

## SIMATIC NET

### PC-Software

### Industrielle Kommunikation mit PG/PC Band 2 - Schnittstellen

Programmierhandbuch

Vorwort

1

OPC-Prozessvariablen für  
SIMATIC NET

2

OPC Alarms & Events-  
Server für SIMATIC NET

3

OPC-Server nutzen

4

SIMATIC Computing

5

Beispielprogramme

6

Referenz Automation-  
Schnittstelle

7

Literaturhinweise

8

# Rechtliche Hinweise

## Warnhinweiskonzept

Dieses Handbuch enthält Hinweise, die Sie zu Ihrer persönlichen Sicherheit sowie zur Vermeidung von Sachschäden beachten müssen. Die Hinweise zu Ihrer persönlichen Sicherheit sind durch ein Warndreieck hervorgehoben, Hinweise zu alleinigen Sachschäden stehen ohne Warndreieck. Je nach Gefährdungsstufe werden die Warnhinweise in abnehmender Reihenfolge wie folgt dargestellt.

### GEFAHR

bedeutet, dass Tod oder schwere Körperverletzung eintreten **wird**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

### WARNUNG

bedeutet, dass Tod oder schwere Körperverletzung eintreten **kann**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

### VORSICHT

bedeutet, dass eine leichte Körperverletzung eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

### ACHTUNG

bedeutet, dass Sachschaden eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Beim Auftreten mehrerer Gefährdungsstufen wird immer der Warnhinweis zur jeweils höchsten Stufe verwendet. Wenn in einem Warnhinweis mit dem Warndreieck vor Personenschäden gewarnt wird, dann kann im selben Warnhinweis zusätzlich eine Warnung vor Sachschäden angefügt sein.

## Qualifiziertes Personal

Das zu dieser Dokumentation zugehörige Produkt/System darf nur von für die jeweilige Aufgabenstellung **qualifiziertem Personal** gehandhabt werden unter Beachtung der für die jeweilige Aufgabenstellung zugehörigen Dokumentation, insbesondere der darin enthaltenen Sicherheits- und Warnhinweise. Qualifiziertes Personal ist auf Grund seiner Ausbildung und Erfahrung befähigt, im Umgang mit diesen Produkten/Systemen Risiken zu erkennen und mögliche Gefährdungen zu vermeiden.

## Bestimmungsgemäßer Gebrauch von Siemens-Produkten

Beachten Sie Folgendes:

### WARNUNG

Siemens-Produkte dürfen nur für die im Katalog und in der zugehörigen technischen Dokumentation vorgesehenen Einsatzfälle verwendet werden. Falls Fremdprodukte und -komponenten zum Einsatz kommen, müssen diese von Siemens empfohlen bzw. zugelassen sein. Der einwandfreie und sichere Betrieb der Produkte setzt sachgemäßen Transport, sachgemäße Lagerung, Aufstellung, Montage, Installation, Inbetriebnahme, Bedienung und Instandhaltung voraus. Die zulässigen Umgebungsbedingungen müssen eingehalten werden. Hinweise in den zugehörigen Dokumentationen müssen beachtet werden.

## Marken

Alle mit dem Schutzrechtsvermerk ® gekennzeichneten Bezeichnungen sind eingetragene Marken der Siemens AG. Die übrigen Bezeichnungen in dieser Schrift können Marken sein, deren Benutzung durch Dritte für deren Zwecke die Rechte der Inhaber verletzen kann.

## Haftungsausschluss

Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so dass wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden regelmäßig überprüft, notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten.

# Inhaltsverzeichnis

<b>1</b>	<b>Vorwort</b>	<b>17</b>
1.1	Wie arbeite ich mit der Dokumentation?	18
1.2	Bestimmungen	19
1.2.1	Welche juristischen Bestimmungen müssen Sie beachten?	19
1.2.2	Welche Sicherheitsbestimmungen müssen Sie kennen?	20
<b>2</b>	<b>OPC-Prozessvariablen für SIMATIC NET</b>	<b>21</b>
2.1	Welche Kommunikationsfunktionen gibt es?	21
2.2	Was sind Prozessvariablen?	21
2.3	Wie werden die ItemIDs der Prozessvariablen gebildet?	22
2.4	PROFIBUS-DP	24
2.4.1	DP Master Klasse 1	25
2.4.1.1	Hochperformanter SIMATIC NET In-Process-Server für das PROFIBUS DP-Protokoll	25
2.4.1.2	Performer SIMATIC NET OPC-Server für das PROFIBUS DP-Protokoll	26
2.4.1.3	Einordnung der DPC1- und DPC2-Dienste	28
2.4.1.4	Prozessvariablen für Dienste des Master Klasse 1	29
2.4.1.5	Syntax der Prozessvariablen für den Master Klasse 1	30
2.4.1.6	Protokoll-ID	31
2.4.1.7	Projektierter CP-Name	31
2.4.1.8	Beispiele für Prozessvariablen für den Master Klasse 1	32
2.4.1.9	DPC1-Dienste	33
2.4.1.10	Syntax der Prozessvariablen für DPC1-Dienste	33
2.4.1.11	Beispiele für Prozessvariablen für DPC1-Dienste	35
2.4.1.12	Fast Logic für CP 5613/CP 5614 und CP 5623/CP 5624	36
2.4.1.13	Syntax der Steuervariablen für Fast Logic	36
2.4.1.14	DP-spezifische Informationsvariablen	38
2.4.1.15	Syntax der DP-spezifischen Informationsvariablen	38
2.4.1.16	Beispiele für DP-spezifische Informationsvariablen	43
2.4.1.17	Syntax der systemspezifischen Informationsvariablen	44
2.4.2	DP Master Klasse 2	44
2.4.2.1	Protokoll-ID	44
2.4.2.2	Projektierter CP-Name	44
2.4.2.3	Syntax der Prozessvariablen für Master Diagnose	45
2.4.2.4	Syntax der Prozessvariablen für Slave Diagnose	52
2.4.2.5	Syntax der Prozessvariablen für I/O Daten	58
2.4.2.6	Syntax der Prozessvariablen für Datensätze	60
2.4.2.7	Beispiele für Prozessvariablen für Datensätze	63
2.4.2.8	Syntax der DP2-spezifischen Informationsvariablen	64
2.4.2.9	Syntax der systemspezifischen Informationsvariablen	65
2.4.3	DP-Slave	65
2.4.3.1	Variabldienste zum Zugriff auf lokale Slave-Daten	66
2.4.3.2	Syntax der Prozessvariablen für den DP-Slave	66
2.4.3.3	Beispiele für Prozessvariablen für den DP-Slave	67
2.4.3.4	DP-Slave-spezifische Informationsvariablen	68

2.4.3.5	Syntax der DP-Slave-spezifischen Informationsvariablen.....	68
2.5	PROFIBUS-DP mit OPC UA.....	69
2.5.1	SIMATIC NET OPC-UA-Server für das DP-Protokoll.....	70
2.5.2	Unterstützung der DP-Dienste unter OPC UA.....	72
2.5.3	Wie wird der DP-OPC-UA-Server adressiert? .....	74
2.5.4	Welche Namensräume bietet der DP-OPC-UA-Server an? .....	76
2.5.5	Die Nodeld .....	77
2.5.6	Board-Objekte für DP-Baugruppen .....	79
2.5.6.1	Übersicht der Board-Objekte für DP-Baugruppen .....	79
2.5.6.2	Board-Namen.....	80
2.5.6.3	Typ-Definition des DP-Master Boardobjekts.....	82
2.5.6.4	Typ-Definition des DP-Slave Boardobjekts.....	83
2.5.7	DP Master Klasse 1 .....	84
2.5.7.1	Prozessvariablen für Dienste des Master Klasse 1 .....	84
2.5.7.2	Syntax der Prozessvariablen für den Master Klasse 1 .....	84
2.5.7.3	Beispiele für Prozessvariablen für den Master Klasse 1 .....	86
2.5.7.4	DPC1-Dienste .....	87
2.5.7.5	Syntax der Prozessvariablen für DPC1-Dienste.....	87
2.5.7.6	Beispiele für Prozessvariablen für DPC1-Dienste .....	90
2.5.7.7	Fast Logic für CP 5613/CP 5614/CP 5623/CP 5624 (nur Master) .....	91
2.5.7.8	Syntax der Steuervariablen für Fast Logic.....	91
2.5.7.9	Steuertelegramme Sync und Freeze .....	93
2.5.7.10	Syntax der Steuervariablen für Sync und Freeze .....	94
2.5.7.11	DP-spezifische Informationsvariablen .....	96
2.5.8	DP-Slave .....	107
2.5.8.1	Variabldienste zum Zugriff auf lokale DP-Slave-Daten.....	107
2.5.8.2	Syntax der Prozessvariablen für den DP-Slave.....	108
2.5.8.3	Beispiele für Prozessvariablen für den DP-Slave .....	109
2.5.8.4	Status des DP-Slave .....	110
2.5.8.5	Betriebszustand des DP-Slave .....	110
2.5.8.6	DP-Slave-spezifische Informationsvariablen .....	111
2.5.8.7	Syntax der DP-Slave-spezifischen Informationsvariablen.....	111
2.5.9	DP-OPC-UA-Template-Datenvariablen .....	114
2.5.9.1	DP-OPC-UA-Template-Datenvariablen .....	114
2.6	S7-Kommunikation.....	117
2.6.1	Performer SIMATIC NET OPC-Server für das S7-Protokoll.....	118
2.6.2	Protokoll-ID .....	120
2.6.3	Verbindungsnamen .....	120
2.6.4	Variabldienste .....	122
2.6.4.1	Syntax der Prozessvariablen für S7-Variabldienste .....	122
2.6.4.2	Beispiele für Prozessvariablen für S7-Variabldienste.....	126
2.6.4.3	Beispiele optimal strukturierter Items.....	127
2.6.5	Blockorientierte Dienste .....	128
2.6.5.1	Syntax der Prozessvariablen für Blockorientierte Dienste.....	129
2.6.5.2	Beispiele für Prozessvariablen für Blockorientierte Dienste .....	132
2.6.6	S7-spezifische Informationsvariablen .....	133
2.6.6.1	Syntax der S7-spezifischen Informationsvariablen.....	133
2.6.6.2	S7-spezifische Diagnosevariablen.....	136
2.6.6.3	Syntax der systemspezifischen Informationsvariablen.....	141
2.6.7	Bausteindienste.....	141
2.6.7.1	Syntax der Steuervariablen für Bausteindienste.....	142

2.6.7.2	Beispiele für die Verwendung der Bausteindienste .....	145
2.6.8	Passwörter .....	147
2.6.8.1	Syntax der Steuervariablen für Passwörter .....	148
2.6.8.2	Beispiel für die Verwendung der Passwörter .....	148
2.6.9	Server-Dienste .....	149
2.6.9.1	Beispiel für die Verwendung von Server-Diensten .....	149
2.6.10	S7 Template-Datenvariablen .....	151
2.6.10.1	Template-Datenvariablen im Namensraum .....	153
2.6.10.2	Syntax der Template-Datenvariablen .....	154
2.6.11	Unprojektierte S7-Verbindung .....	155
2.6.12	COML S7-Verbindung.....	160
2.7	S7-Kommunikation mit OPC UA.....	161
2.7.1	Eigenschaften der S7-Kommunikation mit OPC UA.....	161
2.7.2	SIMATIC NET OPC-UA-Server für das S7-Protokoll .....	162
2.7.3	Wie wird der S7-OPC-UA-Server adressiert?.....	165
2.7.4	Welche Namensräume bietet der S7-OPC-UA-Server an?.....	167
2.7.5	Die Nodeld .....	168
2.7.6	Verbindungsobjekte .....	170
2.7.6.1	Verbindungsnamen.....	171
2.7.7	Aufbau und Funktionen des produktiven S7-Verbindungsobjekts.....	172
2.7.7.1	Typ-Definition des S7-Verbindungsobjekts.....	173
2.7.7.2	S7-Verbindungs-Informationsobjekte .....	174
2.7.7.3	Beispiele für S7-spezifische Informationsvariablen und Rückgabewerte .....	175
2.7.7.4	Methoden für die S7-Bausteindienste.....	176
2.7.8	Variabldienste .....	179
2.7.8.1	Variabldienste .....	179
2.7.8.2	Syntax der Variabldienste .....	180
2.7.8.3	Beispiele für Prozessvariablen für S7-OPC-UA-Variabldienste .....	185
2.7.9	Blockorientierte Dienste .....	186
2.7.9.1	Syntax der blockorientierten Dienste .....	187
2.7.9.2	Beispiele für Prozessvariablen für blockorientierte Dienste .....	189
2.7.10	Baustein-Informations-Objekte einer S7-Verbindung .....	191
2.7.10.1	Längeninformationen .....	191
2.7.10.2	Musterobjekte .....	192
2.7.10.3	Diagnose- und Konfigurations-Informationen .....	192
2.7.11	S7-OPC-UA-Template-Datenvariablen.....	199
2.7.12	Events, Conditions und Alarne .....	200
2.7.12.1	Welche Alarne gibt es? .....	200
2.7.12.2	Was sind Events? .....	200
2.7.12.3	Welche Events des S7-UA-Alarming-Servers gibt es? .....	201
2.7.12.4	Eventtyp hierarchie von S7-UA-Alarming-Server .....	201
2.7.13	Standard-Eventtypen .....	204
2.7.13.1	Standard-Eventtyp mit dem Anzeigenamen "BaseEventType" .....	204
2.7.13.2	Standard-Eventtyp mit dem Anzeigenamen "ConditionType" .....	210
2.7.13.3	Standard Eventtyp mit dem Anzeigenamen "AcknowledgeableConditionType" .....	213
2.7.13.4	Standard-Eventtyp mit dem Anzeigenamen "AlarmConditionType".....	213
2.7.13.5	Standard-Eventtyp mit dem Anzeigenamen "ExclusiveLimitAlarmType" .....	215
2.7.13.6	Standard-Eventtyp mit dem Anzeigenamen "ExclusiveLevelAlarmType" .....	215
2.7.13.7	Standard-Eventtyp mit dem Anzeigenamen "ExclusiveDeviationAlarmType".....	216
2.7.13.8	Standard-Eventtyp mit dem Anzeigenamen "OffNormalAlarmType" .....	216
2.7.14	S7-Eventtypen .....	217
2.7.14.1	S7-Eventtyp mit dem Anzeigenamen "S7StatepathAlarmType".....	217

