则表示客户端未连接到数据访问服务器。

## 示例

```
Dim info As String  
info = AnOPCServer.ServerName
```

### 7.2.1.14 ServerNode

## 说明

(只读) 该属性会输出 OPC 服务器所运行的网络中的节点的计算机名称。

如果您访问该属性，则可以获得自动化服务器在缓存中以本地方式找到的值。

## 语法

ServerNode As String

## 备注

如果客户端未连接到数据访问服务器，则不会输出任何值。如果在 *Connect* 方法中未指定任何主机名称，则字符串也会为空。

## 示例

```
Dim info As String  
info = AnOPCServer.ServerNode
```

### 7.2.1.15 ClientName

#### 说明

(读取和写入) 在此, 客户端可以在服务器上注册 ClientName。

该过程主要用于故障排除。客户端最好在此处指定其节点和 EXE 名称。

#### 语法

```
ClientName As String
```

#### 备注

最好以由分号 (;) 分隔的字符串形式输入节点和客户端名称。

#### 示例

```
' To get the property:  
Dim info As String  
info = AnOPCServer.ClientName  
  
' To specify the property:  
AnOPCServer.ClientName = "NodeName;c:\programfiles\vendor\someapplication.exe"
```

### 7.2.2 OPCServer 对象的方法

- GetOPCServers
- Connect
- Disconnect
- CreateBrowser
- GetErrorString
- QueryAvailableLocaleIDs
- QueryAvailableProperties
- GetItemProperties
- LookupItemIDs

### 7.2.2.1 GetOPCServers

#### 说明

该属性会输出已注册的 OPC 服务器的名称 (ProgID)。在 *Connect* 方法中使用其中的一个 ProgID。该名称将以字符串数组的形式进行输出。

#### 语法

```
GetOPCServers (Optional Node As Variant) As Variant Node
```

#### 节点

利用节点名称，可以指定自动化服务器将输出所有已注册的 OPC 服务器的网络节点。

#### 备注

有关注册自定义服务器的信息，请参见 OPC 数据访问自定义接口的标准。  
节点的用法是可选的。如果使用一个节点名称，则可通过 DCOM 访问另一台计算机。  
允许的节点名称是 UNC 名称和 DNS 名称 (server.com、www.vendor.com 或  
180.151.19.75)。

#### 示例

```
' Get the registered OPC servers (that actually
' exist that will be added to a VB
' list box):
Dim AllOPCServers As Variant
AllOPCServers = AnOPCServer.GetOPCServers
For i = LBound(AllOPCServers) To UBound(AllOPCServers)
    listbox.AddItem AllOPCServers(i)
Next i
```

### 7.2.2.2 Connect

#### 说明

如果您需要任何与 OPC 数据访问服务器的连接（可提供自定义接口），则必须调用此方法。

## 语法

```
Connect (ProgID As String,  
          Optional Node As Variant)
```

### ProgID:

*ProgID* 是一个唯一标识已注册的 OPC 数据访问服务器的字符串（可提供自定义接口）。

### Node:

利用节点名称，可以指定通过 DCOM 建立与另一台计算机的连接。

## 备注

每一个 OPC 自动化服务器实例都会连接到一个 OPC 数据访问服务器（提供自定义接口）。

节点的用法是可选的。如果使用一个节点名称，则可通过 DCOM 访问另一台计算机。允许的节点名称是 UNC 名称和 DNS 名称（server.com、www.vendor.com 或 180.151.19.75）。

使用该方法，自动化包装可调用 *CoCreateInstanceEx*，并在指定的计算机 (*StrNodeName*) 中设置数据访问自定义服务器（由 *ProgID* 指定）。

如果在第二次调用该方法时没有通过调用 *Disconnect* 方法预先终止连接，则自动化包装会自动取消先前的连接。

使用 *GetOPCServers* 方法获得有效 *ProgID*。

## 示例

```

' Establish a connection to the first registered
' OPC Server found by the "GetOPCServers"
' method:

Dim AllOPCServers As Variant
AllOPCServers = AnOPCServer.GetOPCServers
AnOPCServer.Connect(AllOPCServers(1))

' Connection to a specific server or a
' network computer:

Dim ARealOPCServer As String
Dim ARealOPCNodeName As String
ARealOPCServer = "Vendorname.DataAccessCustomServer"
ARealOPCNodeName = "AComputername"
AnOPCServer.Connect (ARealOPCServer, ARealOPCNodeName)

```

### 7.2.2.3 Disconnect

#### 说明

该方法可终止与 OPC 服务器的链接。

#### 语法

```
Disconnect()
```

#### 备注

可以利用此方法终止与服务器的连接；然后，可以与另一台服务器建立新连接，也可删除服务器对象。

如果客户端应用程序使用相应的自动化方法来终止它所建立的全部对象（其中包括全部 **OPCGroup** 和 **OPCItem** 对象），则属于较好的编程方式。*Disconnect* 方法会移除所有组对象及相关 OPC 自定义服务器的参考。

#### 示例

```
AnOPCServer.Disconnect
```

### 7.2.2.4 CreateBrowser

#### 说明

此方法生成了一个 CreateBrowser 类的对象。

#### 语法

```
CreateBrowser() As OPCBrowser
```

#### 备注

OPC 浏览器接口可选，且不一定需要 OPC 自定义接口服务器支持。这意味着，不执行浏览器接口的 OPC 自定义接口服务器也不会返回 OPC 浏览器类的对象。

#### 示例

```
Dim ARealOPCServer As String  
Dim ARealOPCNodeName As String  
ARealOPCServer = "Vendorname.DataAccessCustomServer"  
ARealOPCNodeName = "AComputername"  
AnOPCServer.Connect(ARealOPCServer, ARealOPCNodeName)  
Dim AnOPCServerBrowserObject As OPCBrowser  
Set AnOPCServerBrowserObject = AnOPCServer.CreateBrowser
```

### 7.2.2.5 GetErrorString

#### 说明

该方法可将错误消息转换为可读的字符串。服务器会在其 **LocaleID** 属性中的指定位置输出字符串。

#### 语法

```
GetErrorString (ErrorCode As Long) As String
```

#### ErrorCode

*ErrorCode*

是客户端应用程序已接收回来的来自服务器接口函数的服务器特定的错误代码。对于此错误代码，客户端可查询文本表示。

## 示例

```
Dim AnOPCServerErrorMessage As String
' this example assumes that while adding
' several items, an item is detected as being invalid; the code
' is not given completely here to simplify matters
AnOPCItemCollection.Add AddItemCount,
AnOPCItemIDs,
AnOPCItemServerHandles,
AnOPCItemErrors
' Get and display the error message to inform the user
' why the item could not
' be added
AnOPCServerErrorMessage =
    AnOPCServer.GetErrorString(AnOPCItemErrors (index))
' further code
ErrorBox.Text = AnOPCServerErrorMessage
' further code
```

### 7.2.2.6 QueryAvailableLocaleIDs

#### 说明

该方法将现有服务器-客户端连接的可用 **LocaleID** 输出为长值数组。

#### 语法

```
QueryAvailableLocaleIDs () As Variant
```

## 示例

```

Dim LocaleID As Variant
Dim AnOPCTextString as String
AnOPCServerLocaleID = AnOPCServer.QueryAvailableLocaleIDs()
For i = LBound(LocaleID) To UBound(LocaleID)
    AnOPCTextString = LocaleIDToString(LocaleID(i))
    listbox.AddItem AnOPCTextString
Next i

```

### 7.2.2.7 QueryAvailableProperties

#### 说明

该方法会输出一个由相关 **ItemID** 的可用属性的说明和 ID 代码组成的列表。该列表可根据 **ItemID** 有所不同。它对于一个特定的 **ItemID** 应相对稳定。这意味着，它可能会受系统组态更改影响。

#### 语法

```

QueryAvailableProperties (ItemID As String,
                           ByRef Count As Long,
                           ByRef PropertyIDs () as Long,
                           ByRef Descriptions() As String,
                           ByRef DataTypes() As Integer)

```

##### **ItemID**

*ItemID* 是将在可用属性中查询的 **ItemID**。

##### **Count**

*Count* 指示所找到的属性数。

##### **PropertyIDs**

*PropertyIDs* 是所找到属性的 DWORD 类型的 ID。这些 ID 可以被传递到 *GetItemProperties* 或 *LookupItemIDs*。

##### **Descriptions**

*Descriptions* 是对由供应商提供的每个属性的简要说明。

注意： **LocaleID** 对此说明的语言没有任何影响。

##### **DataTypes**

*DataTypes* 是 *GetItemProperties* 为该属性输出的数据类型。

## 示例

```

' Get the available properties:
Dim OPCItemID As String
Dim ItemCount As Long
Dim PropertyIDs() As Long
Dim Descriptions() As String
Dim DataTypes() As Integer
Dim AnOPCTextString As String
OPCItemID = "SomeOPCDataAccessItem"
AnOPCServer.QueryAvailableProperties (OPCItemID, ItemCount,
                                      PropertyIDs,
                                      Descriptions,
                                      DataTypes)
For i = 1 To ItemCount
    AnOPCTextString = Str(PropertyIDs(i)) + " "
    + Descriptions(i)
    listbox.AddItem AnOPCTextString
Next I

```

### 7.2.2.8 GetItemProperties

#### 说明

该方法会输出一个所输入的 **ItemID** 属性的当前值列表。

#### 语法

```

QueryAvailableProperties (ItemID As String,
                           ByRef Count As Long,
                           ByRef PropertyIDs () as Long,
                           ByRef Descriptions() As String,
                           ByRef DataTypes() As Integer)

```

##### **ItemID**

*ItemID* 是其属性列表被请求的 **ItemID**。

##### **Count**

*Count* 指示返回的属性数。

### PropertyIDs

*PropertyIDs* 是所需属性的 DWORD 类型的 ID。这些 ID 由 *QueryAvailableProperties* 输出。

### PropertyValues

*PropertyValues* 是一个由服务器输出的、与 *Count* 大小相同且带有 VARIANT 类型变量的数组；该数组包含所需属性的当前值。

### Errors

*Errors* 是一个由错误代码组成的数组，指示每个属性是否已被返回。

## 示例

```

Dim OPCItemID as String
Dim ItemCount As Long
Dim PropertyIDs(3) as Long
Dim Data() as Variant
Dim Errors() as Long
Dim AnOPCTextString As String

' Specifies the values for ItemCount and PropertyIDs...
AnOPCServer.GetItemProperties (OPCItemID, ItemCount,
                                 PropertyIDs, Data, Errors)

For i = 1 To ItemCount
    AnOPCTextString = Str(PropertyIDs(i)) + " " + Data(i)
    listbox.AddItem AnOPCTextString
Next i

```

### 7.2.2.9 LookupItemIDs

## 说明

该方法会返回一个与已传递的 *PropertyID* 相对应的 *ItemID*（如果适用）的列表。*ItemID* 可插入到 *OPCGroups* 中，并用于对相关数据项属性进行更加有效的访问。

错误数组中的错误可以指示已返回的属性 ID 不是为相关数据项定义的。

## 语法

```
LookupItemIDs (ItemID As String,
               Count As Long,
               PropertyIDs() as Long,
               ByRef NewItemIDs() As String,
               ByRef Errors () As Long)
```