2.7.14.2	S7-Eventtyp mit dem Anzeigenamen "S7ExclusiveLevelAlarmType" .....	218
2.7.14.3	S7-Eventtyp mit dem Anzeigenamen "S7ExclusiveDeviationAlarmType".....	220
2.7.14.4	S7-Eventtyp mit dem Anzeigenamen "S7OffNormalAlarmType" .....	221
2.7.14.5	S7-Eventtyp mit dem Anzeigenamen "S7DiagnosisEventType" .....	223
2.7.15	Bereichsbbaum und Herkunftsraum .....	224
2.7.16	Empfang von Events .....	225
2.7.17	Methoden von UA-Alarmen.....	226
2.8	Optimierte S7-Kommunikation mit OPC UA .....	227
2.8.1	Eigenschaften der optimierten S7-Kommunikation mit OPC UA.....	227
2.8.2	SIMATIC NET OPC-UA-Server für das optimierte S7-Protokoll.....	227
2.8.3	Wie wird der S7OPT-OPC-UA-Server adressiert? .....	229
2.8.4	Welche Namensräume bietet der S7OPT-OPC-UA-Server an? .....	231
2.8.5	Die Nodeld .....	233
2.8.6	Verbindungsobjekte .....	235
2.8.7	Verbindungsnamen .....	239
2.8.8	Aufbau und Funktionen des produktiven S7OPT-Verbindungsobjekts .....	240
2.8.9	Typ-Definition des S7OPT-Verbindungsobjekts .....	240
2.8.10	S7OPT-Verbindungs-Informationsobjekte .....	241
2.8.11	Beispiele für S7OPT-spezifische Informationsvariablen und Rückgabewerte .....	241
2.8.12	Variabldienste .....	242
2.8.12.1	Standardzugriff.....	242
2.8.12.2	Optimierter Zugriff .....	250
2.8.13	Baustein-Informations-Objekte einer S7-Verbindung .....	255
2.8.13.1	Längeninformationen .....	255
2.8.13.2	Musterobjekte.....	256
2.8.13.3	Diagnose- und Konfigurationsinformationen.....	256
2.8.14	S7OPT-OPC-UA-Template-Datenvariablen .....	258
2.8.15	OPC-UA-Events, -Conditions und -Alarne .....	259
2.8.15.1	Welche OPC-UA-Alarme gibt es? .....	259
2.8.15.2	Was sind OPC-UA-Events? .....	260
2.8.15.3	Welche S7OPT-Eventtypen werden vom S7OPT-OPC-UA-Server unterstützt? .....	260
2.8.15.4	Eventtyp hierarchie von S7OPT-OPC-UA-Server .....	261
2.8.16	Standard-Eventtypen und die Verwendung ihrer Properties .....	264
2.8.16.1	Standard-Eventtyp mit dem Anzeigenamen "BaseEventType" .....	264
2.8.16.2	Bildung von SourceName, Message und Severity .....	265
2.8.16.3	Standard-Eventtyp mit dem Anzeigenamen "ConditionType" .....	268
2.8.16.4	Bildung von ConditionName .....	269
2.8.16.5	Standard-Eventtyp mit dem Anzeigenamen "AcknowledgeableConditionType" .....	270
2.8.16.6	Standard-Eventtyp mit dem Anzeigenamen "AlarmConditionType" .....	270
2.8.16.7	Standard-Eventtyp mit dem Anzeigenamen "OffNormalAlarmType".....	271
2.8.17	S7OPT-Eventtypen .....	272
2.8.17.1	S7OPT-Eventtyp mit dem Anzeigenamen "S7OPTStatepathAlarmType".....	272
2.8.17.2	S7OPT-Eventtyp mit dem Anzeigenamen "S7OPTConsistencyAlarmType" .....	272
2.8.17.3	S7OPT-Eventtyp mit dem Anzeigenamen "S7OPTOffNormalAlarmType".....	273
2.8.17.4	S7OPT-Eventtyp mit dem Anzeigenamen "S7OPTInfoReportEventType" .....	276
2.8.18	Bereichsbbaum und Herkunftsraum .....	278
2.8.19	Empfang von Events .....	280
2.8.20	Methoden von UA-Alarmen.....	282
2.9	Offene Kommunikationsdienste (SEND/RECEIVE).....	282
2.9.1	Offene Kommunikationsdienste (SEND/RECEIVE) über Industrial Ethernet.....	282
2.9.1.1	Perfomanter SIMATIC NET OPC-Server für das SR-Protokoll .....	283

2.9.1.2	Protokoll-ID .....	285
2.9.1.3	Verbindungsnamen.....	285
2.9.1.4	Variablen Dienste .....	285
2.9.1.5	Blockorientierte Dienste .....	288
2.9.1.6	SEND/RECEIVE-spezifische Informationsvariablen .....	292
2.9.1.7	Syntax der systemspezifischen Informationsvariablen.....	293
2.9.2	Offene Kommunikationsdienste (SEND/RECEIVE) über PROFIBUS.....	294
2.9.2.1	Protokoll-ID .....	294
2.9.2.2	Verbindungsnamen.....	294
2.9.2.3	Blockorientierte Dienste .....	295
2.9.2.4	FDL-spezifische Informationsvariablen.....	300
2.9.2.5	Syntax der systemspezifischen Informationsvariablen.....	302
2.10	Offene Kommunikationsdienste (SEND/RECEIVE) mit OPC UA über Industrial Ethernet .....	303
2.10.1	Eigenschaften der SR-Kommunikation mit OPC UA .....	303
2.10.2	SIMATIC NET OPC-UA-Server für das SR-Protokoll.....	303
2.10.3	Wie wird der SR-OPC-UA-Server adressiert? .....	305
2.10.4	Protokoll-ID .....	307
2.10.5	Welche Namensräume bietet der SR-OPC-UA-Server an? .....	307
2.10.6	Verbindungsnamen.....	308
2.10.7	Die Nodeld .....	309
2.10.8	Datenvariablen für S5-Datenbausteine und Bereiche (S5-kompatible Kommunikation).....	311
2.10.9	Blockorientierte Dienste .....	314
2.10.10	SR-spezifische Informationsvariablen .....	318
2.10.11	SR-OPC-UA-Template-Datenvariablen .....	318
2.11	SNMP-Kommunikation über Industrial Ethernet.....	322
2.11.1	Protokoll-ID .....	322
2.11.2	Datentypen des SNMP-Protokolls .....	322
2.11.3	Prozessvariablen für SNMP-Variablen Dienste .....	323
2.11.4	SNMP-spezifische Informationsvariablen .....	324
2.11.5	SNMP-spezifische Traps .....	327
2.12	PROFINET-IO-Kommunikation über Industrial Ethernet .....	328
2.12.1	Performanter SIMATIC NET OPC-Server für das PROFINET-IO-Protokoll.....	328
2.12.2	Wie können IO-Daten adressiert werden?.....	330
2.12.3	Wie kann der projektierte PROFINET-IO-Namensraum durchsucht werden? .....	334
2.12.4	Welche OPC-Variablen für PROFINET IO stehen zur Verfügung? .....	335
2.12.5	Protokoll-ID .....	335
2.12.6	Controller-Name.....	335
2.12.7	PROFINET-IO-Prozessvariablen .....	335
2.12.8	PROFINET-IO-Datenstatus .....	340
2.12.9	PROFINET-IO-Datensätze .....	343
2.12.10	Syntax der systemspezifischen Informationsvariablen .....	347
2.12.11	PROFINET-IO-spezifische Informationsvariablen .....	347
2.13	PROFINET-IO-Kommunikation mit OPC UA über Industrial Ethernet .....	350
2.13.1	SIMATIC NET OPC-UA-Server für das PROFINET-IO-Protokoll.....	350
2.13.2	Wie wird der PROFINET-IO-OPC-UA-Server adressiert? .....	353
2.13.3	Welche Namensräume bietet die PROFINET-IO-Kommunikation mit OPC UA? .....	355
2.13.4	Die Nodeld .....	358
2.13.5	Board-Objekt und PROFINET-IO-Controller.....	359
2.13.6	Datenvariablen des PROFINET-IO-Controllers .....	362

2.13.7	Device-Objekte.....	362
2.13.8	Datenvariablen und Methoden des Device-Objekts .....	364
2.13.9	PROFINET-IO-Modulobjekte .....	365
2.13.9.1	Datenvariablen der PROFINET-IO-Module .....	367
2.13.9.2	IO-Datenvariablen der PROFINET-IO-Module .....	368
2.13.9.3	Datenvariablen für IOPS und IOCS .....	369
2.13.9.4	Datensatz-Datenvariablen der PROFINET-IO-Module.....	373
2.13.10	PROFINET-IO-OPC-UA-Templates.....	376
2.13.10.1	Template-Datenvariablen.....	376
2.13.10.2	Verzeichnis der Template-Datenvariablen.....	377
2.14	Server-Diagnose .....	379
2.14.1	Protokoll-ID .....	379
2.14.2	OPC-DA-Server-Diagnose-Items.....	379
2.15	Blockorientierte Dienste mit der OPC-Schnittstelle .....	384
2.15.1	Blockdienste verwenden .....	384
2.15.2	Eigenschaften blockorientierter Kommunikation.....	384
2.15.3	Abbildung von Datenpuffern auf OPC-Variablen .....	385
2.15.4	Anwendung der blockorientierten Dienste .....	386
2.15.5	Besonderheiten der blockorientierten Dienste über TCP/IP native .....	387
2.16	Zugriffsrechte von OPC-Variablen einschränken .....	387
<b>3</b>	<b>OPC Alarms &amp; Events-Server für SIMATIC NET .....</b>	<b>391</b>
3.1	Event-Server für S7-Kommunikation .....	391
3.1.1	Funktionsprinzip und Alarmkategorien .....	391
3.1.2	Parameter für Ereignisse .....	395
3.1.3	Ereignisattribute .....	397
3.1.4	Attribute für Einträge im Diagnosepuffer der Baugruppe.....	407
3.1.5	Attribute für Alarne, die eine unterbrochene Verbindung anzeigen.....	411
3.1.6	Projektierung von Meldetexten, Source und Area .....	413
3.1.7	Wie können Herkunftsangaben, Quellen und Bedingungen im Namensraum durchsucht und gefiltert werden?.....	419
3.1.8	Abbildung der STEP 7-Projektierungswerte auf OPC S7 Alarne & Events-Parameter.....	421
3.1.9	S7-Demoalarme .....	422
3.2	Simple Event-Server für SNMP-Kommunikation .....	424
3.3	Alarms & Events-Server für S7- und SNMP-Kommunikation .....	427
3.3.1	Die A&E-Server von SIMATIC NET .....	427
3.3.2	Wie sind die Namen (ProgID) der Server? .....	428
3.3.3	Wie kann das Protokoll der Alarmquelle erkannt werden? .....	428
<b>4</b>	<b>OPC-Server nutzen.....</b>	<b>431</b>
4.1	Automation-Schnittstelle programmieren.....	431
4.1.1	Automation-Schnittstelle programmieren für Data Access .....	431
4.1.1.1	Was leistet das Objektmodell von OPC Data Access?.....	431
4.1.1.2	Was müssen Sie bei der Programmierung beachten? .....	432
4.1.1.3	Objekte der Automation-Schnittstelle für Data Access .....	433
4.1.2	Automation-Schnittstelle programmieren für Alarms & Events.....	442
4.1.2.1	Was leistet das Objektmodell von OPC Alarms & Events? .....	442
4.1.2.2	Was müssen Sie bei der Programmierung beachten? .....	443
4.1.2.3	Objekte der Automation-Schnittstelle für Alarms & Events .....	443

4.2	Custom-Schnittstelle programmieren .....	452
4.2.1	COM-Objekt erzeugen und Status des OPC-Servers abfragen .....	453
4.2.2	Objekte der Custom-Schnittstelle für Data Access.....	454
4.2.2.1	Objekt OPCServer .....	454
4.2.2.2	Objekt OPCGroup.....	458
4.2.3	Objekte der Custom-Schnittstelle für Alarms & Events .....	466
4.2.3.1	Objekt OPCEventServer .....	466
4.2.3.2	Objekt OPCEventSubscription.....	469
4.2.3.3	Objekt OPCEventAreaBrowser.....	471
4.2.4	Schnittstellen des Client für Alarms und Events .....	471
4.2.4.1	Schnittstelle IOPCEventSink .....	472
4.2.4.2	Schnittstelle IOPCShutdown.....	473
4.2.5	Fehlermeldungen für OPC-DA-Prozessvariablen.....	474
4.3	XML-Schnittstelle programmieren .....	477
4.3.1	Beschreibung der Elemente .....	479
4.3.2	Basis-Schemas .....	480
4.3.2.1	ItemProperty .....	480
4.3.2.2	ItemValue .....	481
4.3.2.3	OPCError .....	482
4.3.2.4	ReplyBase.....	483
4.3.2.5	RequestOptions .....	483
4.3.3	Read- und Write-Aufträge .....	485
4.3.3.1	Read.....	485
4.3.3.2	ReadResponse .....	487
4.3.3.3	Write.....	489
4.3.3.4	WriteResponse .....	491
4.3.4	Verwendung von Subscriptions .....	492
4.3.4.1	Subscribe .....	494
4.3.4.2	SubscribeResponse.....	496
4.3.4.3	SubscriptionPolledRefresh .....	497
4.3.4.4	SubscriptionPolledRefreshResponse .....	498
4.3.4.5	SubscriptionCancel .....	499
4.3.4.6	SubscriptionCancelResponse.....	500
4.3.5	Weitere Abfragen .....	500
4.3.5.1	Browse .....	500
4.3.5.2	BrowseResponse.....	502
4.3.5.3	GetProperties .....	503
4.3.5.4	GetPropertiesResponse.....	505
4.3.5.5	GetStatus .....	506
4.3.5.6	GetStatusResponse .....	507
4.4	OPC-UA-Schnittstelle programmieren.....	508
4.4.1	OPC-UA-Server konfigurieren .....	509
4.4.1.1	Authentifizierung .....	509
4.4.1.2	Endpunkt-Sicherheit.....	510
4.4.2	Zertifikatsverwaltung für den OPC-UA-Server.....	511
4.4.3	Zertifikatsverwaltung im OPC-UA-Client "OPC Scout V10" .....	513
4.4.4	Zertifikatsverwaltung mit grafischer Oberfläche .....	518
4.4.5	OPC-UA-Dienste.....	520
4.4.6	OPC-UA-Clients erstellen .....	524
4.4.6.1	Schnittstellen unter OPC UA .....	524
4.4.6.2	Die C-Schnittstelle unter OPC UA .....	524

4.4.6.3	Die .NET-Schnittstelle unter OPC UA.....	525
4.4.7	Meldungen der OPC-UA-Server .....	526
4.4.8	Migration von OPC Data Access / Alarms & Events nach OPC UA.....	527
4.5	OPC-UA-Redundanz-Schnittstelle programmieren .....	528
4.5.1	Installieren und Konfigurieren des Feature "Netzwerklastenausgleich" .....	532
4.5.2	Projektierung der Redundanzgruppe .....	538
4.5.3	Auffinden der Server-Endpunkte mit Hilfe des Discovery-Servers .....	546
4.5.4	Der OPC-UA-Reconnect-Vorgang .....	547
4.5.5	Redundanz Unterstützung durch OPC-UA-Clients .....	548
4.5.6	Wann tritt eine Serverumschaltung ein? .....	549
4.5.7	Details zur Serverumschaltung .....	550
4.5.8	Verhalten des SIMATIC NET S7-OPC-UA-A&C bei Redundanzumschaltung .....	552
4.5.9	Generelle Empfehlungen und Hinweise für die redundante S7-OPC-UA-Kommunikation .....	554
4.5.10	Auslesen des Zustands der Redundanzgruppe vom OPC-UA-Client .....	555
<b>5</b>	<b>SIMATIC Computing .....</b>	<b>559</b>
5.1	.NET OPC Client Control .....	560
5.1.1	Übersicht .....	560
5.1.2	Schritt 1 - Installation .....	561
5.1.2.1	Referenzieren der Controls .....	561
5.1.2.2	Einstellung eines Beispielprogramms .....	564
5.1.3	Schritt 2 - Controls in einer WINDOWS FORM .....	566
5.1.3.1	Windows Controls hinzufügen .....	569
5.1.3.2	.NET OPC Client Control hinzufügen .....	571
5.1.4	Schritt 3 - Datenverknüpfungen konfigurieren .....	572
5.1.4.1	OPC-Items browsen .....	573
5.1.4.2	Windows Control Eigenschaften browsen .....	576
5.1.4.3	Liste der Verknüpfungen erzeugen .....	577
5.1.4.4	Trigger zum Schreiben .....	579
5.1.4.5	OPC Quality und Error .....	583
5.1.5	Schritt 4 - Starten des Beispielprogramms .....	586
5.1.5.1	Debug und Release .....	586
5.1.5.2	Applikation verteilen .....	586
5.1.6	Fehlerbehandlung .....	587
5.1.7	OPC-UA-Zertifikat-Erstellung für das NET OPC Data Control .....	588
5.1.8	OPC-UA-Zertifikate bei Verbindung zu einem OPC-UA-Server .....	589
5.2	.NET OPC Client API .....	590
5.2.1	Namensraum SIMATICNET.OPCDAClient .....	592
5.2.1.1	Klassen des Datenmodells .....	592
5.2.1.2	Die Schnittstelle des Objektes DaServerMgt .....	601
5.2.1.3	Fehlerbehandlung .....	619
5.2.2	Beispiel zu Namensraum SIMATICNET.OPCDAClient .....	620
5.2.3	Namensraum SIMATICNET.OPCCMN .....	623
5.2.3.1	Die Schnittstelle des Objektes OPCServerEnum .....	624
5.2.3.2	Klasse ServerIdentifier .....	626
5.2.3.3	Enumerator ServerCategory .....	626
5.2.3.4	Klasse EndpointIdentifier .....	627
5.2.3.5	Klasse PkiCertificate .....	627
5.2.3.6	Erzeugen eines neuen Zertifikats .....	627
5.2.3.7	Methode toDER .....	628

5.2.3.8	Methode fromDER .....	628
5.2.3.9	Methode toWindowsStore .....	629
5.2.3.10	Methode toWindowsStoreWithPrivateKey .....	629
5.2.3.11	Methode fromWindowsStore .....	630
5.2.3.12	Methode fromWindowsStoreWithPrivateKey .....	630
5.2.3.13	Methode fromWindowsStoreWithPrivateKey .....	631
5.2.3.14	Properties .....	631
5.2.3.15	Enumerator WinStoreLocation .....	632
<b>6</b>	<b>Beispielprogramme .....</b>	<b>633</b>
6.1	OPC-Automation-Schnittstelle (Synchrone Kommunikation) in VB.NET .....	633
6.1.1	Aktivieren der Simulationsverbindung .....	633
6.1.2	Bedienung des Beispielprogramms .....	634
6.1.3	Beschreibung des Programmablaufs .....	635
6.1.4	Nutzung der OPC-Automation-Schnittstelle mit dem .NET-Framework .....	636
6.2	OPC-Custom-Schnittstelle (Synchrone Kommunikation) in C++ .....	639
6.2.1	Aktivieren der Simulationsverbindung .....	639
6.2.2	Bedienung des Beispielprogramms .....	640
6.2.3	Beschreibung des Programmablaufs .....	641
6.2.4	Programmbeschreibung OPCDA_SyncDlg.cpp .....	645
6.2.4.1	OnInitDialog .....	645
6.2.4.2	OnStart .....	646
6.2.4.3	OnRead .....	654
6.2.4.4	OnWrite .....	656
6.2.4.5	OnStop .....	658
6.2.4.6	DestroyWindow .....	660
6.2.4.7	GetQualityText .....	660
6.2.5	Hinweise zum Erstellen eigener Programme .....	661
6.3	OPC-Custom-Schnittstelle (Asynchrone Kommunikation) in C++ .....	661
6.3.1	Aktivieren der Simulationsverbindung .....	662
6.3.2	Bedienung des Beispielprogramms .....	662
6.3.3	Programm starten .....	662
6.3.4	Werte lesen und schreiben .....	663
6.3.5	Gruppe aktivieren .....	663
6.3.6	Programm beenden .....	664
6.3.7	Beschreibung des Programmablaufs .....	664
6.3.8	Beschreibung der Programmstruktur .....	671
6.3.9	Die Datei "OPCDA_AsyncDlg.cpp" .....	671
6.3.10	Callback.cpp und Callback.h .....	693
6.3.11	Hinweise zum Erstellen eigener Programme .....	698
6.4	OPC-Custom-Schnittstelle (Asynchrone Kommunikation) in VB.NET .....	698
6.4.1	Bedienung des Beispielprogramms .....	699
6.4.2	Programmbeschreibung .....	700
6.5	OPC-Custom-Schnittstelle (Synchrone Kommunikation) in C# .....	705
6.5.1	Bedienung des Beispielprogramms .....	706
6.5.2	Programmbeschreibung .....	706
6.6	OPC-Custom-Schnittstelle (Asynchrone Kommunikation) in C# .....	710
6.6.1	Bedienung des Beispielprogramms .....	710
6.6.2	Programmbeschreibung .....	711

6.7	OPC-XML-Schnittstelle in C#.....	716
6.7.1	Bedienung des Beispielprogramms .....	716
6.7.2	Web-Dienst zum Projekt hinzufügen.....	718
6.7.3	Die Klasse MainForm.....	719
6.7.4	Der Konstruktor von MainForm und die Methode Dispose.....	721
6.7.5	Erzeugen der Dialogfeldelemente.....	721
6.7.6	Die Methode Main.....	726
6.7.7	Die Methode Button_Start_Sample_Click .....	727
6.7.8	Die Methode Button_Stop_Sample_Click .....	729
6.7.9	Die Methode Button_Read_Value_Click .....	730
6.7.10	Die Methode Button_Write_Value_Click .....	731
6.8	OPC Alarms & Events Custom-Schnittstelle in C++.....	733
6.9	OPC-UA-Schnittstelle in C .....	736
6.9.1	Aktivieren der Simulationsverbindung .....	737
6.9.2	Importieren des Client-Zertifikates .....	737
6.9.3	Bedienung des Beispielprogramms .....	738
6.9.4	Programm starten .....	738
6.9.5	Werte lesen und schreiben .....	738
6.9.6	Variablen beobachten .....	739
6.9.7	Programm beenden .....	739
6.9.8	Beschreibung des Programmablaufs.....	740
6.9.8.1	Verbindungsaufbau.....	740
6.9.8.2	Lesen und Schreiben von Variablen .....	741
6.9.8.3	Beobachten von Variablen und Alarmen .....	741
6.9.8.4	Verbindungsabbau .....	742
6.9.9	Hinweise zum Umstellen auf reale Variablen .....	742
6.10	OPC-UA-Schnittstelle (Asynchrone Kommunikation) in C# .....	743
7	<b>Referenz Automation-Schnittstelle .....</b>	<b>745</b>
7.1	Allgemeine Informationen .....	745
7.1.1	Was ist eine Schnittstelle? .....	745
7.1.2	Die zwei Schnittstellenarten von OPC .....	746
7.1.3	COM-/OLE-Objekte.....	746
7.1.4	Collection-Objekte.....	747
7.1.5	Objektmodell .....	748
7.1.6	Datensynchronisation .....	749
7.1.7	Ausnahmen .....	749
7.1.8	Ereignisse .....	750
7.1.9	Arrays .....	750
7.1.10	Parameter .....	750
7.1.11	Type Library .....	750
7.2	Das Objekt OPCServer.....	751
7.2.1	Eigenschaften des Objekts OPCServer.....	751
7.2.1.1	StartTime.....	752
7.2.1.2	CurrentTime .....	752
7.2.1.3	LastUpdateTime .....	753
7.2.1.4	MajorVersion .....	753
7.2.1.5	MinorVersion .....	754
7.2.1.6	BuildNumber .....	754
7.2.1.7	VendorInfo.....	755

7.2.1.8	ServerState .....	755
7.2.1.9	LocaleID .....	756
7.2.1.10	Bandwidth .....	757
7.2.1.11	OPCGroups .....	757
7.2.1.12	PublicGroupNames .....	758
7.2.1.13	ServerName .....	758
7.2.1.14	ServerNode .....	759
7.2.1.15	ClientName .....	759
7.2.2	Methoden des Objekts OPCServer .....	760
7.2.2.1	GetOPCServers .....	760
7.2.2.2	Connect .....	761
7.2.2.3	Disconnect .....	762
7.2.2.4	CreateBrowser .....	763
7.2.2.5	GetErrorString .....	763
7.2.2.6	QueryAvailableLocaleIDs .....	764
7.2.2.7	QueryAvailableProperties .....	764
7.2.2.8	GetItemProperties .....	765
7.2.2.9	LookupItemIDs .....	766
7.2.3	Ereignisse des Objekts OPCServer .....	767
7.2.3.1	ServerShutDown .....	768
7.3	Das Collection-Objekt OPCGroups .....	768
7.3.1	Eigenschaften des Objekts OPCGroups .....	769
7.3.1.1	Parent .....	769
7.3.1.2	DefaultGroupsActive .....	770
7.3.1.3	DefaultGroupUpdateRate .....	770
7.3.1.4	DefaultGroupDeadband .....	771
7.3.1.5	DefaultGroupLocaleID .....	771
7.3.1.6	DefaultGroupTimeBias .....	772
7.3.1.7	Count .....	772
7.3.2	Methoden des Objekts OPCGroups .....	773
7.3.2.1	Item .....	773
7.3.2.2	Add .....	774
7.3.2.3	GetOPCGroup .....	774
7.3.2.4	Remove .....	775
7.3.2.5	RemoveAll .....	775
7.3.2.6	ConnectPublicGroup .....	776
7.3.2.7	RemovePublicGroup .....	776
7.3.3	Ereignisse des Objekts OPCGroups .....	777
7.3.3.1	GlobalDataChange .....	777
7.4	Das Objekt OPCGroup .....	779
7.4.1	Eigenschaften des Objekts OPCGroup .....	780
7.4.1.1	Parent .....	780
7.4.1.2	Name .....	780
7.4.1.3	IsPublic .....	781
7.4.1.4	IsActive .....	782
7.4.1.5	IsSubscribed .....	782
7.4.1.6	ClientHandle .....	783
7.4.1.7	ServerHandle .....	784
7.4.1.8	LocaleID .....	785
7.4.1.9	TimeBias .....	785
7.4.1.10	DeadBand .....	786

7.4.1.11	UpdateRate .....	787
7.4.1.12	OPCItems.....	787
7.4.2	Methoden des Objekts OPCGroup .....	788
7.4.2.1	SyncRead.....	788
7.4.2.2	SyncWrite.....	790
7.4.2.3	AsyncRead.....	791
7.4.2.4	AsyncWrite .....	793
7.4.2.5	AsyncRefresh.....	794
7.4.2.6	AsyncCancel .....	795
7.4.3	Ereignisse des Objekts OPCGroup .....	796
7.4.3.1	DataChange .....	796
7.4.3.2	AsyncReadComplete .....	797
7.4.3.3	AsyncWriteComplete .....	798
7.4.3.4	AsyncCancelComplete.....	799
7.5	Das Collection-Objekt OPCItems.....	800
7.5.1	Eigenschaften des Collection-Objekts OPCItems .....	800
7.5.1.1	Parent.....	801
7.5.1.2	DefaultRequestedDataType.....	801
7.5.1.3	DefaultAccessPath.....	802
7.5.1.4	DefaultIsActive .....	802
7.5.1.5	Count.....	803
7.5.2	Methoden des Collection-Objekts OPCItems .....	803
7.5.2.1	Item .....	803
7.5.2.2	GetOPCItem.....	804
7.5.2.3	AddItem .....	804
7.5.2.4	AddItems .....	805
7.5.2.5	Remove .....	806
7.5.2.6	Validate .....	807
7.5.2.7	SetActive .....	808
7.5.2.8	SetClientHandles .....	809
7.5.2.9	SetDataTypes .....	810
7.6	Das Objekt OPCItem .....	810
7.6.1	Eigenschaften des Objekts OPCItem .....	811
7.6.1.1	Parent.....	811
7.6.1.2	ClientHandle.....	811
7.6.1.3	ServerHandle .....	812
7.6.1.4	AccessPath .....	812
7.6.1.5	AccessRights .....	813
7.6.1.6	ItemID.....	813
7.6.1.7	IsActive .....	814
7.6.1.8	RequestedDataType .....	814
7.6.1.9	Value .....	815
7.6.1.10	Quality .....	816
7.6.1.11	TimeStamp.....	816
7.6.1.12	CanonicalDataType .....	817
7.6.1.13	EUType .....	817
7.6.1.14	EUIInfo .....	818
7.6.2	Methoden des Objekts OPCItem .....	819
7.6.2.1	Read.....	819
7.6.2.2	Write .....	820
7.7	Definitionen .....	820

7.7.1	Zustand des Servers.....	821
7.7.2	Fehlermeldungen .....	821
7.8	Anhang zur Referenz Automation-Schnittstelle.....	823
<b>8</b>	<b>Literaturhinweise.....</b>	<b>827</b>
8.1	OPC-Spezifikationen.....	828



# Vorwort

## Was ist neu beim SIMATIC NET OPC-Server V13?

In der OPC Unified Architecture Version 1.01 sind die bisherigen Funktionalitäten erweitert worden.