### **ItemID**

*ItemID* 显示了其属性列表被请求的 **ItemID**。

### **Count**

*Count* 与返回的属性数相对应。

### **PropertyIDs**

*PropertyIDs* 是所需属性的 DWORD 类型的 ID。这些 ID 由 *QueryAvailableProperties* 输出。

### **NewItemIDs**

*NewItemIDs* 包含返回的 **ItemID** 列表。

### **Errors**

*Errors* 是一个由错误代码组成的数组，指示是否已返回 **ItemID**。

## 示例

```
Dim OPCItemID as String
Dim Count As Long
Dim PropertyIDs(1) as Long
Dim NewItemIDs () as String
Dim Errors() as Long
Dim AnOPCTextString As String
OPCItemID = "OneOPCDataAccessItem"
Count = 1
PropertyIDs(1) = 5;
AnOPCServer.LookupItemIDs (OPCItemID, Count, PropertyIDs,
                           NewItemIDs, Errors)
For i = 1 To Count
    AnOPCTextString = Str(PropertyIDs(i)) + " " +
                      NewItemIDs(i)
    listbox.AddItem AnOPCTextString
Next i
```

## 7.2.3 OPCServer 对象的事件

OPC 服务器存在的唯一事件是 ServerShutDown 事件。

### 7.2.3.1 ServerShutDown

#### 说明

当服务器将要关闭并且所有激活的客户端将终止其与服务器的连接时，系统会触发此事件。

客户端应用程序支持此命令，以使客户端按照服务器的命令终止其与服务器的连接，并清除所有组和数据项。

#### 语法

```
ServerShutDown (Reason As String)
```

##### ServerReason

服务器的这一可选文本字符串指示服务器关闭的原因。

#### 示例

```
Dim WithEvents AnOPCServer As OPCServer
Dim ARealOPCServer As String
Dim ARealOPCNodeName As String
Set AnOPCServer = New OPCServer
' this example is necessary
' to make creation of an object with the addition "WithEvents"
' easier
ARealOPCServer = "Vendorname.DataAccessCustomServer"
ARealOPCNodeName = "AComputername"
AnOPCServer.Connect(ARealOPCServer, ARealOPCNodeName)
Private Sub AnOPCServer_ServerShutDown
(ByRef aServerReason As String)
' Program code to terminate the connection to the server
End Sub
```

## 7.3 集合对象 *OPCGroups*

### 说明

**OPCGroups** 对象是一个由 **OPCGroup** 类的对象以及创建、组织和移除对象时所采用的方法组成的集合。该对象具有 **OPCGroup** 默认值的属性。当添加了一个 **OPCGroup** 类的对象时，其初始状态将由默认属性 **DefaultGroupXXXX** 指定。默认属性可以被更改以便利用不同的初始状态创建 **OPCGroups**。更改该默认属性对现有的组对象没有任何影响。在组对象已添加后，即可更改其属性。这将减少调用 *Add* 方法所必需的参数的数量。

### 示例

要使以下 VB 示例工作，需要以下代码。

```
Dim WithEvents AnOPCServer As OPCServer
Dim ARealOPCServer As String
Dim ARealOPCNodeName As String
Dim AnOPCServerBrowser As OPCBrowser
Dim MyGroups As OPCGroups
Dim DefaultGroupUpdateRate As Long
Dim OneGroup As OPCGroup
Dim AnOPCItemCollection As OPCItems
Dim AnOPCItem As OPCItem
Dim ClientHandles(100) As Long
Dim AnOPCItemIDs(100) As String
Dim AnOPCItemServerHandles() As Long
Dim AnOPCItemServerErrorHandles() As Long
Set AnOPCServer = New OPCServer
ARealOPCServer = "Vendorname.DataAccessCustomServer"
ARealOPCNodeName = "OneComputername"
AnOPCServer.Connect(ARealOPCServer, ARealOPCNodeName)
Set MyGroups = AnOPCServer.OPCGroups
MyGroups.DefaultGroupIsActive = True
Set OneGroup = MyGroups.Add("OneOPCGroupName")
Set AnOPCItemCollection = OneGroup.OPCItems
```

### 7.3.1 **OPCGroups** 对象的属性

Parent  
DefaultGroupIsActive  
DefaultGroupUpdateRate  
DefaultGroupDeadband  
DefaultGroupLocaleID  
DefaultGroupTimeBias  
Count

#### 7.3.1.1 **Parent**

##### 说明

(只读) 该属性将返回一个更高级别 OPCServer 对象的参考。

##### 语法

```
Parent As OPCServer
```

#### 7.3.1.2 **DefaultGroupIsActive**

##### 说明

(读取和写入) 该属性为一个新生成的 OPCGroup 的 *IsActive* 属性设置初始值。

##### 语法

```
DefaultGroupIsActive As Boolean
```

##### 备注

此属性的默认设置是 *True*。

## 7.3 集合对象 *OPCGroups*

### 示例

```
' Example of VB syntax (getting the property):  
Dim DefaultGroupIsActive As Boolean  
DefaultGroupIsActive = MyGroups.DefaultGroupIsActive  
  
' Example of VB syntax (specifying the property):  
MyGroups.DefaultGroupIsActive = FALSE
```

#### 7.3.1.3 DefaultGroupUpdateRate

### 说明

(读取和写入) 该属性为一个新生成的 OPCGroup 的 *UpdateRate* 属性设置初始值 (以毫秒为单位), 默认值为: 1000 毫秒 (= 1 秒)。

### 语法

```
DefaultGroupUpdateRate As Long
```

### 示例

```
' Example of VB syntax (getting the property):  
Dim DefaultGroupUpdateRate As Long  
DefaultGroupUpdateRate = MyGroups.DefaultGroupUpdateRate  
  
' Example of VB syntax (specifying the property):  
MyGroups.DefaultGroupUpdateRate = 250
```

#### 7.3.1.4 DefaultGroupDeadband

### 说明

(读取和写入) 该属性以百分比形式为使用 *Add* 创建的 OPCGroups 的 *Deadband* 属性设置初始值。系统以整个带宽的百分比形式指定 *Deadband* 属性 (有效值介于 0 到 100% 之间)。

## 语法

```
DefaultGroupDeadband As Single
```

## 备注

该属性的默认设置为“0”。如果该值 >100 或 <0，则将输出错误消息。

## 示例

```
' Example of VB syntax (getting the property):
Dim DefaultGroupDeadband As Single
DefaultGroupDeadband = MyGroups.DefaultGroupDeadband

' Example of VB syntax (specifying the property):
MyGroups.DefaultGroupDeadband = 10
```

### 7.3.1.5 DefaultGroupLocaleID

## 说明

(读取和写入) 该属性为使用 *Add* 创建的 *OPCGroup* 的 *LocaleID* 属性设置初始值。

## 语法

```
DefaultGroupLocaleID As Long
```

## 备注

该属性的默认设置与服务器的 *LocaleID* 的设置不匹配。

## 示例

```
' Example of VB syntax (getting the property):
Dim DefaultGroupLocaleID As Long
DefaultGroupLocaleID = MyGroups.DefaultGroupLocaleID

' Example of VB syntax (setting the property):
MyGroups.DefaultGroupLocaleID =
    ConvertLocaleIdStringToLocaleIdLong("English")
```

### 7.3.1.6 DefaultGroupTimeBias

#### 说明

(读取和写入)。该属性为使用 *Add* 创建的 *OPCGroup* 类对象的 *TimeBias* 属性 (=时间偏移) 设置初始值 (以分钟为单位)。

#### 语法

```
DefaultGroupTimeBias As Long
```

#### 备注

该属性的默认设置是 0 分。

#### 示例

```
' Example of VB syntax (getting the property):  
Dim DefaultGroupTimeBias As Long  
DefaultGroupTimeBias = MyGroups.DefaultGroupTimeBias  
  
' Example of VB syntax (specifying the property):  
MyGroups.DefaultGroupTimeBias = 60
```

### 7.3.1.7 Count

#### 说明

(只读) 这是组的数量。集合对象需要该属性。

#### 语法

```
Count As Long
```

## 示例

```
' Example of VB syntax:  
For index = 1 to MyGroups.Count  
    ' further code  
Next index
```

## 7.3.2 *OPCGroups* 对象的方法

*Item*  
*Add*  
*GetOPCGroup*  
*Remove*  
*RemoveAll*  
*ConnectPublicGroup*  
*RemovePublicGroup*

### 7.3.2.1 *Item*

#### 说明

此方法返回指定数据项的参考。此数据项可以通过其名称或索引进行指定（从 1 开始）。*GetOPCGroup* 将根据 *ServerHandle* 用于参考。在使用 *OPCGroups* 的情况下，*Item* 是默认方法。

#### 语法

```
Item (ItemSpecifier As Variant) As OPCGroup
```

##### **ItemSpecifier**

*Item Specifier* 是 *OPCGroup* 的 *ServerHandle* 或 *OPCGroup* 的名称。  
要使用索引进行参考，则将使用 *Item* 方法。

## 7.3 集合对象 *OPCGroups*

### 示例

```
' Example of VB syntax:  
Dim AnOPCGroup As OPCGroup  
Set AnOPCGroup = MyGroups.Item(3)  
  
' or  
Set AnOPCGroup = MyGroups("Group3")
```

#### 7.3.2.2 Add

### 说明

此方法将创建新的 **OPCGroup** 对象，并将其添加至集合。此新组的属性由 **OPCServer** 对象的当前默认设置进行指定。在一个组已添加后，其属性可根据需要进行修改。

### 语法

```
Add (Optional Name As Variant) As OPCGroup
```

#### Name

**Name** 指定组的名称。在此客户端所创建的全部组内，名称必须唯一。

如果未指定任何名称，则此服务器所生成的名称将在所有现有组中是唯一的。

### 备注

如果未指定任何名称，则服务器本身将生成一个唯一的名称。

如果指定的名称并非唯一，则在尝试调用未创建的对象时，将不会生成任何 **OPCGroup** 对象、**VB** 输出和消息。

有关错误和异常的详细信息，请参见附录中的参考“自动化接口”。

### 示例

```
MyGroups.DefaultGroupIsActive = True  
Set OneGroup = MyGroups.Add("OneOPCGroupName")
```

### 7.3.2.3 GetOPCGroup

#### 说明

此方法将参考返回至用名称或服务器句柄标识的 OPCGroup。

#### 语法

```
GetOPCGroup (ItemSpecifier As Variant) As OPCGroup
```

#### ItemSpecifier

*ItemSpecifier* 是 OPCGroup 的 ServerHandle 或 OPCGroup 的名称。  
要使用索引进行参考，则将使用 *Item* 方法。

#### 示例

```
' The example assumes that
' "OneOPCGroupName" has already been added
Set OneGroup = MyGroups.GetOPCGroup("OneOPCGroupName")
```

### 7.3.2.4 Remove

#### 说明

此方法将移除服务器上的 OPCGroup。

#### 语法

```
Remove (ItemSpecifier As Variant)
```

#### ItemSpecifier

*ItemSpecifier* 是 OPCGroup 的 ServerHandle 或 OPCGroup 的名称。  
要使用索引进行参考，则将使用 *Item* 方法。

#### 备注

如果 OPCGroup 是一个 *PublicGroup*，则将不能使用此方法。  
有关错误和异常的详细信息，请参见附录中的参考“自动化接口”。

## 7.3 集合对象 *OPCGroups*

### 示例

```
Set OneGroup = MyGroups.Add("OneOPCGroupName")
' further code
MyGroups.Remove("OneOPCGroupName")

' or

Set OneGroup = MyGroups.Add("OneOPCGroupName")
' further code
MyGroups.Remove(OneGroup.ServerHandle)
```