Die SIMATIC NET OPC-UA-Server unterstützen zusätzlich:

- S7OPT-UA-Alarming

Was ist neu in diesem Handbuch?

- Die Beschreibung für S7OPT-UA-Alarming wurde im Kapitel "Optimierte S7-Kommunikation mit OPC UA" hinzugefügt.

## SIMATIC NET - Wegweisende Erfolgskonzepte schwarz auf weiß

Diese Dokumentation begleitet Sie auf Ihrem Weg zum erfolgreichen Einsatz von SIMATIC NET. Sie führt anschaulich in das Thema ein und zeigt Ihnen, wie Sie einzelne Komponenten installieren und projektieren, und wie Sie Programme auf der Basis von OPC erstellen. Sie werden sehen, was industrielle Kommunikation mit SIMATIC NET bedeuten kann - für Ihre Automatisierungswelt und vor allem für den Erfolg Ihres Unternehmens.

## SIMATIC NET - Eine gute Entscheidung

Sie kennen die Vorteile verteilter Automatisierungssysteme und wollen die Möglichkeiten industrieller Kommunikation optimal nutzen. Sie bauen auf einen starken Partner; Sie setzen auf innovative und zuverlässige Produkte. Mit SIMATIC NET haben Sie die richtige Wahl getroffen.

Diese Dokumentation baut auf Ihrem Wissen auf und lässt Sie vom Know-how der Spezialisten profitieren.

## Sind Sie Neuling?

Dann können Sie sich systematisch einarbeiten. Beginnen Sie in Band 1 mit der Einführung in die industrielle Kommunikation. In Band 1 erfahren Sie alles Nötige über Kommunikationsprinzip und Funktionsumfang des SIMATIC NET OPC-Servers. Lesen Sie die Grundlagen zur OPC-Schnittstelle, machen Sie sich mit den SIMATIC NET-Kommunikationsprotokollen und deren Vorteilen und Funktionen bekannt.

## Sind Sie Profi?

Dann können Sie sofort durchstarten. Dieser Band 2 gibt Ihnen alle Informationen, die Sie zum Bedienen von SIMATIC NET benötigen.

## *1.1 Wie arbeite ich mit der Dokumentation?*

### **Folgen Sie gerne einem guten Beispiel?**

Dann finden Sie in den mitgelieferten Beispielprogrammen wertvolle Anregungen, die Ihnen helfen, eigene Vorstellungen umzusetzen.

### **SIMATIC NET Glossar**

Erklärungen zu den Fachbegriffen, die in dieser Dokumentation vorkommen, sind im SIMATIC NET-Glossar enthalten.

Sie finden das SIMATIC NET-Glossar hier:

- SIMATIC NET Manual Collection

Die DVD liegt einigen SIMATIC NET-Produkten bei.

- Im Internet unter folgender Beitrags-ID:

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

### **Security-Hinweise**

---

#### **Hinweis**

Siemens bietet für sein Automatisierungs- und Antriebsproduktportfolio Security-Mechanismen, um einen sicheren Betrieb der Anlage/Maschine zu unterstützen. Unsere Produkte werden auch unter dem Gesichtspunkt Industrial Security ständig weiterentwickelt. Wir empfehlen Ihnen daher, dass Sie sich regelmäßig über Aktualisierungen und Updates unserer Produkte informieren und nur die jeweils aktuellen Versionen bei sich einsetzen. Informationen dazu finden Sie unter:  
(<http://support.automation.siemens.com>)

Hier können Sie sich für einen produktspezifischen Newsletter registrieren.

Für den sicheren Betrieb einer Anlage/Maschine ist es darüber hinaus notwendig, die Automatisierungskomponenten in ein ganzheitliches Industrial Security-Konzept der gesamten Anlage/Maschine zu integrieren, das dem aktuellen Stand der Technik entspricht. Hinweise hierzu finden Sie unter:  
(<http://www.siemens.com/industrialsecurity>)

Dabei sind auch eingesetzte Produkte von anderen Herstellern zu berücksichtigen.

---

## **1.1 Wie arbeite ich mit der Dokumentation?**

Um Ihnen das Arbeiten mit der Dokumentation zu erleichtern, werden verschiedene Darstellungsmittel und Bedienelemente verwendet. Sie dienen insbesondere dazu, Informationsaufnahme und Informationszugriff zu beschleunigen.

## Darstellungsmittel

Diese Dokumentation setzt unterschiedliche Darstellungsmittel ein, damit Sie unterschiedliche Informationen besser differenzieren können und spezielle Information leichter erkennen. Die Tabelle zeigt, welche Darstellungsmittel verwendet werden.

Dieses Darstellungsmittel	zeigt an:
<i>kursive</i> Schrift	Dies sind <i>betonte</i> Ausdrücke oder <i>Namen</i> von Items, Variablen, Programmen, Dialogen etc.
<Text in spitzen Klammern>	Dies sind <variable Begriffe>, die Sie durch den jeweils aktuellen Begriff ersetzen müssen.
<b>fett gedruckte Begriffe</b> am Anfang der Textspalte	<b>Syntaxeelemente</b> , die nachfolgend beschrieben werden.
Text in Schriftart Courier New	Hier steht Programmcode.
{Syntax-Text in geschweiften Klammern}	Optionale Angaben in Syntaxbeschreibungen
"Syntax-Text in Anführungszeichen"	Variable Begriffe in Syntaxbeschreibungen

## 1.2 Bestimmungen

Bei den Produkten von SIMATIC NET müssen Sie folgende Bestimmungen beachten:

### 1.2.1 Welche juristischen Bestimmungen müssen Sie beachten?

Wir weisen darauf hin, dass der Inhalt dieses Dokuments nicht Teil einer früheren oder bestehenden Vereinbarung, Zusage oder eines Rechtsverhältnisses ist oder diese abändern soll. Sämtliche Verpflichtungen von Siemens ergeben sich aus dem jeweiligen Kaufvertrag, der auch die vollständige und allein gültige Gewährleistungsregel enthält. Diese vertraglichen Gewährleistungsbestimmungen werden durch die Ausführungen dieses Dokuments weder erweitert noch beschränkt.

Wir weisen außerdem darauf hin, dass aus Gründen der Übersichtlichkeit nicht jede nur erdenkliche Problemstellung im Zusammenhang mit dem Einsatz des OPC-Servers beschrieben werden kann. Sollten Sie weitere Informationen benötigen oder sollten besondere Probleme auftreten, die hier nicht ausführlich genug behandelt werden, können Sie die erforderliche Auskunft über die örtliche Siemens-Niederlassung anfordern.

### Haftungsausschluss

Wir haben den Inhalt dieses Dokuments auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so dass wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in diesem Dokument werden regelmäßig überprüft, und notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten. Für Verbesserungsvorschläge sind wir dankbar.

Technische Änderungen vorbehalten.

Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts ist nicht gestattet, soweit nicht ausdrücklich zugestanden. Zu widerhandlungen verpflichten

zu Schadenersatz. Alle Rechte vorbehalten, insbesondere für den Fall der Patenterteilung oder GM-Eintragung.

Siemens AG  
I IA SC CI  
Postfach 4848  
D-90026 Nürnberg

### 1.2.2 Welche Sicherheitsbestimmungen müssen Sie kennen?

#### Qualifiziertes Personal

Inbetriebsetzung und Betrieb dieses Produkts darf nur von qualifiziertem Personal vorgenommen werden. Qualifiziertes Personal im Sinne der sicherheitstechnischen Hinweise dieser Dokumentation sind Personen, die die Berechtigung haben, Geräte, Systeme und Stromkreise gemäß den Standards der Sicherheitstechnik in Betrieb zu nehmen, zu erden und zu kennzeichnen.

#### Bestimmungsgemäßer Gebrauch

##### **WARNUNG**

Das Gerät darf nur für die im entsprechenden Katalog und in der technischen Beschreibung vorgesehenen Einsatzfälle und nur in Verbindung mit von Siemens empfohlenen bzw. zugelassenen Fremdgeräten und -komponenten verwendet werden.

Der einwandfreie und sichere Betrieb des Produkts setzt sachgemäßen Transport, sachgemäße Lagerung, Aufstellung und Montage sowie sorgfältige Bedienung und Instandhaltung voraus.

Mit den in diesem Dokument beschriebenen Produkten ist es auf einfache Weise möglich, auf Prozessdaten zuzugreifen und diese zu ändern. Durch das Ändern von Prozessdaten können unvorhersehbare Reaktionen im Prozess ausgelöst werden, die zu Tod, schwerer Körperverletzung und/oder Sachschaden führen können.

Gehen Sie deshalb vorsichtig vor und achten Sie darauf, dass Sie nicht auf Daten zugreifen, die unerwartete Reaktionen in den gesteuerten Geräten hervorrufen könnten.

# OPC-Prozessvariablen für SIMATIC NET

Dieses Kapitel zeigt die Syntax der Namen von Prozessvariablen. Diese Namen legen an der OPC-Schnittstelle fest, welche Prozesswerte angesprochen werden. Die Variablennamen müssen Sie bei der Programmierung oder Konfiguration eines OPC-Client angeben.

## 2.1

### Welche Kommunikationsfunktionen gibt es?

Der OPC-Server bietet einen standardisierten Zugriff auf die industriellen Kommunikationsnetze von SIMATIC NET.

Der SIMATIC NET OPC-Server unterstützt die Anbindung von Anwendungen an beliebige Automatisierungskomponenten, die über PROFIBUS oder Industrial Ethernet vernetzt sind. Er bietet folgende Kommunikationsfunktionen:

- S7-Kommunikation
- Offene Kommunikationsdienste (SEND/RECEIVE)
- PROFIBUS DP und FDL
- SNMP
- PROFINET IO

Die Kommunikationsfunktionen sind speziell für die verschiedenen Anforderungen optimiert. Es können mehrere Kommunikationsfunktionen gleichzeitig vom OPC-Server für Data Access und Unified Architecture unterstützt werden. S7-Kommunikation, PROFINET IO-Kommunikation, SEND/RECEIVE-Kommunikation und DP-Kommunikation gibt es auch für OPC UA.

## 2.2

### Was sind Prozessvariablen?

Eine Prozessvariable ist ein schreibbares und/oder lesbares Datum der Prozessperipherie, wie beispielsweise die Temperatur eines Kessels als Eingangswert einer speicherprogrammierbaren Steuerung.

#### OPC COM (Data Access)

Prozessvariablen werden im OPC Data Access-Klassenmodell durch die Klasse OPC-Item vertreten. Nur die Elemente dieser Klasse repräsentieren in OPC eine reale Größe aus dem Prozess.

## **ItemID**

Die ItemID ist eine Zeichenfolge, die eine Prozessvariable eindeutig identifiziert. Sie gibt dem Server an, welche Prozessvariablen dem OPC-Item zugeordnet werden. Über das OPC-Item kann dann auf den Prozesswert zugegriffen werden.

Der OPC-Server von SIMATIC NET bildet die verschiedenen Kommunikationsdienste der SIMATIC NET Kommunikationsprotokolle über OPC-Items ab, indem er Teile der ItemID als Parameter für den Aufruf einer Kommunikationsfunktion verwendet.

## **OPC UA**

Alle Elemente eines Prozesses, werden im OPC-UA-Objektmodell durch geeignete Objekte vertreten, z.B. Prozessvariablen durch Datenvariablen oder Properties, Alarne durch Alarminstanzen oder Kommunikationsdienste durch Methoden. Die Objekte können untereinander in Beziehung stehen.

## **Nodeld**

Die Nodeld setzt sich aus einem Namensraum-Index und einem Bezeichner (Zeichenfolge, numerischer Wert, Byte-String oder GUID) zusammen. Die Nodeld identifiziert in einem OPC-UA-Server ein Objekt aus dem Prozess.

Die OPC-UA-Server von SIMATIC NET bilden die verschiedenen Kommunikationsdienste der Protokolle über den Zugriff auf Objekte ab.

## **2.3**

## **Wie werden die ItemIDs der Prozessvariablen gebildet?**

Prozessvariablen werden an der OPC-Schnittstelle durch einen eindeutigen Namen, die ItemID, identifiziert. Die ItemID setzt sich wie folgt zusammen:

## **Syntax**

<Protokoll-ID>: [<Verbindungsname>] <Variablenname>

## **Erklärungen**

### **<Protokoll-ID>**

gibt das Protokoll für den Zugriff auf die Prozessvariable an.

Es gibt folgende Protokoll-IDs:

DP	DP-Protokoll einschließlich DP Master, DP Slave und DPC1
S7	S7-Funktionen über PROFIBUS und Industrial Ethernet
SR	Offene Kommunikationsdienste über Industrial Ethernet
FDL	Offene Kommunikationsdienste über PROFIBUS
SNMP	SNMP-Kommunikation über Industrial Ethernet
PNIO	PROFINET-IO-Kommunikation über Industrial Ethernet
DP2	DP Master Klasse 2 und DPC2

**<Verbindungsname>**

Der Verbindungsname identifiziert die Verbindung oder die Kommunikationsbaugruppe, über die der Partner (z.B. SPS, andere PC-Station oder DP-Slave) erreicht werden kann. Er wird in der Projektierung mit STEP 7 / SIMATIC NCM PC bei der Netzwerkkonfiguration angegeben.

Die Angabe des Verbindungsnamens ist abhängig vom gewählten Protokoll. Innerhalb eines Protokolls muss der Verbindungsname eindeutig sein.

---

**Hinweis****Zugelassene Zeichen für Verbindungsnamen**

Folgende Zeichen sind im Zeichenvorrat bei Verbindungsnamen zugelassen:

A-Z, a-z, 0-9, \_, -, ^, !, #, \$, %, &, ', (, ), <, >, =, ?, ~, +, \*, ', :, @, {, }, " und das Leerzeichen (Leerzeichen nicht am Anfang oder Ende des Verbindungsnamens)

Folgende Zeichen sind im Zeichenvorrat bei Verbindungsnamen nicht zugelassen:

- Punkt ". "
- Pipesymbol "|"
- Backslash "\"
- Slash "/"
- Eckige Klammern "[" und "]"

Ausgeschlossen ist außerdem das Leerzeichen am Anfang und Ende des Verbindungsnamens.

---

**<Variablenname>**

Variable, die angesprochen werden soll.