#### 7.3.2.5 RemoveAll

### 说明

移除 **OPCGroup** 和 **OPCItem** 类的所有现有对象以为关闭服务器做准备。

### 语法

```
RemoveAll()
```

### 备注

在删除服务器对象后，此方法可简化客户端的子对象的完全释放。当为仍然存在的 **OPCGroup** 和 **OPCItem** 的所有对象调用 *RemoveAll* 方法时，该方法与 *Remove* 方法相对应。因为 **OPCBrowser** 类的对象不是服务器的子对象，所以不能使用 *RemoveAll* 方法移除这些对象。

### 示例

```
Set OneGroup = MyGroups.Add("EinOPCGroupName")
Set OneGroup = MyGroups.Add("EinOPCGroupName1")
Set OneGroup = MyGroups.Add("EinOPCGroupName2")
' further code
MyGroups.RemoveAll
```

### 7.3.2.6 ConnectPublicGroup

#### 说明

*PublicGroups* 是服务器的默认组。可建立与这些组的连接，而不必先将其添加至服务器。

#### 语法

```
ConnectPublicGroup (Name As String) As OPCGroup
```

##### Name

*Name* 是要连接的组的名称。

#### 备注

如果名称无效或不支持 *PublicGroups*，则不能使用此方法。

有关错误和异常的更多信息，请参见“自动化接口”参考的附录。

#### 示例

```
Set OneGroup = MyGroups.ConnectPublicGroup ("AnOPCServerDefinedPublicGroup")
```

### 7.3.2.7 RemovePublicGroup

#### 说明

此方法将移除被指定为参数的 OPCGroup。

#### 语法

```
RemovePublicGroup (ItemSpecifier As Variant)
```

##### ItemSpecifier

*ItemSpecifier* 是由 *ConnectPublicGroup* 返回的 OPCGroup 的 ServerHandle 或 OPCGroup 的名称。

#### 备注

如果不支持 *PublicGroups* 或者未使用 *ConnectPublicGroup* 方法连接组，则不能使用 *RemovePublicGroup*。

有关错误和异常的详细信息，请参见附录中的参考“自动化接口”。

## 7.3 集合对象 *OPCGroups*

### 示例

```

Set OneGroup = MyGroups.ConnectPublicGroup("OneOPCGroupName")
' further code
MyGroups.RemovePublicGroup ("OneOPCGroupName")

' or

Set OneGroup = MyGroups.ConnectPublicGroup ("OneOPCGroupName")
' further code
MyGroups.RemovePublicGroup(OneGroup.ServerHandle)

```

### 7.3.3 *OPCGroups* 对象的事件

*OPCGroups* 集合对象仅具有 **GlobalDataChange** 事件。

#### 7.3.3.1 **GlobalDataChange**

##### 说明

###### *GlobalDataChange*

通过告知所有组中所有数据项的值和状态更改，来简化在所有集合组中处理事件的过程。

##### 语法

```

GlobalDataChange (TransactionID As Long,
                  GroupHandle As Long,
                  NumItems As Long,
                  ClientHandles() As Long,
                  ItemValues() As Variant,
                  Qualities() As Long,
                  TimeStamps() As Date)

```

###### **TransactionID**

*TransactionID* 是由客户端指定的事务 ID。如果值不等于 0，此事件将因 *AsyncRefresh* 而发生。如果值等于 0，此事件将因正常事件处理而发生。

###### **GroupHandle**

*GroupHandle* 是已更改数据所属的 OPCGroup 的 OPCGroup 的 ClientHandle。

###### **NumItems**

*NumItems* 指定数据项输出的数量。

**ClientHandles**

*ClientHandles* 是数据项客户端句柄的数组。

**Item Values**

*ItemValues* 是一个带有值的数组。

**Qualities**

*Qualities* 是每个数据项的值的质量数组。

**TimeStamps**

*TimeStamps* 是每个数据项值的带有 UTC 时间戳的数组。

**备注**

请注意，*OnDataChange* 事件通常应与 *OPCGroup* 配合使用。此事件允许设置 *EventHandle*，以便其能够处理来自 *OPCGroups* 类的若干对象的数据更改。

通常，一个应用程序的每个组都会有一个 *EventHandle* 以允许接收和处理数据更改。这意味着，您仅需要一个 *EventHandle*，通过使用 *GroupHandle*，触发事件的组将会显而易见。

为 *OPCGroup* 类的每个对象触发 *GlobalDataChange*

事件，而这个类包含值或状态在事件被最后一次触发后已发生更改的数据项。系统也会触发 *OPCGroup* 对象的相关事件。

如果应用程序使用全局和组特定的事件，则它将收到两个通知：

一个是针对全局特定事件的通知，一个是针对组特定事件的通知。

## 7.4 OPCGroup 对象

### 示例

```

Dim WithEvents AnOPCGroupCollection As OPCGroups
Private Sub AnOPCGroupCollection_GlobalDataChange
    (TransactionID As Long,
     GroupHandle As Long,
     MasterQuality As Long,
     MasterError As Long,
     NumItems As Long,
     ClientHandles() As Long,
     ItemValues() As Variant,
     Qualities() As Long,
     TimeStamps() As Date)

    ' Here, enter the client code for processing changed
    ' values
    .
    .
End Sub

```

## 7.4 OPCGroup 对象

### 说明

一个客户端可以使用 **OPCGroup** 管理数据。

例如，此组可以在特定用户接口或报告中表示数据项。可以读取和写入数据。

必要时，可启用和禁用从组中的数据项到客户端的核查。客户端可以组态 OPC 服务器中数据更改的更新率。

### 语法

**OPCGroup**

### 示例

要使以下 VB 示例工作，需要以下代码。

```

Dim WithEvents AnOPCServer As OPCServer
Dim ARealOPCServer As String
Dim ARealOPCNodeName As String

```

```
Dim AnOPCServerBrowser As OPCBrowser
Dim MyGroups As OPCGroups
Dim DefaultGroupUpdateRate As Long
Dim WithEvents OneGroup As OPCGroup
Dim AnOPCItemCollection As OPCItems
Dim AnOPCItem As OPCItem
Dim ClientHandles(100) As Long
Dim AnOPCItemIDs(100) As String
Dim AnOPCItemServerHandles() As Long
Dim AnOPCItemServerErrorErrors() As Long
Set AnOPCServer = New OPCServer
ARealOPCServer = "Vendorname.DataAccessCustomServer"
ARealOPCNodeName = "OneComputername"
AnOPCServer.Connect(ARealOPCServer, ARealOPCNodeName)
Set MyGroups = AnOPCServer.OPCGroups
MyGroups.DefaultGroupIsActive = True
Set OneGroup = MyGroups.Add("OneOPCGroupName")
Set AnOPCItemCollection = OneGroup.OPCItems
For x = 1 To AddItemCount
    ClientHandles(x) = x + 1
    AnOPCItemID(x) = "Register_" & x
Next x
AnOPCItemCollection.AddItems AddItemCount, AnOPCItemIDs,
    AnOPCItemServerHandles, AnOPCItemServerErrorErrors
```

## 7.4.1 OPCGroup 对象的属性

Parent  
Name  
IsPublic  
IsActive  
IsSubscribed  
ClientHandle  
ServerHandle  
LocaleID  
TimeBias  
DeadBand  
UpdateRate  
OPCItems

### 7.4.1.1 Parent

#### 说明

(只读) 该属性将返回更高级别 OPCServer 对象的参考。

#### 语法

```
Parent As OPCServer
```

### 7.4.1.2 Name

#### 说明

(读取和写入) **Name** 包含组的名称。

#### 语法

```
Name As String
```

#### Name

**Name** 包含组的名称。在客户端所创建的全部组内，名称必须唯一。

## 备注

作为选择，组可以被指定一个名称。 用户可以使用此属性分配一个名称。  
名称必须唯一。 如果未指定任何名称，则服务器将使用 **OPCGroups** 对象的 *Add* 方法为组生成一个唯一名称。

## 示例

```
' Example of VB syntax: (getting the property):
Dim CurrentValue As String
Set OneGroup = MyGroups.Add("OneOPCGroupName")
CurrentValue = OneGroup.Name

' Example of VB syntax (specifying the property):
Set OneGroup = MyGroups.Add("OneOPCGroupName")
OneGroup.Name = "OneName"
```

### 7.4.1.3 IsPublic

## 说明

(只读) 如果涉及 **PublicGroup**，则此属性将返回 *True*，否则将返回 *False*。

## 语法

```
IsPublic As Boolean
```

## 示例

```
Dim CurrentValue As Boolean
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("OneOPCGroupName")
' further code
Set CurrentValue = OneGroup.IsPublic
' this obtains the value
```

#### 7.4.1.4 IsActive

##### 说明

(读取和写入) 此属性指定组是否处于激活状态。激活的组包含数据。  
通常，未激活的组将不包含任何数据，除非其对于读写操作必不可少。

##### 语法

```
IsActive As Boolean
```

##### 备注

此属性的默认设置是添加时间 *Add()* 的 OPCGroups 对象值的默认值。

##### 示例

```
' Example of VB syntax: (getting the property):
Dim CurrentValue As Boolean
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("OneOPCGroupName")
' further code
Set CurrentValue = OneGroup.IsActive
' this obtains the value

' Example of VB syntax (specifying the property):
Dim CurrentValue As Boolean
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("OneOPCGroupName")
' further code
OneGroup.IsActive = True
```

#### 7.4.1.5 IsSubscribed

##### 说明

(读取和写入) 此属性将检查组的非同步消息。  
在服务器上订阅的组将收到来自服务器的有关已更改的数据的通知。

## 语法

```
IsSubscribed As Boolean
```

## 备注

此属性的默认设置是添加时间 *Add()* 的 OPCGroups 对象值的默认值。

## 示例

```
' Example of VB syntax: (getting the property):
Dim CurrentValue As Boolean
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("OneOPCGroupName")
' further code
Set CurrentValue = OneGroup.IsSubscribed
' this obtains the value

' Example of VB syntax (specifying the property):
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("OneOPCGroupName")
' further code
OneGroup.IsSubscribed = True
' this specifies the value
```

### 7.4.1.6 ClientHandle

## 说明

(读取和写入) *ClientHandle* 是一个代表组的长值。

此属性允许客户端快速标识数据的接收。

通常，此值是一个索引或类似物，并与有关状态的数据或信息一起返回至客户端。

## 语法

```
ClientHandle As Long
```

## 7.4 OPCGroup 对象

### 示例

```
' Example of VB syntax (getting the property):  
Dim CurrentValue As Long  
Set MyGroups = AnOPCServer.OPCGroups  
Set OneGroup =  
    MyGroups.ConnectPublicGroup ("OneOPCGroupName")  
    ' further code  
Set CurrentValue = OneGroup.ClientHandle  
    ' this obtains the value  
  
' Example of VB syntax (specifying the property):  
Set MyGroups = AnOPCServer.OPCGroups  
Set OneGroup =  
    MyGroups.ConnectPublicGroup ("OneOPCGroupName")  
    ' further code  
OneGroup.ClientHandle = 1975  
    ' this obtains the value
```

#### 7.4.1.7 ServerHandle

### 说明

(只读) 这是分配至组的服务器的句柄。

此属性的值被指定为长值；这允许对组进行唯一标识。在使用 **OPCItem** 类（例如 **OPCGroups.Remove**）的对象进行操作时，客户端必须将此句柄作为参数传输至多个方法。

### 语法

```
ServerHandle As Long
```

## 示例

```
' Example of VB syntax (getting the property):
Dim CurrentValue As Long
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("OneOPCGroupName")
' further code
Set CurrentValue = OneGroup.ServerHandle
' this obtains the value
```

### 7.4.1.8 LocaleID

#### 说明

(读取和写入) 此属性会标识语言

ID; 服务器返回的字符串可以使用此属性进行本地化。此属性的默认值取决于在 OPCGroups 集合中定义的值 (*DefaultGroupLocaleID*)。

#### 语法

```
LocaleID As Long
```

#### 示例

```
' Example of VB syntax (getting the property):
Dim CurrentValue As Long
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("OneOPCGroupName")
' further code
Set CurrentValue = OneGroup.LocaleID

' Example of VB syntax (specifying the property):
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("OneOPCGroupName")
' further code
OneGroup.LocaleID = StringToLocaleID("English")
```

#### 7.4.1.9 TimeBias

##### 说明

(读取和写入) 此属性需要将数据的时间戳转换为设备的本地时间。单位：毫秒。

##### 语法

```
TimeBias As Long
```

##### 示例

```
' Example of VB syntax (getting the property):  
Dim CurrentValue As Long  
Set MyGroups = AnOPCServer.OPCGroups  
Set OneGroup = MyGroups.ConnectPublicGroup("OneOPCGroupName")  
' further code  
Set CurrentValue = OneGroup.TimeBias  
  
' Example of VB syntax (specifying the property):  
Set MyGroups = AnOPCServer.OPCGroups  
Set OneGroup = MyGroups.ConnectPublicGroup("OneOPCGroupName")  
' further code  
OneGroup.TimeBias = 100
```

#### 7.4.1.10 DeadBand

##### 说明

(读取和写入) 系统以整个带宽的百分比形式指定此属性的值(有效值介于 0 到 100 之间)。此属性的默认值取决于在 OPCGroups 集合中定义的值 (*DefaultGroupDeadband*)。

##### 语法

```
DeadBand As Single
```

## 示例

```
' Example of VB syntax (getting the property):
Dim CurrentValue As Single
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("OneOPCGroupName")
' further code
Set CurrentValue = OneGroup.DeadBand

' Example of VB syntax (specifying the property):
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("OneOPCGroupName")
' further code
OneGroup.DeadBand = 5
```

### 7.4.1.11 UpdateRate

#### 说明

(读取和写入) DataChange 事件被触发的最大频率 (毫秒)。

如果过程较慢，则可能导致数据更改发生的频率不如此频率高，并且无论如何都决不会更频繁。此属性的默认值取决于在 **OPCGroups** 集合中定义的值

(*DefaultGroupUpdateRate*)。此属性的新值通过对某个新值的查询进行指定。

服务器无法支持特定频率，并且在读入属性时可能产生的结果是一个不同的频率（服务器使用下一个高于请求频率的频率）。

#### 语法

UpdateRate As Long

## 7.4 OPCGroup 对象

### 示例

```
' Example of VB syntax (getting the property):  
Dim CurrentValue As Long  
Set MyGroups = AnOPCServer.OPCGroups  
Set OneGroup = MyGroups.ConnectPublicGroup("EinOPCGroupName")  
' further code  
DefaultGroupUpdateRate = OneGroup.UpdateRate  
  
' Example of VB syntax (specifying the property):  
Set MyGroups = AnOPCServer.OPCGroups  
Set OneGroup = MyGroups.ConnectPublicGroup("EinOPCGroupName")  
' further code  
OneGroup.UpdateRate = 50
```

### 7.4.1.12 OPCItems

#### 说明

OPCItems 是 OPCItem 对象的集合；这是 OPCGroup 对象的默认属性。

#### 语法

```
OPCItems As OPCItems
```

#### 示例

```
' Example of VB syntax (getting the property):  
Dim AnOPCItemCollection As OPCItems  
Set MyGroups = AnOPCServer.OPCGroups  
Set OneGroup = MyGroups.ConnectPublicGroup("OneOPCGroupName")  
' further code  
Set AnOPCItemCollection = OneGroup.OPCItems
```

## 7.4.2 OPCGroup 对象的方法

SyncRead  
SyncWrite  
AsyncRead  
AsyncWrite  
AsyncRefresh  
AsyncCancel

### 7.4.2.1 SyncRead

#### 说明

此函数为一个数据项或一个组的多个数据项读取值、质量和时间戳。

#### 语法

```
SyncRead (Source As Integer,
          NumItems As Long,
          ServerHandles() As Long,
          ByRef Values() As Variant,
          ByRef Errors() As Long,
          Optional ByRef Qualities As Variant,
          Optional ByRef TimeStamps As Variant)
```

#### Source

*Source* 是数据源; OPC\_DS\_CACHE 或 OPC\_DS\_DEVICE

#### NumItems

*NumItems* 包含要读取的数据项的数量。

#### ServerHandles

*ServerHandles* 是要读取的数据项的服务器句柄的数组。

#### Values

*Values* 是一个值的数组。

#### Errors

*Errors* 是一个长值的数组, 用于指定相关数据项是否成功读取, 换言之, 是否使用 *Read* 方法收到了已定义的值、质量和时间戳。

注意: 如果在错误代码中输出 *FAILED*, 则将不会定义值、质量和时间戳 (未定义) !

### Qualities

*Qualities* 含有一个 VARIANT 类型的值，而该值包含了带有与质量相关的值的整数数组。

### TimeStamps

*TimeStamps* 含有 VARIANT 类型的值，而该值包含了带有 UTC 时间戳的数据数组。

如果设备不能返回时间戳，则它将由服务器进行创建。

## 备注

在输出结果前，功能已完成。可以从缓存中读取数据；在这种情况下，数据应符合 *UpdateRate* 和组的 *Deadband* 属性。

数据也可以从设备进行读取；在这种情况下，数据实际上应从实际设备中读取。

缓存和设备读取方法的准确执行不会在此处进行更详细的处理。

仅当组和数据项处于激活状态，从缓存读取的数据才会有效。

如果这两个元素中的一个元素处于未激活状态，则会输出是常量

`OPC_QUALITY_OUT_OF_SERVICE`，而非质量。

组或数据项的状态（激活/未激活）对设备读取方法没有任何影响。

## 示例

```
Private Sub ReadButton_Click()
    Dim Source As Integer
    Dim NumItems As Long
    Dim ServerIndex As Long
    Dim ServerHandles(10) As Long
    Dim Values() As Variant
    Dim Errors() As Long
    Dim Qualities() As Variant
    Dim TimeStamps() As Variant
    Source = OPC_DS_DEVICE
    NumItems = 10
    For ServerIndex = 1 to NumItems
        ' specifies which item will be read
        ServerHandles(ServerIndex) = AnOPCItemServerHandles(ServerIndex)
    Next ServerIndex
    OneGroup.SyncRead Source,
                    NumItems,
                    ServerHandles,
                    Values,
                    Errors,
                    Qualities,
                    TimeStamps
    For ServerIndex = 1 to NumItems
        ' processes the values
        TextBox(ServerIndex).Text = Values(ServerIndex)
    Next ServerIndex
End Sub
```

### 7.4.2.2 SyncWrite

#### 说明

此方法为组的一个或多个数据项写入值。该功能被完全执行。

值将被写入设备；这意味着，只有在确定设备已接受数据（或拒绝数据）之后，才会返回函数。

## 语法

```
SyncWrite (NumItems As Long,  
          ServerHandles() As Long,  
          Values() As Variant,  
          ByRef Errors() As Long)
```

### **NumItems**

*NumItems* 指定要写入的数据项的数量。

### **ServerHandles**

*ServerHandles* 是要写入的数据项的服务器句柄的数组。

### **Values**

*Values* 是一个值的数组。

### **Errors**

*Errors* 是指定数据项是否已成功写入的长值的数组。

## 备注

组或数据项的状态（激活/未激活）对写入方法没有任何影响。

## 示例

```

Private Sub WriteButton_Click()
    Dim Source As Integer
    Dim NumItems As Long
    Dim ServerIndex As Long
    Dim ServerHandles() As Long
    Dim Values() As Variant
    Dim Errors() As Long
    NumItems = 10
    For ServerIndex = 1 to NumItems
        ' specifies which item will be written
        ServerHandles(ServerIndex) = AnOPCItemServerHandles(ServerIndex)
        Values(ServerIndex) = ServerIndex * 2
        ' the value of the item can be any value here
    Next ServerIndex
    OneGroup.SyncWrite NumItems, ServerHandles, Values, Errors
    For ServerIndex = 1 to NumItems
        ' processes the errors
        TextBox(ServerIndex).Text = Errors(ServerIndex)
    Next ServerIndex
End Sub

```

### 7.4.2.3 AsyncRead

#### 说明

此方法读取一个组的一个或多个数据项。 系统将返回结果以及已分配至 OPCGroup 对象的 *AsyncReadComplete* 事件。

#### 语法

```

AsyncRead (NumItems As Long,
           ServerHandles() As Long,
           ByRef Errors() As Long,
           TransactionID As Long,
           ByRef CancelID As Long)

```

#### NumItems

*NumItems* 包含要读取的数据项的数量。

#### **ServerHandles**

*ServerHandles* 是要读取的数据项的服务器句柄的数组。

#### **Errors**

*Errors* 是指定要读取的数据项状态的长值数组。

#### **TransactionID**

*TransactionID* 是由客户端指定的事务 ID。此 ID 被包含在相关事件中。

#### **CancelID**

*CancelID* 是由服务器生成的事务 ID，客户端可以使用此 ID 取消事务。

### 备注

*AsyncRead* 方法假设 OPCGroup 类的对象是使用事件声明 (Dim WithEvents xxx As OPCGroup)，以便 *AsyncRead* 的结果能够返回至客户端应用程序。已分配至 OPCGroup 对象的 *AsyncReadComplete* 事件由带有来自 *AsyncRead* 的结果的自动化服务器触发。

组或数据项的状态（激活/未激活）对设备读取方法没有任何影响。

有关更多详细信息，请参见 *OPC 数据访问自定义接口规范* 中的 *IOPCAsyncIO2::Read*。

## 示例

```

Private Sub AsyncReadButton_Click()
    Dim NumItems As Long
    Dim ServerIndex As Long
    Dim ServerHandles(10) As Long
    Dim Values() As Variant
    Dim Errors() As Long
    Dim ClientTransactionID As Long
    Dim ServerTransactionID As Long
    Dim Qualities() As Variant
    Dim TimeStamps() As Variant
    NumItems = 10
    ClientTransactionID = 1975
    For ServerIndex = 1 to NumItems
        ' specifies which item will be read
        ServerHandles(ServerIndex) = AnOPCItemServerHandles(ServerIndex)
    Next ServerIndex
    OneGroup.AsyncRead NumItems, ServerHandles, Errors,
                      ClientTransactionID, ServerTransactionID
End Sub

```

### 7.4.2.4 AsyncWrite

#### 说明

此方法为组中的一个或多个数据项写入值。 系统将返回结果以及已分配至 OPCGroup 对象的 *AsyncWriteComplete* 事件。

#### 语法

```

AsyncWrite (NumItems As Long,
           ServerHandles() As Long,
           Values() As Variant,
           ByRef Errors() As Long,
           TransactionID As Long,
           ByRef CancelID As Long)