Der Variablenname muss für die im Verbindungsnamen angegebene Verbindung eindeutig sein. Der Aufbau des Variablenamens ist abhängig vom gewählten Protokoll.

---

**Hinweis**

Die ItemIDs unterscheiden nicht zwischen Groß- und Kleinschreibung.

---

**Hinweis**

Die Syntax von OPC-UA-Nodelds finden Sie im Kapitel "OPC-UA-Schnittstelle programmieren (Seite 508)".

---

**Hinweis**

Geschweifte Klammern kennzeichnen optionale Syntax-Elemente.

---

## **2.4 PROFIBUS-DP**

### **Prozessvariablen für den DP-Master**

Der OPC-Server von SIMATIC NET für den DP-Master-Betrieb bietet für folgende Dienste Prozessvariablen an:

- Dienste für den Master Klasse 1  
Zugriff und Beobachtung von DP-Eingängen und Ausgängen
- DPC1-Dienste  
Azyklische Übertragung von Datenblöcken
- Fast Logic für
  - CP 5613/ CP 5613 A3 und CP 5614/ CP 5614 A3 (nur DP-Master)  
Automatische Überwachung von Slave-Daten
  - CP 5623 und CP5624 (nur DP-Master)  
Automatische Überwachung von Slave-Daten
- Diagnosevariablen  
Auswertung der statischen Diagnose

### **Prozessvariablen für den DP Master Klasse 2**

- Konfigurationsdienste
  - Lesen der Ein-/Ausgangsdaten eines Slave
  - Lesen der Slave-Konfiguration
  - Datensätze für einen Slave schreiben
- Online-Diagnose
  - Lesen der Diagnosedaten eines Slave
  - Lesen der Master Klasse 1 Diagnosedaten
- Datensätze für einen Slave schreiben und lesen (DPC2)

### **Prozessvariablen für den DP-Slave**

Der OPC-Server von SIMATIC NET für den DP-Slave-Betrieb bietet für folgende Dienste Prozessvariablen an:

- Variabldienste zum Zugriff auf lokale Slave-Daten  
Zugriff auf die Ein- und Ausgänge des Slaves
- Diagnosevariablen  
Auswertung der statischen Diagnose des Slaves

## 2.4.1 DP Master Klasse 1

### 2.4.1.1 Hochperformanter SIMATIC NET In-Process-Server für das PROFIBUS DP-Protokoll

Das PROFIBUS DP-Protokoll enthält ein Abbild der Ein- und Ausgangsdaten im Kommunikationsprozessor des Rechners. Die Zugriffe auf diese Prozessdaten erfolgen innerhalb des lokalen Rechners und können deshalb sehr schnell erfolgen, besonders bei Verwendung der SIMATIC NET Baugruppen CP 5613/CP 5614 und CP 5623/ CP 5624 über die Schnittstelle "DP-Base".

In einigen Anwendungsfällen, z.B. bei Verwendung von PC-basierten Steuerungen, sind extrem kurze Zeiten zum Zugriff auf Prozessdaten erforderlich. Für das sehr schnelle DP-Protokoll bietet SIMATIC NET einen In-Process-Server, der die Leistungsfähigkeit des DP-Protokolls auch OPC-Client nahezu ungeschmälert bereitstellt.

Der Verwendung von OPC als COM-basierte Client-Server-Architektur sind je nach Ausprägung des OPC-Servers gewisse interne Laufzeiten vorgegeben. Diese entstehen vor allem bei Verwendung eines Local Servers (auch "Out-Process-Server" genannt; ".exe"-Datei mit eigenem Prozessraum) durch Prozesswechsel und Übermittlung der Funktionsparameter vom Client zum Server (sog. Marshalling) und zurück.

Bei Ausprägung des OPC-Servers als In-Process-Server fallen die Laufzeiten für Prozesswechsel und Marshalling weg, da der OPC-Server als dynamisch ladbare Bibliothek (DLL) implementiert ist und im Prozess des Client abläuft.

Bei der Verwendung eines In-Process-Servers muss allerdings folgendes berücksichtigt werden:

- Nur 1 Client kann den Server zu einer Zeit benutzen.

Die gleichzeitige Benutzung des In-Process-OPC-Servers durch mehrere Clients würde eine mehrfache Erzeugung des Servers in verschiedenen Prozessräumen bewirken, die jedoch alle gleichzeitig und nicht koordiniert auf die gleiche Hardware zugreifen würden. Die Folge ist, dass nur der zuerst gestartete Client-Zugriff auf die Prozessdaten hat, der Zugriff weiterer Clients jedoch abgewiesen würde.

- Die Stabilität des OPC-Servers ist vom Client abhängig.

Verhält sich der OPC-Client unkontrolliert z.B. bei Zugriffsverletzungen, ist unweigerlich der OPC-Server mit betroffen. Die Folge ist, dass das ggf. notwendige Rücksetzen der Kommunikationsbaugruppe durch den OPC-Server nicht mehr erfolgen kann. Auch ein explizites Beenden des OPC-Servers über die Konfigurationsprogramme ist nicht möglich.

### Aufruf des In-Process-Server für DP

Der In-Process-Server für DP wird über eine separate ProgID angesprochen. Die ProgID lautet "OPC.SimaticNET.DP".

Die ProgID wird in Funktionsaufrufen oder in OPC-Clients angegeben, um einen Server auszuwählen.

**Beispiele für Funktionsaufrufe:**

**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 Performanter SIMATIC NET OPC-Server für das PROFIBUS DP-Protokoll**

##### **Performanceverbesserung auch bei mehreren Clients**

Wie im vorangegangenen Abschnitt beschrieben, kann der hochperformante Inproc-Server für PROFIBUS-DP nur von einem Client genutzt werden. Um bei höheren Performanceanforderungen die gleichzeitige Nutzung von zwei oder mehr Clients zu ermöglichen, wird eine weitere Konfigurationsvariante angeboten.

Hierfür werden alle unterlagerten DP-Protokollbibliotheken und der COM-Server als Inproc-Server in den Outproc-OPC-Server geladen. Die Protokollbearbeitung läuft im Prozess des OPC-Servers ab, weitere Laufzeiten für Prozesswechsel und Multiprotokollbetrieb fallen weg. Der Prozesswechsel zwischen OPC-Client und OPC-Server ist allerdings noch vorhanden.

## Konfiguration

Die Aktivierung dieser performanten Variante erfolgt implizit dadurch, dass im Konfigurationsprogramm "Kommunikations-Einstellungen" das Protokoll "DP" als einziges Protokoll ausgewählt wird (bei Auswahl weiterer Protokolle oder der OPC-UA-Schnittstelle entfällt der beschriebene Performance-Vorteil):

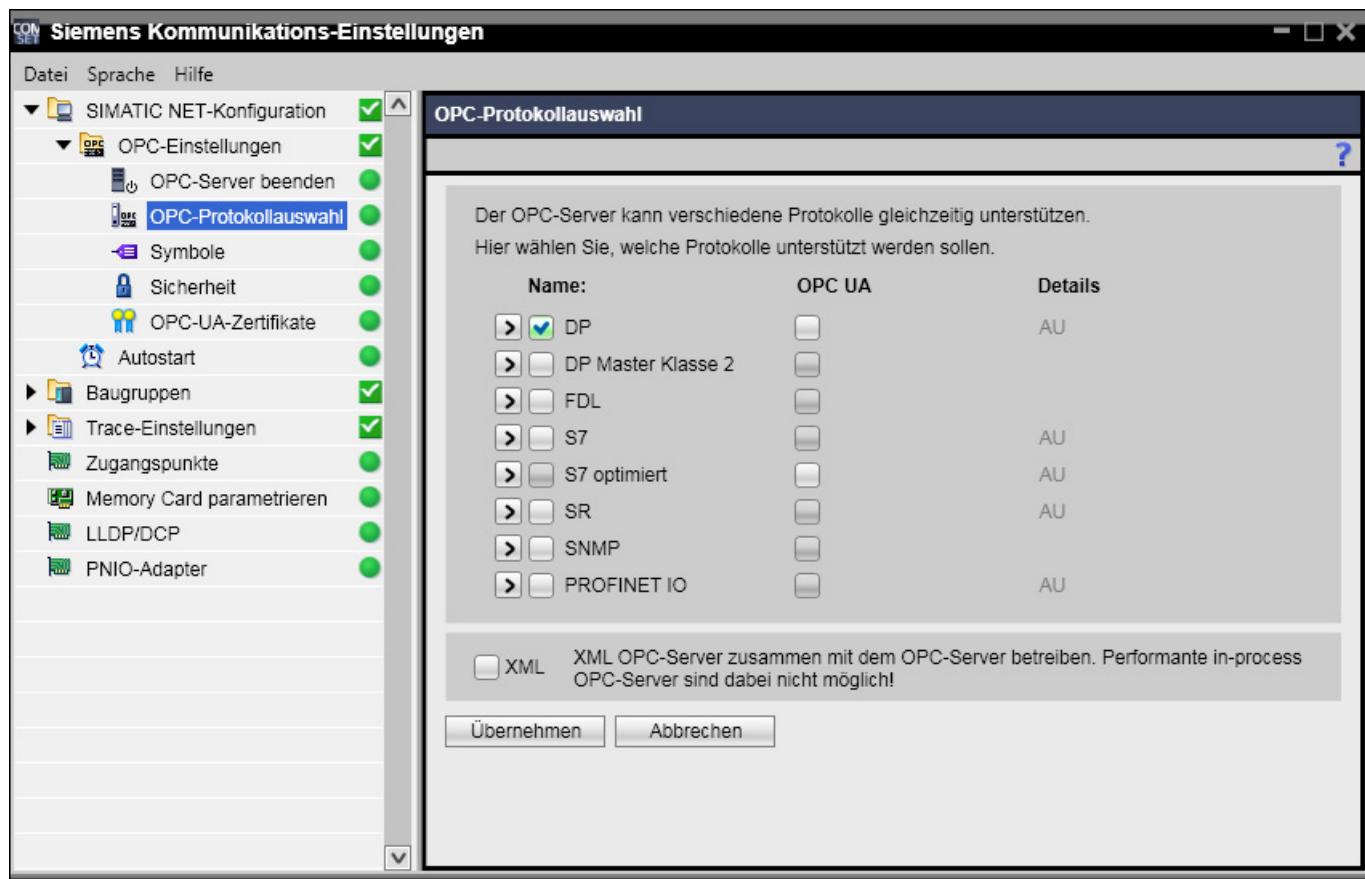


Bild 2-1 Fenster des Konfigurationsprogramms " Kommunikations-Einstellungen " zur Auswahl des DP-Protokolls

"Symbolik" darf zusätzlich ausgewählt werden.

### Hinweis

Für die Erstellung von Symbolen mit Symbol-Editor ist die Verwendung folgender Zeichen erlaubt: A-Z, a-z, 0-9, \_, -, ^, !, #, \$, %, &, ', /, (, ), <, >, =, ?, ~, +, \*, '!', :, |, @, [, ], {, }, ". Zusätzlich sollten Sie bei der Erstellung von Symbolen mit STEP 7 darauf achten, dass es bei der Array-Auflösung zu Problemen kommen kann, wenn Ihre Symboldatei gleichzeitig Symbole der Form <Symbolname> und <Symbolname>[<Index>] enthält.

## Vorteile / Nachteile

Die Verwendung des performanten DP-OPC-Servers hat den Nachteil, dass nur der Einzelprotokollbetrieb (DP) möglich ist. Dem stehen folgende Vorteile gegenüber:

- Höhere Performance als beim Multiprotokollbetrieb.
- Einfache Konfiguration.
- Mehrere Clients können den Server zur gleichen Zeit nutzen.
- Die Stabilität des OPC-Servers ist nicht von den Clients abhängig.

### 2.4.1.3 Einordnung der DPC1- und DPC2-Dienste

#### DP-Master Klasse 1

Der DP-Master Klasse 1 führt die zyklische Kommunikation zu den DP-Slaves aus. Durch DPC1 kann ein zyklisch arbeitender Master zusätzlichen, azyklischen Datenverkehr (Datensatz lesen, Datensatz schreiben) durchführen und auf Alarne von Slaves reagieren. Voraussetzung für DPC1 ist, dass die DP-Slaves diese DP-V1-Zusatzfunktionen unterstützen.

Die Kommunikation enthält zentrale Funktionen wie:

- Parametrierung und Konfigurierung der Slaves
- zyklischer Datentransfer zu den DP-Slaves
- Überwachen der DP-Slaves
- Bereitstellen von Diagnoseinformationen

#### DP-Master Klasse 2

Durch DPC2 kann ein DP-Master Klasse 2 weitere zusätzliche Kommunikationsbeziehungen zu DP-Slaves aufbauen. Voraussetzung ist, dass die DP-Slaves die DP-V1-Zusatzfunktionen unterstützen. Die wesentlichen DPC2-Funktionszusätze sind:

- Verbindungsaufbau
- Verbindungsabbau
- Datensatz lesen
- Datensatz schreiben

#### Slaves mit DP-V1-Zusatzfunktionen

Slaves mit DP-V1-Zusatzfunktionen können mit DP-Master Klasse 1 und DP-Master Klasse 2 kommunizieren.

## Übersicht DP-Protokoll

In der folgenden Abbildung sehen Sie die Protokollteile von DP bzw. DP-V1:

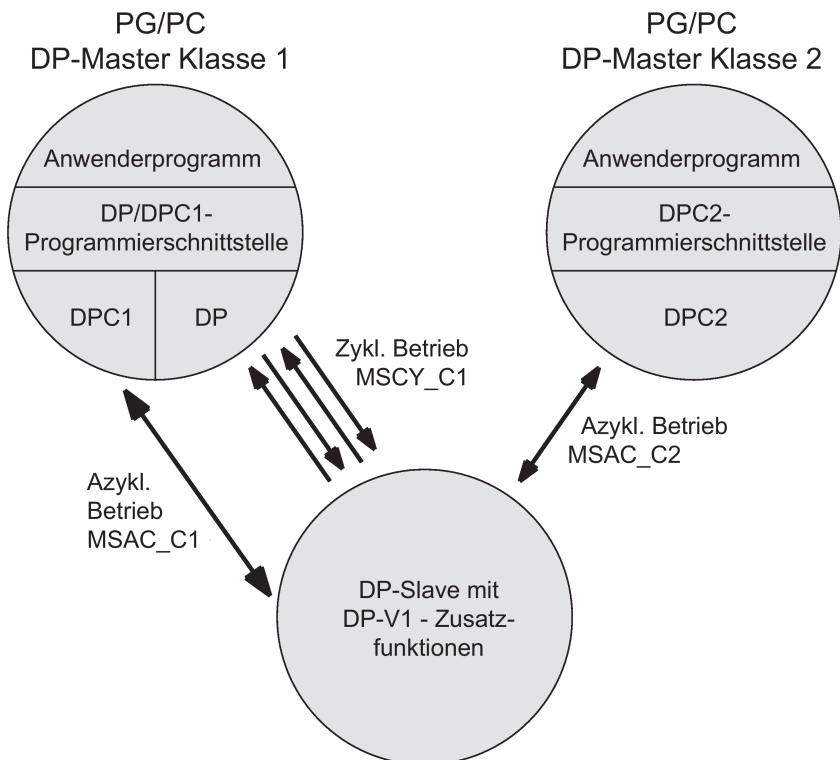


Bild 2-2 Die Protokollteile von DP bzw. DP-V1

MSCY_C1	Master-Slave, zyklischer C1-Betrieb
MSAC_C1	Master-Slave, azyklischer C1-Betrieb
MSAC_C2	Master-Slave, azyklischer C2-Betrieb

### 2.4.1.4 Prozessvariablen für Dienste des Master Klasse 1

Mit den Diensten zum Zugriff auf zyklische Daten können Sie auf die Eingänge und Ausgänge der Slaves zugreifen und diese beobachten und steuern.