```

#### NumItems

*NumItems* 指定要写入的数据项的数量。

**ServerHandles**

*ServerHandles* 是要写入的数据项的服务器句柄的数组。

**Values**

*Values* 是一个值的数组。

**Errors**

*Errors* 是指定要写入的数据项状态的长值数组。

**TransactionID**

*TransactionID* 是由客户端指定的事务 ID。此 ID 被包含在相关事件中。

**CancelID**

*CancelID* 是由服务器生成的事务 ID，客户端可以使用此 ID 取消事务。

**备注**

*AsyncWrite* 方法假设 OPCGroup 类的对象是使用事件标注尺寸 (Dim WithEvents xxx As OPCGroup)，以便 *AsyncWrite* 的结果能够返回至客户端应用程序。已分配至 OPCGroup 对象的 *AsyncWriteComplete* 事件由带有来自 *AsyncWrite* 的结果的自动化服务器触发。

有关更多详细信息，请参见 *OPC* 数据访问自定义接口规范中的 *IOPCAsyncIO2::Write*。

## 示例

```

Private Sub AsyncWriteButton_Click()
    Dim NumItems As Long
    Dim ServerIndex As Long
    Dim ServerHandles(10) As Long
    Dim Values() As Variant
    Dim Errors() As Long
    Dim ClientTransactionID As Long
    Dim ServerTransactionID As Long
    NumItems = 10
    For ServerIndex = 1 to NumItems
        ClientTransactionID = 1957
        ' specifies which item will be written
        ServerHandles(ServerIndex) = AnOPCItemServerHandles(ServerIndex)
        Values(ServerIndex) = ServerIndex * 2
        ' the value of the item can be any value here
    Next ServerIndex
    OneGroup.AsyncWrite NumItems, ServerHandles, Values, Errors,
                        ClientTransactionID, ServerTransactionID
End Sub

```

### 7.4.2.5 AsyncRefresh

#### 说明

此方法为组的所有激活的数据项创建了一个事件（指示它们是否已发生更改）。不会执行未激活的数据项。系统将输出结果、已分配至 **OPCGroup** 对象的 *DataChange* 事件和已分配至 **OPCGroup** 对象的 *GlobalDataChange* 事件。

#### 语法

```

AsyncRefresh (Source As Integer,
              TransactionID As Long,
              ByRef CancelID As Long)

```

##### Source

*Source* 是数据源；OPC\_DS\_CACHE 或 OPC\_DS\_DEVICE

##### TransactionID

*TransactionID* 是由客户端指定的事务 ID。此 ID 被包含在相关事件中。

**CancellID**

*CancellID* 是由服务器生成的事务 ID，客户端可以使用此 ID 取消事务。

**备注**

*AsyncRefresh* 方法假设 OPCGroup 类的对象是使用事件标注尺寸 (Dim WithEvents xxx As OPCGroup)，以便 *AsyncRefresh* 的结果能够返回至客户端应用程序。已分配至 OPCGroup 对象的 *DataChange* 事件由带有来自 *AsyncRefresh* 的结果的自动化服务器触发。

如果客户端应用程序使用 *WithEvents* 补充声明 OPCGroups 类的对象 (Dim WithEvents xyz As OPCGroups)，则此组的 *GlobalDataChange* 事件将使用数据更新的结果进行触发。

有关更多详细信息，请参见 *OPC* 数据访问自定义接口规范中的 *IOPCAsyncIO2::Refresh*。

**说明**

OPC 自动化接口的 *Async2RefreshDevice* 函数的调用将返回所有数据项，甚至“只读”的数据项。

**示例**

```

Dim MyGroups As OPCGroups
Dim DefaultGroupUpdateRate As Long
Dim WithEvents OneGroup As OPCGroup
Private Sub AsyncRefreshButton_Click()
    Dim ServerIndex As Long
    Dim Source As Long
    Dim ClientTransactionID As Long
    Dim ServerTransactionID As Long
    ClientTransactionID = 2125
    Source = OPC_DS_DEVICE
    OneGroup.AsyncRefresh Source, ClientTransactionID
        ServerTransactionID
End Sub

```

### 7.4.2.6 AsyncCancel

#### 说明

使用此方法，服务器将被请求取消所有未完成的事务。 *AsyncCancelComplete* 事件指示是否进行取消操作。

#### 语法

```
AsyncCancel (CancelID As Long)
```

##### CancelID

*CancelID* 是由方法 *AsyncRead*、*AsyncWrite* 或 *AsyncRefresh* 之一返回的、通过服务器创建的 *CancelID*。

#### 备注

*AsyncCancel* 方法假设 OPCGroup 类的对象是使用事件声明 (Dim WithEvents xxx As OPCGroup)，以便 *AsyncCancel* 的结果能够返回至客户端应用程序。已分配至 OPCGroup 对象的 *AsyncCancelComplete* 事件由带有来自 *AsyncCancel* 的结果的自动化服务器触发。由客户端指定的 TransactionID 将被返回至带有 *AsyncCancelComplete* 事件的自动化客户端应用程序。

有关更多详细信息，请参见 *OPC 数据访问自定义接口规范* 中的 *IOPCAsyncIO2::Cancel*。

#### 示例

```
Private Sub AsyncCancelButton_Click()
    Dim ServerIndex As Long
    Dim CancelID As Long
    CancelID = 1 ' a transaction ID
    ' from one of the "Async" calls
    ' such as AsyncRead, AsyncWrite or AsyncRefresh
    OneGroup.AsyncCancel CancelID
End Sub
```

### 7.4.3 OPCGroup 对象的事件

DataChange  
 AsyncReadComplete  
 AsyncWriteComplete  
 AsyncCancelComplete

#### 7.4.3.1 DataChange

##### 说明

当数据项的值或该值的质量在组内发生更改时，此事件会被触发。  
 此事件的触发速度不会比组更新率所指定的速度快。  
 这意味着，数据项的值将被服务器缓冲，直到当前时间 + 更新率比先前更新的时间长。  
 此行为也取决于组和数据项是否处于激活状态。  
 数据项和相应的组必须先处于激活状态，然后值才能被发送至事件中的客户端。

##### 语法

```
DataChange (TransactionID As Long,
           NumItems As Long,
           ClientHandles() As Long,
           ItemValues() As Variant,
           Qualities() As Long,
           TimeStamps() As Date)
```

##### TransactionID

*TransactionID* 是由客户端指定的事务 ID。如果输出的值不等于 0，调用将作为 *AsyncRefresh* 方法的结果进行生成。如果值等于 0，则调用将作为正常事件处理的结果进行生成。

##### NumItems

*NumItems* 指定数据项输出的数量。

##### ClientHandles

*ClientHandles* 是数据项客户端句柄的数组。

##### Item Values

*ItemValues* 是一个带有值的数组。

##### Qualities

*Qualities* 是每个数据项的值的质量数组。

**TimeStamps**

*TimeStamps* 是每个数据项值的带有 UTC 时间戳的数组。

如果设备不能返回时间戳，则它将由服务器进行处理。

**备注**

如果数据项值的更改速度比更新率快，则将仅缓冲每个数据项的最后当前值，并将其返回至事件中的客户端。

**示例**

```
Dim WithEvents AnOPCGroup As OPCGroup

Private Sub AnOPCGroup_DataChange (TransactionID As Long,
                                   NumItems As Long, ClientHandles() As Long,
                                   ItemValues() As Variant, Qualities() As Long,
                                   TimeStamps() As Date)

    ' insert your program code here,
    ' to process the changed data
End Sub
```

**7.4.3.2 AsyncReadComplete****说明**

在 *AsyncRead* 方法完成时触发此事件。

**语法**

```
AsyncReadComplete (TransactionID As Long,
                   NumItems As Long,
                   ClientHandles() As Long,
                   ItemValues() As Variant,
                   Qualities() As Long,
                   TimeStamps() As Date,
                   Errors() As Long)
```

**TransactionID**

*TransactionID* 是由客户端指定的事务 ID。

**NumItems**

*NumItems* 指定数据项输出的数量。

**ClientHandles**

*ClientHandles* 是数据项客户端句柄的数组。

**ItemValues**

*ItemValues* 是一个带有值的数组。

**Qualities**

*Qualities* 是每个数据项的值的质量数组。

**TimeStamps**

*TimeStamps* 是每个数据项值的带有 UTC 时间戳的数组。

如果设备不返回时间戳，则它将由服务器进行处理。

**Errors**

*Errors* 是一个长值的数组，用于指定相关数据项是否成功读取，换言之，是否使用 *Read* 方法获得了已定义的值、质量和时间戳。如果在区域代码中输出 *FAILED*，则将不会定义值、质量和时间戳（未定义）！

**示例**

```
Dim WithEvents AnOPCGroup As OPCGroup

Private Sub AnOPCGroup_AsyncReadComplete (
    TransactionID As Long,
    NumItems As Long,
    ClientHandles() As Long,
    ItemValues() As Variant,
    Qualities() As Long,
    TimeStamps() As Date)

    ' insert your program code here,
    ' to process the changed data
End Sub
```

**7.4.3.3 AsyncWriteComplete****说明**

在 *AsyncWrite* 方法完成时触发此事件。

## 语法

```
AsyncWriteComplete (TransactionID As Long,
                    NumItems As Long,
                    ClientHandles() As Long,
                    Errors() As Long)
```

### **TransactionID**

*TransactionID* 是由客户端指定的事务 ID。

### **NumItems**

*NumItems* 指定数据项输出的数量。

### **ClientHandles**

*ClientHandles* 是数据项客户端句柄的数组。

### **Errors**

*Errors* 是指定数据项是否已成功写入的长值的数组。

## 示例

```
Dim WithEvents AnOPCGroup As OPCGroup
Private Sub AnOPCGroup_AsyncWriteComplete (TransactionID As Long,
                                           NumItems As Long, ClientHandles() As Long, ItemValues() As Variant,
                                           Qualities() As Long, TimeStamps() As Date)
    ' insert your program code here,
    ' to process the errors
End Sub
```

### 7.4.3.4 AsyncCancelComplete

## 说明

在 **AsyncCancel** 方法完成时触发此事件。

## 语法

```
AsyncCancelComplete (TransactionID As Long)
```

### **TransactionID**

*TransactionID* 是由客户端指定的事务 ID。

## 示例

```

Dim WithEvents AnOPCGroup As OPCGroup
Private Sub AnOPCGroup_AsyncCancelComplete(TransactionID As Long)
    ' Enter your program code here ,
    ' that will be executed after the operation is canceled
End Sub

```

## 7.5 集合对象 *OPCItems*

### 说明

此集合对象具有 *OPCItem* 类对象的默认值的属性。

如果添加此类数据项，则会为数据项的属性提供这些默认值 (*DefaultXXXX*)

作为其初始状态。如果想要添加具有不同初始状态的数据项，则可以更改这些默认值。

当然，在添加数据项后，也可以更改数据项的属性。这将减少调用 *AddItems* 方法所必需的参数的数量。

### 语法

*OPCItems*

### 示例

要使以下 VB 示例工作，需要以下代码：

```

Dim AnOPCServer As OPCServer
Dim ARealOPCServer As String
Dim ARealOPCNodeName As String
Dim AnOPCServerBrowser As OPCBrowser
Dim MyGroups As OPCGroups
Dim DefaultGroupUpdateRate As Long
Dim OneGroup As OPCGroup
Dim AnOPCItemCollection As OPCItems
Dim AnOPCItem As OPCItem
Dim ClientHandles(100) As Long
Dim AnOPCItemIDs(100) As String
Dim AnOPCItemServerHandles(10) As Long
Dim AnOPCItemServerErrorErrors() As Long