Der Zugriff erfolgt über:

- Slave Nummer.  
Diese Nummer entspricht der PROFIBUS-Adresse.
- Modulnummer.  
DP-Slaves können mehrere Module mit unterschiedlichen Ein-/Ausgangsbereichen enthalten.
- Ein-/Ausgangsbereich

### 2.4.1.5 Syntax der Prozessvariablen für den Master Klasse 1

#### Syntax

Eingänge:

```
DP: [<Verbindungsname>] Slave<Adresse>{M<Nummer>}_E{<Format>
<Offset>{.<Bit>}{,<Anzahl>}}
```

Ausgänge:

```
DP: [<Verbindungsname>] Slave<Adresse>{M<Nummer>}_A{<Format>
<Offset>{.<Bit>}{,<Anzahl>}}
```

#### Erklärungen

##### **DP**

DP-Protokoll für den Zugriff auf die Prozessvariable.

##### **<Verbindungsname>**

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

Beim DP-Protokoll ist der Verbindungsname der projektierte Name der Kommunikationsbaugruppe.

##### **Slave**

Kennzeichen, dass ein DP-Slave angesprochen wird.

##### **<Adresse>**

PROFIBUS-Adresse des Slaves.

Bereich: 0 ... 126.

##### **M**

Kennzeichen für die Nummer des Moduls, das den Ein- oder Ausgangsbereich enthält.

##### **<Nummer>**

Nummer des Moduls, das den Ein- oder Ausgangsbereich enthält.

##### **\_E**

Kennzeichen für einen Eingang. Eingänge sind nur lesbar.

##### **\_A**

Kennzeichen für einen Ausgang. Ausgänge sind les- und schreibbar.

##### **<Format>**

Format der gelieferten Daten.

Durch die Formatangabe wird der Datentyp festgelegt.

Über die Automation- und Custom Schnittstelle von OPC können prinzipiell alle angegebenen OLE-Datentypen gelesen werden. Einige Entwicklungswerkzeuge, wie z.B. Visual Basic, bieten jedoch nur eine eingeschränkte Menge von Datentypen an.

Die folgende Tabelle listet den entsprechenden Visual Basic-Typ auf, in dem der Variablenwert dargestellt werden kann.

Formatbezeichner	Beschreibung	OLE-Datentyp	Visual Basic-Typ
X	Bit	VT_BOOL	Boolean
BYTE oder B	Byte (unsigned 8)	VT_UI1	Byte
CHAR	Character (signed 8)	VT_I1	Integer
WORD oder W	Wort (unsigned 16)	VT_UI2	Long
INT	Integer (signed 16)	VT_I2	Integer
DWORD oder D	Doppelwort (unsigned 32)	VT_UI4	Double
DINT	Doppel-Integer (signed 32)	VT_I4	Long
REAL	Fließkommazahl (IEEE 4 Byte)	VT_R4	Single

#### <Offset>

Byteoffset im Adressraum des Slaves, an dem das Element liegt, das angesprochen werden soll. Wird ein Modul spezifiziert, so gilt der Offset innerhalb des Moduls. Ohne Angabe des Moduls bezieht sich der Offset auf den gesamten Ein-/Ausgabebereich des Slaves.

#### <Bit>

Bitnummer im adressierten Byte.

Die Spezifizierung eines Bits ist nur beim Formatbezeichner X zulässig, der Bereich liegt zwischen 0 und 7.

#### <Anzahl>

Anzahl der Elemente.

Der Datentyp (VT-ARRAY) der Variable ist ein Feld mit Elementen des angegebenen Formats. Wird Anzahl weggelassen, wird als Anzahl 1 angenommen und der Datentyp der Variable ist kein Feld.

---

#### Hinweis

Wird die Angabe des Moduls weggelassen, liefert die Variable den gesamten Ein- bzw. Ausgangsbereich über alle Module hinweg. Ein strukturierter Zugriff durch Angabe von Format, Offset und Anzahl ist möglich.

---

### 2.4.1.6 Protokoll-ID

Die Protokoll-ID für das DP-Protokoll lautet "DP".

### 2.4.1.7 Projektierter CP-Name

Der Verbindungsname spezifiziert den Kommunikationszugang durch Angabe des Kommunikationsprozessors, der mit dem PROFIBUS-Netz verbunden ist.

Beim DP-Protokoll ist der projektierte CP-Name der Name der Kommunikationsbaugruppe.

## Syntax

DP: [<ProjektiertterCPName>]

## Erklärung

<ProjektiertterCPName>

Der projektierte CP-Name wird mit SIMATIC STEP 7 / NCM projektiert.

## Beispiele für "Projektiertter CP-Name"

Typische Projektierte CP-Namen sind:

CP 5611 A2

CP 5621

CP 5613

CP 5614

CP 5623

CP 5624

### 2.4.1.8

## Beispiele für Prozessvariablen für den Master Klasse 1

Hier finden Sie Beispiele, die die Syntax von Variablennamen für DP-Variablen verdeutlichen.

## Eingänge

DP:[CP 5623]Slave005M003\_EB0

Slave005M003\_EB0

Eingangsbyte 0 des Moduls 3 von Slave 5

DP:[CP 5623]Slave005M003\_EB1,3

Slave005M003\_EB1,3

Feld mit 3 Bytes ab Eingangsbyte 1 des Moduls 3 von Slave 5

DP:[CP 5623]Slave005M003\_ED2

Slave005M003\_ED2

Doppelwort ab Eingangsbyte 2 des Moduls 3 von Slave 5

DP:[CP 5623]Slave004M003\_EReal0

Slave004M003\_EReal0

Fließkommazahl im Eingangsbereich von Slave 4, Modul 3

DP:[CP 5623]Slave004\_EB0,8

Slave004\_EB0,8

Die ersten 8 Bytes des gesamten Eingangsbereichs von Slave 4 über alle Module hinweg

## Ausgänge

*DP:[CP 5623]Slave005M007\_AB1*

Slave005M007\_AB1

Ausgangsbyte 1 des Moduls 7

*DP:[CP 5623]Slave005M007\_AX2.5*

Slave005M007\_AX2.5

Bit 5 im Ausgangsbyte 2 des Slave 5, Modul 7

*DP:[CP 5623]Slave004\_AW0,8*

Slave004\_AW0,8

Feld mit 8 Wörtern aus dem Ausgangsbereich von Slave 4 über alle Module hinweg

### 2.4.1.9 DPC1-Dienste

Mit den DPC1-Diensten können Sie auf Datensätze der DPC1-Slaves zugreifen. Die Übertragung der Datensätze erfolgt azyklisch.

Die Bedeutung der Datensätze ist durch den Hersteller des Slave festgelegt. Sie können beispielsweise für Konfigurationsdaten eines Antriebs genutzt werden.

Der OPC-Server kann beim Anmelden einer DPC1-Variablen nur die Syntax auf Korrektheit überprüfen, aber nicht, ob auf Grund der Projektierung des DPC1-Slave die Variable im Partnergerät gültig und die Größe des Datensatzes ausreichend ist.

### 2.4.1.10 Syntax der Prozessvariablen für DPC1-Dienste

#### Syntax

```
DP: [<Verbindungsname>] Slave<Adresse>S<Slot>Data<Index>
{,<Länge>} {,<Teilbereich>}
```

#### Erklärungen

##### DP

DP-Protokoll für den Zugriff auf die Prozessvariable.

##### <Verbindungsname>

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

##### Slave

Kennzeichen für den Zugriff auf einen Slave über das DP-Protokoll.

##### <Adresse>

PROFIBUS-Adresse des Slaves.

Bereich: 0 ... 126.

##### S

Kennzeichen für den Slot des Slaves, typischerweise ein Modul.

**<Slot>**

Slot im erweiterten Speicherbereich eines Slaves für azyklische Dienste. Slot und Index kennzeichnen einen Datensatz.

**Data**

Kennzeichen für den Zugriff auf einen Datensatz.

**<Index>**

Index innerhalb eines Slots im erweiterten Speicherbereich eines Slaves für azyklische Dienste. Slot und Index kennzeichnen einen Datensatz.

**<Länge>**

Länge des Datensatzes. Bereich zwischen 1 und 240.

**<Teilbereich>**

Der Teilbereich setzt sich wie folgt zusammen:

<Format><Offset>{.<Bit>}{,<Anzahl>}

**<Format>**

Format, in dem die Daten geliefert werden.

Wenn keine Formatangabe gemacht wird, wird das Format *Byte* verwendet.

Format-bezeichner	Beschreibung	OLE-Datentyp	Visual Basic-Typ
X	Bit	VT_BOOL	Boolean
BYTE oder B	Byte (unsigned 8)	VT_UI1	Byte
CHAR	Character (signed 8)	VT_I1	Integer
WORD oder W	Wort (unsigned 16)	VT_UI2	Long
INT	Integer (signed 16)	VT_I2	Integer
DWORD oder D	Doppelwort (unsigned 32)	VT_UI4	Double
DINT	Doppel-Integer (signed 32)	VT_I4	Long
REAL	Fließkommazahl	VT_R4	Single

**<Offset>**

Byteadresse im Datensatz für das Element, das angesprochen werden soll.

**<Bit>**

Bitnummer im adressierten Byte.

Die Spezifizierung eines Bits ist nur beim Formatbezeichner X zulässig.

**<Anzahl>**

Anzahl der Elemente

Der Datentyp der Variable ist ein Feld (VT\_ARRAY) mit Elementen des angegebenen Formats. Wird <Anzahl> weggelassen, wird als Anzahl 1 angenommen. Bei Anzahl 1 ist der Datentyp der Variable kein Feld.

**Wert des Datensatzes**

Rückgabewert des Datensatzitems

Der Datentyp ist VT\_ARRAY | VT\_UI1, falls kein Teilbereich angeben wird.

Der Wert ist nur lesbar; falls die Datensatzlänge angegeben wird, auch schreibbar.

---

**Hinweis**

Die durch die Parameter *Anzahl* und *Format* bestimmte Datenlänge darf die Größe des Datensatzes im Slave nicht überschreiten. Die Größe eines Datensatzes ist slave-abhängig und kann vom OPC-Server nicht kontrolliert werden.

Wenn Sie Teilbereiche definieren, müssen Sie folgendes beachten: Wenn ein Lesezugriff auf einen Datensatz erfolgt, wird vom Partnergerät zunächst immer der gesamte Datensatz gelesen. Erst anschließend wird der Teilbereich ausgewertet.

Beim Schreiben wird ebenfalls der gesamte Datensatz an das Partnergerät gesendet. Werden beim Schreibauftrag mehrere Teilbereiche über einen Mengenauftrag beschrieben, werden zuerst alle Teilbereiche in den Datensatz eingetragen und der Datensatz erst danach gesendet.

Aus diesem Grund sollten Sie alle OPC-Items mit Teilzugriffen auf einen Datensatz in einer Gruppe zusammenfassen und die gesamte Gruppe schreiben.

---

**Hinweis****Schreiben von Datensätzen an das Partnergerät**

Vermeiden Sie beim Schreiben von Datensätzen grundsätzlich Überschneidungen und Lücken, da nicht vorhersehbar ist, welcher Wert in diesem Fall geschrieben wird.

---

#### 2.4.1.11 Beispiele für Prozessvariablen für DPC1-Dienste

Hier finden Sie Beispiele, die die Syntax von Variablennamen für DPC1-Dienste verdeutlichen.

##### Variablennamen für DPC1

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

Slave005S003Data2,120,DWORD7

Zugriff auf das Doppelwort ab Offset 7 in einem Datensatz der Länge 120 Bytes in Slot 3, Index 2 des Slaves 5.

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

Slave005S003Data2,120,B8,4

Zugriff auf ein Feld mit 4 Bytes ab Offset 8 in einem Datensatz der Länge 120 Bytes in Slot 3, Index 2 des Slaves 5.

### 2.4.1.12 Fast Logic für CP 5613/CP 5614 und CP 5623/CP 5624

Der CP 5613/CP 5623 und der DP-Master-Teil des CP 5614/CP 5624 unterstützen die Eigenschaft Fast Logic. Dadurch können Sie den CP so parametrieren, dass er als Reaktion auf Datenänderung beim gleichen oder bei anderen Slaves Werte schreibt. Die Anwenderapplikation wird dabei über die Datenänderung informiert.

Der CP 5613/CP 5614 und CP 5623/CP 5624 stellen 4 Fast Logic Auslöser bereit, die über OPC-Steuervariablen konfiguriert und ausgewertet werden können.

#### Vorteile von Fast Logic

Die Verwendung von Fast Logic hat folgende Vorteile:

- Der OPC-Server und der OPC-Client sind entlastet.
- Die Datenübertragung findet schneller statt, weil sie unabhängig von der auf dem PC laufenden Software in der Hardware des CPs abläuft.

---

#### Hinweis

Nachdem ein Fast Logic-Trigger ausgelöst wurde, wird er anschließend automatisch deaktiviert. Sie müssen den Trigger dann erneut über die Variable FLActivate aktivieren.

Fast Logic funktioniert nur dann korrekt, wenn der DP-Master im Zustand OPERATE ist und die beteiligten Slaves im Zustand READY sind. Deshalb sollte ein Fast Logic Trigger von dem DP-Anwendungsprogramm erst aktiviert werden, wenn das Anwendungsprogramm den DP-Master in den Zustand OPERATE versetzt hat.

Solange Fast Logic-Triggere aktiv sind, darf kein DP-Anwendungsprogramm schreibend auf die Ausgangsbytes zugreifen, die über Fast Logic mit Eingangsbytes verknüpft sind.

---

### 2.4.1.13 Syntax der Steuervariablen für Fast Logic

#### Syntax

DP: [<Verbindungsname>] FL<Parameter><N>

#### Erklärungen

##### DP

DP-Protokoll für den Zugriff auf die Prozessvariable.

##### <Verbindungsname>

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

##### FL

Kennzeichen für Fast Logic.

##### <Parameter>

Mögliche Werte sind:

**State**

Gibt den Fast Logic Status zurück.

Rückgabewerte:

*CLEARED*

Auslöser *N* ist nicht aktiviert.

*ACTIVATED*

Auslöser *N* ist aktiviert.

*TRIGGERED*

Auslöser *N* hat die Überwachung ausgeführt.

OLE-Datentyp	Visual Basic Typ
VT_BSTR	String

**Activate**

Activate kann nur geschrieben werden.

Wenn ein Parameterblock geschrieben wird, wird die Fast Logic-Eigenschaft für den in der ItemID angegebenen Auslöser festgelegt.

Der Parameterblock, der geschrieben wird, ist ein Feld mit 8 Bytes und hat folgenden Aufbau:

*slave\_addr\_in\_byte*

Adresse des Slaves, dessen Eingänge für den Trigger selektiert werden

*index\_in\_byte*

Offset des Eingangsbytes des Triggers

*cmp\_value\_in\_byte*

Vergleichswert für das Eingangsbyte

*mask\_in\_byte*

Einzelne Bits im Eingangsbyte können maskiert werden, so dass sie beim Vergleich nicht berücksichtigt werden. Maskiert wird durch eine "1" im entsprechenden Bit, das heißt für *mask\_in\_byte==0x00* werden alle Bits von *cmp\_value\_in\_byte* für den Vergleich herangezogen. Der Trigger wird ausgelöst, wenn alle nicht maskierten Bits im selektierten Eingangsbyte gleich den Bits in *cmp\_value\_in\_byte* sind.

*slave\_addr\_out\_byte*

Selektiert den Slave, dessen Ausgangsbyte beim Eintreffen der Trigger-Bedingung verändert werden soll

*index\_out\_byte*

Offset des Ausgangsbytes

*value\_out\_byte*

Wert, der in das Ausgangsbyte geschrieben werden soll

***mask\_out\_byte***

Einzelne Bits im Ausgangsbyte können maskiert werden, so dass sie beim Eintreffen der Trigger-Bedingung nicht verändert werden. Maskiert wird durch eine "1" im entsprechenden Bit, das heißt für *mask\_out\_byte*==0x00 werden alle Bits von *value\_out\_byte* in das selektierte Ausgangsbyte geschrieben.

OLE-Datentyp	Visual Basic Typ
VT_ARRAY of VT_UI1	Byte()

**Clear**

Clear kann nur geschrieben werden.

Wenn der boolsche Wert TRUE geschrieben wird, wird der in der ItemID spezifizierte Fast Logic-Auslöser deaktiviert.

OLE-Datentyp	Visual Basic Typ
VT_BOOL	Boolean

**<N>**

Nummer des verwendeten Fast Logic Auslösers. Wert zwischen 1 und 4.

#### 2.4.1.14 DP-spezifische Informationsvariablen

Mit den DP-spezifischen Informationsvariablen für den Master Klasse 1 werden Informationen über den DP-Master und die angeschlossenen DP-Slaves abgefragt.

Sie können folgende Informationen abfragen:

- Betriebsart des DP-Masters
- Ereignismeldungen des DP-Masters
- Aktivitätsüberwachung des CPs
- Betriebszustand eines DP-Slaves
- Typ eines DP-Slaves
- sonstige Informationen zur tiefergehenden Diagnose

#### 2.4.1.15 Syntax der DP-spezifischen Informationsvariablen

##### Syntax

DP: [<Verbindungsname>]<Informationsparameter>

##### Erklärungen

###### DP

DP-Protokoll für den Zugriff auf die Prozessvariable.

**<Verbindungsname>**

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

**<Informationsparameter>**

Mögliche Werte sind:

- Masterstate
- EvAutoclear
- EvWatchdog
- EvClass2Master
- WatchdogTimeout
- Slave/nSlvState
- Slave/nSlvDiag
- Slave/nSlvRestart
- Slave/nMiscSlvType
- sonstige Slave-Informationen

**Masterstate**

Aktuelle Betriebsart des DP-Masters.

Die aktuelle Betriebsart kann sowohl gelesen als auch geschrieben werden. Das Einstellen der Betriebsart durch Schreiben eines der unten genannten Werte ist nur in Abhängigkeit von der DP-Anwendungsumgebung möglich.

Eingabe- und Rückgabewerte:

*OFFLINE*

Keine Kommunikation zwischen Master und Slave.

*STOP*

Bis auf Diagnosedaten keine Kommunikation zwischen Master und Slave

*CLEAR*

Parametrier- und Konfigurierphase

*AUTOCLEAR*

Autoclear-Phase, der DP-Master erreicht nicht mehr alle Slaves

*OPERATE*

Produktivphase

OLE-Datentyp	Visual Basic Typ
VT_BSTR	String

**EvAutoclear**

Signalisiert einen Fehler während der Kommunikation mit DP-Slaves. Das DP-System wird selbstständig in die Betriebsart CLEAR heruntergefahren.

Rückgabewerte:

*True*

Es ist ein Fehler während der Kommunikation mit DP-Slaves aufgetreten und das System ist in die Betriebsart CLEAR heruntergefahren

*False*

Es ist kein Fehler aufgetreten

OLE-Datentyp	Visual Basic Typ
VT_BOOL	Boolean

---

#### Hinweis

Sie können EvAutoclear nur verwenden, wenn AUTOCLEAR während der Projektierung eingestellt worden ist.

Wenn dieser Informationsparameter ausgegeben wird, ist keine Reaktion erforderlich.

---

#### EvWatchdog

Signalisiert die Überschreitung der Auftragsüberwachungszeit der Baugruppe. Der OPC-Server hat innerhalb der Überwachungszeit keinen DP-Funktionsaufruf durchgeführt. Möglicherweise sind der OPC-Client oder der OPC-Server nicht mehr verfügbar. Die Überwachungszeit stellen Sie in der Projektierung ein.

Rückgabewerte:

*True*

Die Überwachungszeit der Baugruppe ist überschritten

*False*

Die Überwachungszeit der Baugruppe ist nicht überschritten

OLE-Datentyp	Visual Basic Typ
VT_BOOL	Boolean

#### EvClass2Master

Signalisiert einen Zugriff eines DP-Masters der Klasse 2.

Rückgabewerte:

*True*

Ein DP-Master der Klasse 2 nimmt am Busverkehr teil und greift auf interne Diagnoselisten des DP-Masters der Klasse 1 zu

*False*

Es greift kein DP-Master der Klasse 2 zu

OLE-Datentyp	Visual Basic Typ
VT_BOOL	Boolean

---

**Hinweis**

Eine Reaktion auf dieses Ereignis durch ein Anwenderprogramm ist nicht erforderlich.

---

**WatchdogTimeout**

Aktivitätsüberwachung auf dem CP. Wenn ein Wert geschrieben wird, kann die Aktivitätsüberwachung eingestellt werden. Der Standardwert wird in der Projektierung vorgegeben.

Rückgabewerte:

*0*

Überwachung aus

*400 - 102000*

Beliebiger Wert in ms. Der Wert wird auf ein Vielfaches von 400 gerundet.

*6000*

Default

OLE-Datentyp	Visual Basic Typ
VT_UI4	Double

**Slave/nSlvState**

Aktueller Betriebszustand des DP-Slaves mit der Adresse *n*.

Rückgabewerte:

*OFFLINE*

Keine Kommunikation zwischen Master und Slave

*NOT\_ACTIVE*

DP-Slave ist nicht aktiviert

*READY*

DP-Slave ist in der Datentransferphase

*READY\_DIAG*

DP-Slave ist in der Datentransferphase; zusätzlich sind Diagnosedaten vorhanden

*NOT\_READY*

DP-Slave ist nicht in der Datentransferphase

*NOT\_READY\_DIAG*

DP-Slave ist nicht in der Datentransferphase; zusätzlich sind Diagnosedaten vorhanden

OLE-Datentyp	Visual Basic Typ
VT_BSTR	String

**Slave/nSlvDiag**

Letzte Diagnosedaten des DP-Slaves mit der Adresse *n*. Der Aufbau der Diagnosedaten ist slave-abhängig und muss der Dokumentation des Slave entnommen werden.

OLE-Datentyp	Visual Basic Typ
VT_ARRAY   VT_UI1	Byte()

**Slave/nSlvRestart**

Item zum Neustarten von DP-Slaves, d. h. Deaktivieren und gleich wieder Aktivieren der DP-Slaves mit einem Item. Diese beiden internen Funktionsaufrufe werden mit dem zeitlichen Abstand der projektierten Zykluszeit durchgeführt.

OLE-Datentyp	Visual Basic Typ
VT_BOOL	Boolean

Rückgabewerte:

*True*

Slave *n* neu starten

*False*

Keine Funktion

**Slave/nMiscSlvType**

Typ des DP-Slave mit der Adresse *n*.

Rückgabewerte:

*NO\_SLV*

Kein DP-Slave

*NORM*

Norm-DP-Slave

*ET200\_U*

Nicht Norm-Slave: ET 200 U

*ET200K\_B*

Nicht Norm-Slave: ET 200 K/B

*ET200\_SPM*

Nicht Norm-Slave: Allgemeine SPM-Station

*UNDEFINED*

Unbekannter DP-Slave

OLE-Datentyp	Visual Basic Typ
VT_BSTR	String

**sonstige Slave-Informationen**

Die folgenden Slave-spezifischen Informationsparameter sind nicht für den Produktivbetrieb vorgesehen. Bei Problemen kann es jedoch nützlich sein, die aufgeführten Variablen auszuwerten.

Alle Variablen sind in der Regel direkt Low-Level-DP-Diensten zugeordnet. Das Lesen dieser Variablen erfolgt wie bei den anderen Informationsvariablen. Das Ergebnis des Aufrufes ist der Byte-Dump des jeweiligen Low-Level-DP-Dienstes.

*SlavenMiscReadSlvParCfgData*

Konfigurationsdaten des Slaves an der Adresse *n* zur Beschreibung der Ein- und Ausgangsbereiche und der Datenkonsistenz

*SlavenMiscReadSlvParUserData*

Anwendungsdaten des Slaves *n* (Teil der Parameterdatei)

*SlavenMiscReadSlvParPrmData*

Parametrierdaten des Slaves *n*

*SlavenMiscReadSlvParType*

SI-Flag und Typ des Slaves *n* (Slave-bezogene Flags, Slave-Typ und weitere reservierte Daten)

*SlavenMiscSlvDiag*

Diagnosedaten des Slaves *n*

#### 2.4.1.16 Beispiele für DP-spezifische Informationsvariablen

Hier finden Sie einige Beispiele für Rückgabewerte von DP-spezifischen Informationsvariablen.

##### Betriebsart DP-Master

*DP:[CP 5623]MasterState*

MasterState

kann beispielsweise folgenden Wert zurückgeben:

*OPERATE*

Der DP-Master befindet sich in der Produktivphase.

##### Aktueller Betriebszustand eines Slaves

*DP:[CP 5623]Slave3SlvState*

Slave3SlvState

kann beispielsweise folgenden Wert zurückgeben:

*READY*

Der DP-Slave mit der Adresse 3 ist in der Datentransferphase.

##### Typ eines Slaves

*DP:[CP 5623]Slave3MiscSlvType*

Slave3MiscSlvType

kann beispielsweise folgenden Wert zurückgeben:

*ET200\_U*

Bei dem Slave 3 handelt es sich um eine ET 200U.

### 2.4.1.17 Syntax der systemspezifischen Informationsvariablen

DP: [SYSTEM] &version()

#### &version()

Liefert eine Versionskennung für den DP-OPC-Server, hier z.B. die Zeichenfolge  
*SIMATIC NET Core Server DP V 7.xxxx.yyyy.zzzz Copyright 2012*

Datentyp: VT\_BSTR

Zugriffsrecht: nur lesbar

## 2.4.2 DP Master Klasse 2

### DP-Master Klasse 2

Ein DP-Master Klasse 2 kann Konfigurations- und Online-Diagnoseaufträge durchführen. Die wesentlichen DP-Master Klasse 2-Funktionen sind:

- Master-Diagnose: Lesen der Master-Klasse-1-Diagnosedaten
- Slave-Diagnose: Lesen von Diagnosedaten eines Slaves und der Slave-Konfiguration
- I/O Daten: Lesen von Ein- und Ausgangsdaten eines Slaves
- Datensätze: Datensätze für einen Slave schreiben und lesen

Mit den azyklischen Diensten zum Zugriff auf E/A-Daten können Sie lesend auf die Eingänge und Ausgänge der Slaves und auf deren Diagnosedaten zugreifen und diese beobachten. Voraussetzung ist, dass die Slaves die DP-V1-Zusatzfunktionen unterstützen. Zudem haben Sie Zugriff auf die Daten eines Master Klasse 1.

Der Zugriff erfolgt:

- über die PROFIBUS-Adresse des Slaves  
 Slave-Geräte können nicht von sich aus eine Kommunikation aufnehmen. Sie erhalten den Token nicht und antworten nur auf die an sie gerichteten Anfragen der Master.  
 Slaves bezeichnet man deshalb als passive Teilnehmer. Diese Nummer entspricht der PROFIBUS-Adresse.
- über die PROFIBUS-Adresse des Masters  
 Zugriff auf die Diagnosedaten eines Masters Klasse 1 von einem Master Klasse 2 aus.

### 2.4.2.1 Protokoll-ID

Alle Items beginnen mit dem Präfix "DP2".

### 2.4.2.2 Projektierter CP-Name

Der projektierte CP-Name spezifiziert den eingesetzten Kommunikationsprozessor (CP).

## Syntax

DP2: [<ProjektierterCPName>]

## Erklärung

### <ProjektiertterCPName>

Der projektierte CP-Name wird mit SIMATIC STEP 7 / NCM projektiert.

## Beispiel

*DP2:[CP 5623]*

### 2.4.2.3 Syntax der Prozessvariablen für Master Diagnose

## Einleitung

Der folgende Abschnitt beschreibt die verschiedenen Syntaxformen, mit denen Sie auf die Diagnosedaten eines DP-Master Klasse 1 zugreifen können.