```

```

Set AnOPCServer = New OPCServer
ARealOPCServer = "Vendorname.DataAccessCustomServer"
ARealOPCNodeName = "OneComputername"
AnOPCServer.Connect(ARealOPCServer, ARealOPCNodeName)
Set MyGroups = AnOPCServer.OPCGroups
MyGroups.DefaultGroupIsActive = True
Set OneGroup = MyGroups.Add("OneOPCGroupName")
Set AnOPCItemCollection = OneGroup.OPCItems

```

## 7.5.1 **OPCItems** 集合对象的属性

Parent  
**DefaultRequestedDataType**  
**DefaultAccessPath**  
**DefaultIsActive**  
Count

### 7.5.1.1 Parent

#### 说明

(只读) 该属性将返回更高级别 **OPCGroup** 对象的参考。

#### 语法

```
Parent As OPCGroup
```

### 7.5.1.2 DefaultRequestedDataType

#### 说明

(读取和写入) **DefaultRequestedDataType** 是用于 *Add* 调用的已请求的数据类型 (*AddItem*、*AddItems*) ; 换言之, 新添加数据项的 **RequestedDataType** 属性的初始值。此属性的默认设置是 VT\_EMPTY (换言之, 服务器将以其自身的数据格式发送数据)。

## 语法

```
DefaultRequestedDataType As Integer
```

## 备注

每个有效的 VARIANT 类型都可以被指定为已请求的数据类型。  
有关错误和异常的详细信息，请参见附录中的参考“自动化接口”。

## 示例

```
' Example of VB syntax (getting the property):
Dim CurrentValue As Integer
Dim SomeValue As Integer
CurrentValue = AnOPCItemCollection.DefaultRequestedDataType

' Example of VB syntax (specifying the property):
AnOPCItemCollection.DefaultRequestedDataType = SomeValue
```

### 7.5.1.3 DefaultAccessPath

## 说明

(读取和写入) *DefaultAccessPath* 是 *Add*  
调用中使用的默认访问路径 (*AddItem*、*AddItems*)。此属性的默认设置是  
"" (空字符串)。

## 语法

```
DefaultAccessPath As String
```

## 示例

```
' Example of VB syntax (getting the property):
Dim CurrentValue As String
Dim SomeValue As String
CurrentValue = AnOPCItemCollection.DefaultAccessPath

' Example of VB syntax (specifying the property):
AnOPCItemCollection.DefaultAccessPath = SomeValue
```

### 7.5.1.4 DefaultIsActive

#### 说明

(读取和写入) ) 该属性为新添加数据项的 *ActiveState* 属性设置初始值。  
此属性的默认值是 *True*。

#### 语法

```
DefaultIsActive As Boolean
```

#### 示例

```
' Example of VB syntax (getting the property):
Dim CurrentValue As Boolean
Dim SomeValue As Boolean
CurrentValue = AnOPCItemCollection.DefaultIsActive

' Example of VB syntax (specifying the property):
AnOPCItemCollection.DefaultIsActive = SomeValue
```

### 7.5.1.5 Count

#### 说明

(只读) 集合对象需要该属性。

#### 语法

```
Count As Long
```

#### 示例

```
' Example of VB syntax (getting the property):
Dim CurrentValue As Long
Dim SomeValue As Long
CurrentValue = AnOPCItemCollection.Count
```

## 7.5.2 *OPCItems* 集合对象的方法

*Item*  
*GetOPCItem*  
*AddItem*  
*AddItems*  
*Remove*  
*Validate*  
*SetActive*  
*SetClientHandles*  
*SetDataTypes*

### 7.5.2.1 *Item*

#### 说明

集合对象需要该方法。

#### 语法

```
Item (ItemSpecifier As Variant) As OPCItem
```

#### **ItemSpecifier**

*ItemSpecifier* 是集合内以 1 开始的索引，并且它将使用 *ItemSpecifier* 返回 *OPCItem* 类的对象。

#### 备注

*Item* 将参考返回至 *ItemSpecifier* 索引所描述的集合的数据项。 *ItemSpecifier* 是集合的索引（从 1 开始）。 *GetOPCItem* 使用 *ServerHandle* 参考数据项。

自动化属性 *Item* 经常与 *OPCItem* 对象相混淆。 自动化属性 *Item* 是一个由自动化集合用于参考所包含数据项的特殊保留属性。 *OPCItem* 对象是可包含在 *OPC* 数据项集合中的特定于 *OPC* 自动化的对象类型。

## 7.5.2.2 *GetOPCItem*

#### 说明

此方法将返回属于服务器句柄的 *OPCItem*。 服务器句柄由 *Add* 方法提供 (*AddItem*、*AddItems*)。 使用 *Item* 属性，根据索引进行参考。

## 语法

```
GetOPCItem (ServerHandle As Long) As OPCItem
```

### **ServerHandle**

这是 **OPCItem** 类对象的 **ServerHandle**。 使用 *Item*, 根据索引进行参考。

## 示例

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(SomeItemServerHandle)
```

### 7.5.2.3      AddItem

## 说明

此方法将创建 **OPCItem** 类的新对象，并将其添加到集合。此新对象的属性将由 **OPCItems** 集合对象中的当前默认值进行确定。如果添加了 **OPCItem** 类的对象，则可以更改其属性。

## 语法

```
AddItem (ItemID As String,
          ClientHandle As Long)
```

### **ItemID**

*ItemID* 是完整的 ItemID。

### **ClientHandle**

这是相应组的 ClientHandle。

## 备注

使用此属性，一次只能向集合添加一个数据项。

如果想要一次添加多个数据项，则应使用 **AddItems** 方法。

有关错误和异常的详细信息，请参见附录中的参考“自动化接口”。

## 示例

```

Dim AnOPCItemID as String
Dim AnClientHandle as Long
AnOPCItemID = "N7:0"
AnClientHandle = 1975
AnOPCItemCollection.AddItem AnOPCItemID AnClientHandle

```

### 7.5.2.4 AddItems

#### 说明

此方法将创建 **OPCItem** 类的新对象，并将其添加到集合。此新对象的属性将由 **OPCItems** 集合对象中的当前默认值进行确定。如果添加了 **OPCItem** 类的对象，则可以更改其属性。

#### 语法

```

AddItems (Count As Long,
          ItemIDs() As String,
          ClientHandles() As Long,
          ByRef ServerHandles() As Long,
          ByRef Errors() As Long,
          Optional RequestedDataTypes As Variant,
          Optional AccessPaths As Variant)

```

##### **Count**

*Count* 指示所涉及数据项的数量。

##### **ItemIDs**

*ItemIDs* 包含带有完整 **ItemID** 的数组。

##### **ClientHandles**

*ClientHandles* 是用于所处理数据项的客户端句柄的数组。

##### **ServerHandles**

*ServerHandles* 是所处理数据项的服务器句柄的数组。

##### **Errors**

*Errors* 是指示各数据项操作成功的长值的数组。

**RequestedDataTypes**

*RequestedDataTypes* 是包含已请求的数据类型的整数数组的 VARIANT 类型的可选变量。

**AccessPaths**

*AccessPaths* 是包含字符串数组的 VARIANT 类型的可选参数。

**备注**

有关错误和异常的详细信息，请参见附录中的参考“自动化接口”。

**示例**

```
Dim AddItemCount as long
Dim AnOPCItemIDs() as String
Dim AnOPCItemServerHandles as long
Dim AnOPCItemServerErrors as long
Dim AnOPCRequestedDataTypes as variant
Dim AnOPCAccessPathss as variant
For x = 1 To AddItemCount
    ClientHandles(x) = x + 1
    AnOPCItemID(x) = "Register_" & x
Next x
AnOPCItemCollection.AddItems AddItemCount,
                               AnOPCItemIDs,
                               ClientHandles,
                               AnOPCItemServerHandles,
                               AnOPCItemServerErrors,
                               AnOPCRequestedDataTypes,
                               AnOPCAccessPaths
' Code for the error handling
' individual errors are listed in the "Errors" array
```

**7.5.2.5 Remove****说明**

此方法将移除 OPCItem 类的对象。

## 语法

```
Remove (Count As Long,
        ServerHandles() As Long,
        ByRef Errors() As Long)
```

### **Count**

*Count* 指定要移除数据项的数量。

### **ServerHandles**

这是所涉及数据项的服务器句柄的数组。

### **Errors**

*Errors* 是指示各数据项操作成功的长值的数组。

## 示例

```
AnOPCItemCollection.Remove AnOPCItemServerHandles,
                           AnOPCItemServerErrors
                           ' Code for the error handling
                           ' individual errors are listed in the "Errors" array
```

### 7.5.2.6 **Validate**

## 说明

此方法将检查是否可以在不添加任何对象的情况下使用 *Add* 方法 (*AddItem*、*AddItems*) 成功创建 OPCServer 类的一个或多个对象。

## 语法

```
Validate (Count As Long,
          ItemIDs() As String,
          ByRef Errors() As Long,
          Optional RequestedDataTypes As Variant,
          Optional AccessPaths As Variant)
```

### **Count**

*Count* 指示所涉及数据项的数量。

### **ItemIDs**

*ItemIDs* 是带有完整 *ItemIDs* 的数组。

**Errors**

**Errors** 是指示各数据项操作成功的长值的数组。

**RequestedDataTypes**

*RequestedDataTypes* 是包含所请求数据类型的整数数组的变量。

**AccessPaths**

这是包含 **AccessPaths** 的字符串数组的变量。

**备注**

有关错误和异常的详细信息，请参见附录中的参考“自动化接口”。

**示例**

```

Dim AddItemCount as long
Dim AnOPCItemIDs() as String
Dim AnOPCItemServerHandles as long
Dim AnOPCItemServerErrors as long
Dim AnOPCRequestedDataTypes as variant
Dim AnOPCAccessPathss as variant
For x = 1 To AddItemCount
    ClientHandles(x) = x + 1
    AnOPCItemID(x) = "Register_" & x
Next x
AnOPCItemCollection.Validate AddItemCount,
                               AnOPCItemIDs,
                               AnOPCItemServerErrors,
                               AnOPCRequestedDataTypes,
                               AnOPCAccessPaths
' Code for the error handling
' individual errors are listed in the "Errors" array

```

**7.5.2.7 SetActive****说明**

使用此方法，**OPCItem** 类的各个对象可以在 **OPCItems** 集合中进行激活或禁用。

## 语法

```
SetActive (Count As Long,
           ServerHandles() As Long,
           ActiveState As Boolean,
           ByRef Errors() As Long)
```

### **Count**

*Count* 指示所涉及数据项的数量。

### **ServerHandles**

*ServerHandles* 是所涉及数据项的服务器句柄的数组。

如果数据项处于激活状态，**ActiveState**

*ActiveState* 将输出值 *True*; 如果数据项处于未激活状态，则其将输出值 *False*。

### **Errors**

指示各数据项操作成功的长值的数组。

## 备注

有关错误和异常的详细信息，请参见附录中的参考“自动化接口”。

## 示例

```
' Activate items (assign corresponding
' property the value TRUE)
AnOPCItemCollection.SetActive ItemCount,
                               AnOPCItemServerHandles,
                               TRUE,
                               AnOPCItemServerError
                               AnOPCItemServerError

' Code for the error handling
' individual errors are listed in the "Errors" array
```

### 7.5.2.8      **SetClientHandles**

## 说明

更改组的一个或多个数据项的 **ClientHandles**。

## 语法

```
SetClientHandles (Count As Long,
                  ServerHandles() As Long,
                  ClientHandles() (As Long,
                  ByRef Errors() As Long)
```

### **Count**

*Count* 指示所涉及数据项的数量。

### **ServerHandles**

*ServerHandles* 是所处理数据项的服务器句柄的数组。

### **ClientHandles**

*ClientHandles* 是要存储的新客户端句柄的数组。 客户端句柄不必唯一。

### **Errors**

*Errors* 是指示各数据项操作成功的长值的数组。

## 示例

```
For x = 1 To ItemCount
    ClientHandles(x) = x + 1975
Next x
AnOPCItemCollection.SetClientHandles ItemCount,
                                      AnOPCItemServerHandles,
                                      ClientHandles,
                                      AnOPCItemServerErrors
```

### 7.5.2.9 SetDataTypes

## 说明

此方法将更改一个或多个数据项所需的数据类型。

## 语法

```
SetDataTypes (Count As Long,
              ServerHandles() As Long,
              RequestedDataTypes() As Long,
              ByRef Errors() As Long)
```

**Count**

*Count* 指示所涉及数据项的数量。

**ServerHandles**

*ServerHandles* 是所处理数据项的服务器句柄的数组。

**RequestedDataTypes**

*RequestedDataTypes* 是包含要存储的新指定的数据类型的数组。

**Errors**

*Errors* 是指示各数据项操作成功的长值的数组。

**备注**

有关错误和异常的详细信息，请参见附录中的参考“自动化接口”。

**示例**

```
Dim RequestedDataTypes(100) As Long
For x = 1 To ItemCount
    RequestedDataTypes(x) = "a VB integer"
Next x
AnOPCItemCollection.SetDataTypes ItemCount,
                                AnOPCItemServerHandles,
                                RequestedDataTypes,
                                AnOPCItemServerErrors
```

**7.6**

**OPCItem 对象**

OPCItem 类的对象表示服务器内与数据源的连接。 值、质量和时间戳属于每个数据项。 值被指定为 VARIANT 类型的变量并具有一个质量。

## 7.6.1 OPCItem 对象的属性

Parent  
ClientHandle  
ServerHandle  
AccessPath  
AccessRights  
ItemID  
IsActive  
RequestedDataType  
Value  
Quality  
TimeStamp  
CanonicalDataType  
EUType  
EUInfo

### 7.6.1.1 Parent

#### 说明

(只读) 该属性将返回更高级别 OPCGroup 对象的参考。

#### 语法

```
Parent As OPCGroup
```

### 7.6.1.2 ClientHandle

#### 说明

(读写) 它返回属于 OPCItem 的长整型值。

客户端根据此属性将客户端识别为数据目标。 ClientHandle 通常是 (或类似于) 索引, 数据或状态随 OPCGroup 对象的事件发生变化时, 它返回到客户端。

#### 语法

```
ClientHandle As Long
```

## 示例

```
Dim AnOPCItem as OPCItem  
Set OPCItem = GetOPCItem(EinItemServerHandle)  
  
' Example of VB syntax (getting the property):  
Dim CurrentValue As Long  
Dim SomeValue As Long  
CurrentValue = AnOPCItem.ClientHandle  
  
' Example of VB syntax (specifying the property):  
AnOPCItem.ClientHandle = SomeValue
```

### 7.6.1.3 ServerHandle

#### 说明

(只读) 这是服务器分配给 OPCItem 的句柄，可使用它唯一标识 OPCItem。  
在有多种方法处理 OPCItem 类的对象时（例如  
*OPCItems.Remove*），客户端必须使该句柄可用于这些方法。

#### 语法

```
ServerHandle As Long
```

#### 示例

```
Dim AnOPCItem as OPCItem  
Set OPCItem = GetOPCItem(OneItemServerHandle)  
  
' Example of VB syntax (getting the property):  
Dim CurrentValue As Long  
Dim SomeValue As Long  
CurrentValue = AnOPCItem.ServerHandle
```

### 7.6.1.4 AccessPath

#### 说明

(只读) *AccessPath* 提供客户端使用 *Add* 调用 (*AddItem*、*AddItems*) 指定的访问路径。

#### 语法

```
AccessPath As String
```

#### 示例

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(OneItemServerHandle)

' Example of VB syntax (getting the property):
Dim CurrentValue As String
Dim SomeValue As String
CurrentValue = AnOPCItem.AccessPath
```

### 7.6.1.5 AccessRights

#### 说明

(只读) *AccessRights* 输出数据项的访问权限。

#### 语法

```
AccessRights As Long
```

#### 备注

此属性指明对应数据项是否存在只读、只写或读写权限。

## 示例

```
Dim AnOPCItem as OPCItem  
Set OPCItem = GetOPCItem(OneItemServerHandle)  
  
' Example of VB syntax (getting the property):  
Dim CurrentValue As Long  
Dim SomeValue As Long  
CurrentValue = AnOPCItem.AccessRights
```

### 7.6.1.6      **ItemID**

#### 说明

(只读) *ItemID* 是数据项的唯一标识。

#### 语法

```
ItemID As String
```

#### 示例

```
Dim AnOPCItem as OPCItem  
Set OPCItem = GetOPCItem(EinItemServerHandle)  
  
' Example of VB syntax (getting the property):  
Dim CurrentValue As String  
Dim SomeValue As String  
CurrentValue = AnOPCItem.ItemID
```

### 7.6.1.7      **IsActive**

#### 说明

(读写) 此属性指定是否为此数据项生成通知事件。

#### 语法

```
IsActive As Boolean
```

## 备注

如果数据项已激活，则输出值 *True*，如果值未激活，则输出 *False*。

## 示例

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(OneItemServerHandle)

' Example of VB syntax (getting the property):
Dim CurrentValue As Boolean
Dim SomeValue As Boolean
CurrentValue = AnOPCItem.IsActive

' Example of VB syntax (specifying the property):
AnOPCItem.IsActive = SomeValue
```

### 7.6.1.8 RequestedDataType

## 说明

(读写) *RequestedDataType* 是输出数据项值的数据类型。

注意：

如果请求的数据类型被拒绝，则数据项失效，直到请求的数据类型具有有效值为止。

## 语法

```
RequestedDataType As Integer
```

## 备注

有关错误和异常的详细信息，请参见附录中的参考“自动化接口”。

## 示例

```
Dim AnOPCItem as OPCItem  
Set OPCItem = GetOPCItem(OneItemServerHandle)  
  
' Example of VB syntax (getting the property):  
Dim CurrentValue As Integer  
Dim SomeValue As Integer  
CurrentValue = AnOPCItem.RequestedDataType  
  
' Example of VB syntax (specifying the property):  
AnOPCItem.RequestedDataType = SomeValue
```

### 7.6.1.9 Value

#### 说明

(只读) 此属性输出从服务器读取的最后一个值。这是 *OPCItem* 的默认属性。

#### 语法

```
Value As Variant
```

#### 示例

```
Dim AnOPCItem as OPCItem  
Set OPCItem = GetOPCItem(OneItemServerHandle)  
  
' Example of VB syntax (getting the property):  
Dim CurrentValue As Variant  
Dim SomeValue As Variant  
CurrentValue = AnOPCItem.Value
```

### 7.6.1.10 Quality

#### 说明

(只读) 此属性输出从服务器读取的最后一个质量值。

## 语法

```
Quality As Long
```

## 示例

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(OneItemServerHandle)

' Example of VB syntax (getting the property):
Dim CurrentValue As Long
Dim SomeValue As Long
CurrentValue = AnOPCItem.Quality
```

### 7.6.1.11 TimeStamp

## 说明

(只读) 此属性输出从服务器读取的最后一个时间戳值。

## 语法

```
TimeStamp As Date
```

## 示例

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(EinItemServerHandle)

' Example of VB syntax (getting the property):
Dim CurrentValue As Date
Dim SomeValue As Date
CurrentValue = AnOPCItem.TimeStamp
```

### 7.6.1.12 CanonicalDataType

#### 说明

(只读) 此属性输出数据项的原始数据类型。此值可能不同于请求的数据类型 (RequestedDataType)。

#### 语法

```
CanonicalDataType As Integer
```

#### 备注

有关错误和异常的详细信息，请参见附录中的参考“自动化接口”。

#### 示例

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(OneItemServerHandle)

' Example of VB syntax (getting the property):
Dim CurrentValue As Integer
Dim SomeValue As Integer
CurrentValue = AnOPCItem.CanonicalDataType
```

### 7.6.1.13 EUType

#### 说明

(只读) *EUType* 属性指定 EUInfo 属性所含 EU 信息的类型 (EU = 工程单位)。

#### 语法

```
EUType As Integer
```

## 备注

0 表示没有 EU 信息可用 (EUInfo 属性的值为 VT\_EMPTY)。

1 表示模拟量; EU 信息包含只由两个 Double 类型变量组成的 SAFEARRAY (VT\_ARRAY | VT\_R8), 两个变量分别对应于最小值和最大值。

2 表示枚举; EU 信息包含由字符串组成的 SAFEARRAY (VT\_ARRAY | VT\_BSTR), 其中包含一列字符串 (例如 OPEN、CLOSE 和 IN TRANSIT 等), 分别对应于连续数值 (0、1 和 2 等)。

有关详细信息, 请参见 *OPCItem* 属性中的 *OPC* 数据访问自定义接口规范。

## 示例

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(OneItemServerHandle)

' Example of VB syntax (getting the property):
Dim CurrentValue As Integer
Dim SomeValue As Integer
CurrentValue = AnOPCItem.EUType
```

### 7.6.1.14 EUInfo

#### 说明

(只读) *EUInfo* 包含带有单位信息的变量 (EU = 工程单位)。

#### 语法

```
EUInfo As Variant
```

#### 备注

有关详细信息, 请参见 *OPCItem* 属性中的 *OPC* 数据访问自定义接口规范。

## 示例

```

Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(OneItemServerHandle)

' Example of VB syntax (getting the property):
Dim CurrentValue As Variant
Dim SomeValue As Variant
CurrentValue = AnOPCItem.EUIInfo

```

## 7.6.2 OPCItem 对象的方法

**Read**

**Write**

### 7.6.2.1 Read

#### 说明

可通过此方法从服务器读取数据项。只能通过一个源（OPCCache 或 OPCDevice）调用 **Read** 来更新数据项的值、质量和时间戳。

如果值、质量和时间戳需要同步，则此方法的可选参数同时返回更新值。

#### 语法

```

Read (Source As Integer,
      Optional ByRef Value As Variant,
      Optional ByRef Quality As Variant,
      Optional ByRef TimeStamp As Variant)

```

##### Source

*Source* 表示数据源；即 OPC\_DS\_CACHE 或 OPC\_DS\_DEVICE

##### Value

*Value* 输出从服务器读取的最后一个值。

##### Quality

*Quality* 输出从服务器读取的最后一个质量值。

##### Timestamp

*Timestamp* 输出从服务器读取的最后一个时间戳。

## 示例

```
Private Sub ReadButton_Click()
    Dim AnOPCItem as OPCItem
    Set OPCItem = GetOPCItem(OneItemServerHandle)
    Dim Source As Integer
    Dim Value As Variant
    Dim Quality As Variant
    Dim TimeStamp As Variant
    Source = OPC_DS_DEVICE
    AnOPCItem.Read Source,
        ServerHandles,
        Value,
        Quality,
        TimeStamp
    ' processing of the values
    TextBox.Text = Value
End Sub
```

### 7.6.2.2 Write

#### 说明

可通过此方法将值写入 OPC 服务器。

#### 语法

```
Write (Value As Variant)
```

##### Value

*Value* 包含要写入的值。

## 示例

```
Private Sub WriteButton_Click()
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(OneItemServerHandle)
Dim Value As Variant
Value = 1975
AnOPCItem.Write Value
End Sub
```

# 7.7 定义

在此，您将找到与服务器状态和错误信息解释有关的常量定义。

## 7.7.1 服务器状态

### OPCRunning

服务器正常运行，这是正常的服务器状态。

### OPCFailed

服务器发生供应商特定错误，服务器不再正常运行。

这种情况下如何消除问题取决于供应商。通常，另一种服务器方法输出错误代码 *E\_FAIL*。

### OPCNoconfig

服务器正在运行但未加载任何组态信息；也就是说，服务器并未正常运行。  
不需要此类信息的服务器不输出此状态消息。

### OPCSuspended

服务器运行被供应商特定方法临时中断，也就是说，服务器并未收发数据。  
有关质量的信息如下所示： OPC\_QUALITY\_OUT\_OF\_SERVICE。

**OPCTest**

服务器处于测试模式。 数据输出与硬件断开连接，其它方面服务器响应正常。

根据供应商的实现方法，数据输入基于真正存在或仿真得出的值。 正常输出质量信息。

**OPCDisconnected**

自动化服务器对象并未连接到 OPC 自定义接口服务器。

## 7.7.2 错误消息

**OPCInvalidHandle****0xC0040001L**

此句柄的值无效。 客户端不能将无效句柄传递到服务器。

如果发生此错误，说明客户端或在服务器上可能存在编程错误。

**OPCBadType****0xC0040004L**

服务器无法在所需数据类型和原始数据类型之间转换数据。

**OPCPublic****0xC0040005L**

无法对声明为 *public* 的组执行所需操作。

**OPCBadRights****0xC0040006L**

数据项的访问权限不允许所需操作。

**OPCUknownItemID****0xC0040007L**

ItemID 未在服务器的地址空间内定义或已不存在（针对读写操作）。

**OPCInvalidItemID**

**0xC0040008L**

ItemID 与服务器的语法不匹配。

**OPCInvalidFilter**

**0xC0040009L**

过滤器的字符串无效。

**OPCUncnownPath**

**0xC004000AL**

数据项的访问路径对服务器来说是未知路径。

**OPCRage**

**0xC004000BL**

值超出有效范围。

**OPCDuplicateName**

**0xC004000CL**

*DuplicateName* 无效。

**OPCUncnspportedRate**

**0x0004000DL**

服务器不支持所需数据速率，但会使用最接近的速率。

**OPCClamp**

**0x0004000EL**

已接受但未输出写入值。

**OPCInuse**

**0x0004000FL**

操作无法执行，因为仍有对象引用。

**OPCInvalidConfig****0xC0040010L**

服务器组态的文件格式无效。

**OPCNotFound****0xC0040011L**无法找到所需对象（例如 *PublicGroup*）。**OPCInvalidPID****0xC0040203L**

所传递数据项的 PropertyID 无效。

## 7.8 自动化接口参考附录

### 错误

如果发生运行错误，则 Visual Basic 的 *Err* 对象明确标识错误。

如果 VB 没有使用 *On Err* 机制处理错误，则产生异常，根据上下文（处于调试模式的 Visual Basic 或可执行程序），会显示一个带有下列信息的消息框：

运行错误：以十进制数字（十六进制字符）表示的错误代码

对象 *Y* 的方法 *X* 失败（如果使用可执行应用程序，不为 *X* 和 *Y* 指定值）。

因此，OPC

基金会强烈建议应用程序采取适当措施检测在指定属性或执行方法时可能出现的 OPC 自动化错误。

错误处理程序是检测应用程序中的错误并对其做出响应的例行程序。

如果应用程序指定属性或调用方法，OPC

自动化客户端应为每项应用程序功能提供错误处理程序。

错误处理程序由以下三部分组成：

1. 创建或激活错误情况分支，指定发生错误时应用程序应跳转到何处（使用哪一例程进行错误处理）。*On Error* 中的语句激活分支并将应用程序带到指明例程起点的标签。
2. 编写错误处理例程处理因指定属性或使用方法而产生的错误。
3. 退出例程。

如果发生错误，必须指明应用程序应采取何种措施。

例如，如果正在添加的一个组的名称（用户指定）不惟一，则可通知用户该组并未添加；也可提示用户输入其它名称。另一方面，应用程序也可尝试重新添加该组（通过“”），然后采用服务器生成的名称。

## 找出错误

如果出现错误，则在 VB 执行 **On Error**

语句时激活一个指定错误处理程序的引用。只要包含分支的程序本身处于激活状态，分支就保持激活状态；也就是说直到运行 *Exit Sub*、*Exit Function*、*Exit Property*、*End Sub*、*End Function* 或 *End Property* 为止。

要创建一个跳转到错误处理例程的分支，使用 *On Error Go To Line* 语句，其中 *Line* 是标识错误处理代码的标签。

## 处理错误

首先在错误处理例程开始时设置标签。标签名称应具备说明性且必须以冒号结尾。

主体部分包含 *If(...Then(...Else(...))* 循环或 *Case* 语句形式的错误处理代码。

在此执行用于处理最常见错误的方法。

*Err* 对象中的 *Number* 属性包含数字代码，表示近期最常发生的运行错误。

此属性的错误编号包含值，用于调用 *GetErrorString* 将错误编号转换为可读字符串。

## 示例

```
Dim AnOpcServer As OPCServer
Private Sub Command1_Click()
    On Error GoTo testerror
    Set AnOpcServer = New OPCServer
    ' if "fuzz" does not exist so that the connection
    ' cannot be established, there is a jump to the label
    ' "testerror"
    AnOpcServer.Connect ("fuzz")
    Time = AnOpcServer.CurrentTime
    Debug.Print Time
    testerror:
    Debug.Print Err.Number
End Sub
```

## 相关文献

您要阅读 OPC 基金会的规范或需要有关特定主题的详细信息。

您感兴趣的领域	请参见:
有关特定主题的详细信息	其它可用文献
有关 OPC 数据访问和 OPC 报警和事件的详细信息	OPC 规范

### 查找 SIMATIC NET 文档

- 目录

可以在以下目录中找到 Siemens 相关产品的订货号:

- SIMATIC NET 工业通信/工业标识, 目录 IK PI
- 用于全集成自动化和小型自动化的 SIMATIC 产品, 目录 ST 70

可以从 Siemens 代表处获得这些目录和其它信息。

可从以下 Internet 网址访问 Industry Mall:

<https://mall.industry.siemens.com>

- Internet 上的文档

在 Siemens 自动化客户支持 Internet 页面上可找到 SIMATIC NET 手册:

<http://support.automation.siemens.com>

转到所需产品组并进行以下设置:

“条目列表”(Entry list) 选项卡, 条目类型“手册/操作说明”(Manuals / Operating Instructions)

- STEP 7 安装文档

可以通过开始菜单“开始 > 所有程序 > Siemens Automation > 文档”(Start > All Programs > Siemens Automation > Documentation) 找到 PG/PC 上 STEP 7 产品在线文档中包含的手册。

## 参见

SIMATIC NET 手册: (<http://support.automation.siemens.com/WW/view/zh/10805878>)

SIMATIC 文档: (<http://www.siemens.com/simatic-docu>)

SIMATIC NET

PROFIBUS 网络手册

Siemens AG

(SIMATIC NET Manual Collection)

请参见 Internet 上的以下条目 ID: 35222591

(<http://support.automation.siemens.com/WW/view/zh/35222591>)

SIMATIC NET

工业以太网手册

Siemens AG

(SIMATIC NET Manual Collection)

请参见 Internet 上的以下条目 ID: 27069465

(<http://support.automation.siemens.com/WW/view/zh/27069465>)

SIMATIC NET

CP 5613/CP 5614 的编程接口

Siemens AG

(SIMATIC NET Manual Collection)

请参见 Internet 上的以下条目 ID: 1653531

(<http://support.automation.siemens.com/WW/view/zh/1653531>)

SIMATIC NET

IO Base 用户接口

Siemens AG

(SIMATIC NET Manual Collection)

请参见 Internet 上的以下条目 ID: 19779901

(<http://support.automation.siemens.com/WW/view/zh/19779901>)

## 8.1 OPC 规范

### OPC 规范有哪些？

OPC 基金会的第一项也是最基本的一项规范是 OPC 数据访问规范。此规范提供过程变量管理以及访问这些变量的多种途径。

OPC 报警和事件规范涉及过程报警和事件的传送。它与数据访问规范无关。

到目前为止，OPC 基金会已发布以下规范（SIMATIC NET 手册集均有收录）：

- **OPC 概述**

**版本 1.0, 1998 年 10 月 27 日**

OPC 基本概念简介

- **OPC 通用定义和接口**

**版本 1.0, 1998 年 10 月 27 日**

此文档介绍所有 OPC 服务器都有的重要接口。

- **数据访问自定义接口**

**版本 1.1, 1997 年 9 月 11 日**

数据访问自定义接口规范，版本 1.1

- **数据访问自定义接口**

**版本 2.04, 2000 年 9 月 5 日**

数据访问自定义接口规范，版本 2.04

- **数据访问自定义接口**

**版本 2.05, 2001 年 12 月 17 日**

数据访问自定义接口规范，版本 2.05

- **数据访问自定义接口**

**版本 3.00, 2003 年 3 月 4 日**

数据访问自定义接口规范，版本 3.00

- **数据访问自动化接口**

**版本 2.02, 1999 年 2 月 4 日**

数据访问自动化接口规范，版本 2.02

## 8.1 OPC 规范

- **OPC 报警和事件自定义接口**

**版本 1.10, 2002 年 10 月 2 日**

OPC 报警和事件服务器的说明以及此服务器的自定义接口规范

- **报警和事件自动化接口**

**版本 1.01, 1999 年 12 月 15 日**

OPC 报警和事件服务器自动化接口规范

- **OPC XML-DA 规范**

**版本 1.01, 2004 年 12 月 21 日**

数据访问 OPC XML 接口规范

- **OPC 统一架构**

**1.0 或更高版本**

<http://www.opcfoundation.org/UA/>

第 1 部分 - 概念

第 2 部分 - 安全模型

第 3 部分 - 地址空间模型

第 4 部分 - 服务

第 5 部分 - 信息模型

第 6 部分 - 服务映射

第 7 部分 - 配置文件

第 8 部分 - 数据访问

第 9 部分 - 报警和条件

第 10 部分 - 程序

第 11 部分 - 历史访问

第 12 部分 - 搜索

**OPC UA 规范。**

有关自动化任务的附加规范随后会陆续发表。