## Syntax

DP2 : [<ProjektiertterCPName>] Master<AdresseMaster>MstDiag

### DP2-Items des Busteilnehmers

#### Master

Kennzeichen für den Zugriff auf einen Master über das DP-Protokoll.

#### <AdresseMaster>

PROFIBUS Stationsadresse des Busteilnehmers.

Bereich: 0 bis 126

## Erklärung

### DP2-Items des Busteilnehmers - Item für die Master-Diagnose

#### MstDiag

Letzte Systemdiagnosedaten des DP-Master.

Datentyp: VT\_ARRAY

Zugriffsrecht: nur lesbar

OLE-Datentyp: VT\_BOOL

Beschreibung: 126 Elemente

Jedes Arrayelement signalisiert, ob der zugeordnete Slave Diagnosedaten gesendet hat, wobei der Arrayindex der PROFIBUS-Adresse entspricht.

Wert	Bedeutung
FALSE	Slave hat keine Diagnosedaten gesendet
TRUE	Slave hat Diagnosedaten gesendet

## Syntax

```
DP2 : [<ProjektierterCPName>]Master<AdresseMaster>MstState
```

### Erklärung

#### DP2-Items des Busteilnehmers - Diagnose-Items Master

##### **MstState**

Status des DP-Master. Enthalten den aktuellen Betriebszustand sowie einige Versionsinformationen.

Datentyp: VT\_ARRAY

Zugriffsrecht: nur lesbar

OLE-Datentyp: VT\_UI1

Beschreibung: 16 Elemente

Byte	Beschreibung
1	Betriebszustand 0x40 – STOP 0x80 – CLEAR 0xC0 - OPERATE
2 und 3	Ident-Nummer
4	Version der Hardware (DDLM/user interface)
5	Version der Software (DDLM/user interface)
6	Version der Hardware
7	Version der Software
8 bis 16	Reserviert

## Syntax

```
DP2 : [<ProjektierterCPName>]Master<AdresseMaster>
      DataTransferList
```

### Erklärung

#### DP2-Items des Busteilnehmers - Diagnose-Items Master

##### **DataTransferList**

Data-Transfer-Liste des DP-Master.

Datentyp: VT\_ARRAY

Zugriffsrecht: nur lesbar

OLE-Datentyp: VT\_BOOL

Beschreibung: 126 Elemente

Jedes Arrayelement signalisiert, ob der zugeordnete Slave in der Produktivphase ist und gesendet hat, wobei der Arrayindex der PROFIBUS-Adresse entspricht.

Wert	Beschreibung
FALSE	Slave ist nicht in der Datentransferphase
TRUE	Slave ist in der Datentransferphase

## Syntax

```
DP2:[<ProjektierterCPName>]Master<AdresseMaster>Slave
<Adresse>SlvDiag
```

### Erklärung

#### DP2-Items des Busteilnehmers - Diagnose-Items Master

##### <Adresse>

PROFIBUS Stationsadresse des Busteilnehmers. Der Busteilnehmer muss nicht notwendigerweise am PROFIBUS angeschlossen sein, damit das Item angelegt und (ggf. mit einem Zugriffsfehler) verwendet werden kann.

Bereich: 0 bis 126

##### Hinweis

Der SIMATIC NET OPC-Server kann erst durch den Erfolg eines auszuführenden Slave-Dienstes erkennen, ob der adressierte Busteilnehmer tatsächlich DP-fähig ist. Bei kritischen Busteilnehmern, die auf solche Dienste möglicherweise fehlerhaft reagieren, liegt deshalb die Erzeugung entsprechender Items in der Verantwortung des Anwenders.

##### SlvDiag

Letzte Diagnosedaten des DP-Slaves, die beim DP-Master hinterlegt sind.

Datentyp: VT\_ARRAY

Zugriffsrecht: nur lesbar

OLE-Datentyp: VT\_UI1

Für die speziellen Inhalte und Beschreibung der Elemente des Feldes *SlvDiag* gibt es folgende weiteren Einzel-Items:

## Syntax

```
DP2:[<ProjektierterCPName>]Master<AdresseMaster>Slave
<Adresse>SlvDiagMasterLock
```

Dieses Bit wird gesetzt, wenn der DP-Slave bereits von einem anderen Master parametrisiert worden ist, d. h. der eigene Master hat momentan keinen Zugriff auf diesen Slave.

Bit 7 des 1. (Stationsstatus-)Byte des Feldes *SlvDiag*.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

```
DP2 : [<ProjektierterCPName>]Master<AdresseMaster>Slave
      <Adresse>SlvDiagPrmFault
```

Dieses Bit wird vom DP-Slave gesetzt, falls das letzte Parametriertelegramm für diesen Slave fehlerhaft war (beispielsweise falsche Länge, falsche Ident-Number, ungültige Parameter).

Bit 6 des 1. (Stationsstatus-)Byte des Feldes *SlvDiag*.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

```
DP2 : [<ProjektierterCPName>]Master<AdresseMaster>Slave
      <Adresse>SlvDiagInvalidSlaveResponse
```

Dieses Bit wird gesetzt, sobald von einem angesprochenen DP-Slave eine unplausible Antwort empfangen wird.

Bit 5 des 1. (Stationsstatus-)Byte des Feldes *SlvDiag*.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

```
DP2 : [<ProjektierterCPName>]Master<AdresseMaster>Slave
      <Adresse>SlvDiagNotSupported
```

Dieses Bit wird gesetzt, sobald eine Funktion angefordert wurde, die von diesem Slave nicht unterstützt wird (beispielsweise wird der Betrieb im SYNC-Mode gefordert, vom Slave aber nicht unterstützt).

Bit 4 des 1. (Stationsstatus-)Byte des Feldes *SlvDiag*.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

```
DP2 : [<ProjektierterCPName>]Master<AdresseMaster>Slave
      <Adresse>SlvDiagExtDiag
```

Dieses Bit wird vom DP-Slave gesetzt. Wenn das Bit gesetzt ist, dann muss in dem Slave-spezifischen Diagnosebereich (Ext\_Diag\_Data) ein Diagnoseeintrag vorliegen.

Wenn das Bit nicht gesetzt ist, dann kann in dem Slave-spezifischen Diagnosebereich (Ext\_Diag\_Data) eine Statusmeldung vorliegen. Die Bedeutung dieser Statusmeldung ist applikationsspezifisch zu vereinbaren.

Bit 3 des 1. (Stationsstatus-)Byte des Feldes *SlvDiag*.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

```
DP2 : [<ProjektierterCPName>]Master<AdresseMaster>Slave
      <Adresse>SlvDiagCfgFault
```

Dieses Bit wird gesetzt, sobald die vom DP-Master zuletzt gesendeten Konfigurationsdaten mit denjenigen, die der DP-Slave ermittelt hat, nicht übereinstimmen. Das bedeutet, es liegt ein Konfigurationsfehler vor.

Bit 2 des 1. Stationsstatus-Byte des Feldes *SlvDiag*.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

```
DP2 : [<ProjektierterCPName>]Master<AdresseMaster>Slave
      <Adresse>SlvDiagStationNotReady
```

Dieses Bit wird gesetzt, wenn der DP-Slave noch nicht für den Produktivdatenaustausch bereit ist.

Bit 1 des 1. (Stationsstatus-)Byte des Feldes *SlvDiag*.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

```
DP2 : [<ProjektierterCPName>]Master<AdresseMaster>Slave
      <Adresse>SlvDiagStationNonExistent
```

Dieses Bit setzt der DP-Master, falls der DP-Slave nicht über den Bus erreichbar ist. Wenn dieses Bit gesetzt ist, dann enthalten die Diagnosebits den Zustand der letzten Diagnosemeldung oder den Initialwert. Der DP-Slave setzt dieses Bit fest auf Null.

Bit 0 des 1. (Stationsstatus-)Byte des Feldes *SlvDiag*.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

```
DP2 : [<ProjektierterCPName>]Master<AdresseMaster>Slave
      <Adresse>SlvDiagDeactivated
```

Dieses Bit wird gesetzt, sobald der DP-Slave im lokalen Parametersatz als nicht aktiv gekennzeichnet und aus der zyklischen Bearbeitung herausgenommen wurde.

Bit 7 des 2. (Stationsstatus-)Byte des Feldes *SlvDiag*.

Bit 6 des 2. (Stationsstatus-)Byte des Feldes *SlvDiag* ist reserviert.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

```
DP2 : [<ProjektierterCPName>]Master<AdresseMaster>Slave
      <Adresse>SlvDiagSyncMode
```

Dieses Bit wird vom DP-Slave gesetzt, sobald dieser DP-Slave das Sync-Steuerkommando erhalten hat.

Bit 5 des 2. (Stationsstatus-)Byte des Feldes *SlvDiag*.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

```
DP2 : [<ProjektierterCPName>]Master<AdresseMaster>Slave
      <Adresse>SlvDiagFreezeMode
```

Dieses Bit wird vom DP-Slave gesetzt, sobald dieser DP-Slave das Freeze-Steuerkommando erhalten hat.

Bit 4 des 2. (Stationsstatus-)Byte des Feldes *SlvDiag*.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

```
DP2 : [<ProjektierterCPName>]Master<AdresseMaster>Slave
      <Adresse>SlvDiagWDOn
```

Dieses Bit wird vom DP-Slave gesetzt. Wenn dieses Bit auf 1 gesetzt ist, dann ist die Ansprechüberwachung beim DP-Slave aktiviert.

Bit 3 des 2. (Stationsstatus-)Byte des Feldes *SlvDiag*.

Bit 2 des 2. (Stationsstatus-)Byte des Feldes *SlvDiag* ist fest auf 1 gesetzt.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

```
DP2 : [<ProjektierterCPName>]Master<AdresseMaster>Slave
      <Adresse>SlvDiagStatDiag
```

Wenn der DP-Slave dieses Bit setzt, dann muss der DP-Master solange Diagnoseinformationen abholen, bis dieses Bit wieder gelöscht wird. Der DP-Slave setzt zum Beispiel dieses Bit, wenn er keine gültigen Nutzdaten zur Verfügung stellen kann.  
Bit 1 des 2. (Stationsstatus-)Byte des Feldes *SlvDiag*.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

```
DP2 : [<ProjektierterCPName>]Master<AdresseMaster>Slave
      <Adresse>SlvDiagPrmReq
```

Wenn der DP-Slave dieses Bit setzt, dann muss er neu parametriert und konfiguriert werden. Das Bit bleibt solange gesetzt, bis eine Parametrierung erfolgt ist.  
Bit 0 des 2. (Stationsstatus-)Byte des Feldes *SlvDiag*.

### Anmerkung

Wenn Bit 1 (*SlvDiagStatDiag*) und Bit 0 (*SlvDiagPrmReq*) gesetzt sind, dann hat Bit 0 die höhere Priorität.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

```
DP2 : [<ProjektierterCPName>]Master<AdresseMaster>Slave
      <Adresse>SlvDiagExtDiagOverflow
```

Wenn dieses Bit gesetzt ist, dann liegen mehr Diagnoseinformationen vor, als in *SlvDiagExtDiagData* angegeben sind. Zum Beispiel setzt der DP-Slave dieses Bit, wenn mehr Kanaldiagnosen vorliegen, als der DP-Slave in seinen Sendepuffer eintragen kann. Oder der DP-Master setzt dieses Bit, wenn der DP-Slave mehr Diagnosedaten sendet, als der DP-Master in seinem Diagnosepuffer berücksichtigen kann.

Bit 7 des 3. (Stationsstatus-)Byte des Feldes *SlvDiag*.

Bit 6 bis 0 des 3. (Stationsstatus-)Byte des Feldes *SlvDiag* sind reserviert.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

```
DP2 : [<ProjektierterCPName>]Master<AdresseMaster>Slave
      <Adresse>SlvDiagMasterAddr
```

In Byte 4 des Feldes *SlvDiag* (das *Diag.MasterAdd*-Byte) wird die Adresse des DP-Masters eingetragen, der diesen DP-Slave parametriert hat. Wenn der DP-Slave von keinem DP-Master parametriert ist, dann setzt der DP-Slave in dieses Byte die Adresse 255 ein.  
Byte 4 (*Diag.MasterAdd*-Byte) des Feldes *SlvDiag*.

Datentyp: VT\_UI1

Zugriffsrecht: nur lesbar

## Syntax

```
DP2 : [<ProjektierterCPName>]Master<AdresseMaster>Slave
      <Adresse>SlvDiagIdentNumber
```

In diesem Wort wird die Herstellerkennung für einen DP-Slave-Typ angegeben. Diese Kennung kann zum Einen für Prüfungszwecke und zum Anderen zur genauen Identifizierung herangezogen werden.

5. und 6. (IdentNumber -)Byte des Feldes *SlvDiag*.

Datentyp: VT\_UI2

Zugriffsrecht: nur lesbar

## Syntax

```
DP2 : [<ProjektierterCPName>]Master<AdresseMaster>Slave
      <Adresse>SlvDiagExtDiagData
```

In diesem Feld von Bytes (die *ExtDiagData*-Bytes) mit einer Länge von bis zu 26 Byte kann der DP-Slave seine spezifische Diagnose ablegen. Es wird eine Blockstruktur mit je einem Header-Byte für die geräte- und kennungsbezogene Diagnose vorgeschrieben.

7. bis 32. (*ExtDiagData*-)Byte des Feldes *SlvDiag*.

Datentyp: VT\_ARRAY

Zugriffsrecht: nur lesbar

OLE-Datentyp: VT\_UI1

### 2.4.2.4 Syntax der Prozessvariablen für Slave Diagnose

#### Einleitung

Der folgende Abschnitt beschreibt die verschiedenen Syntaxformen, mit denen Sie auf die Diagnosedaten eines Slave zugreifen können.

## Syntax

```
DP2 : [<ProjektierterCPName>]Slave<Adresse>SlvDiag
```

#### Erklärung

##### DPMCL2-Items des Busteilnehmers - Diagnose-Items Slave

###### **SlvDiag**

Letzte Diagnosedaten des DP-Slave.

Datentyp: VT\_ARRAY

Zugriffsrecht: nur lesbar

OLE-Datentyp: VT\_UI1

Für die speziellen Inhalte und Beschreibung der Elemente des Feldes *SlvDiag* gibt es folgende weiteren Einzelitems:

## Syntax

```
DP2 : [<ProjektierterCPName>]Slave<Adresse>SlvDiagMasterLock
```

Dieses Bit wird gesetzt, wenn der DP-Slave bereits von einem anderen Master parametriert worden ist, d. h. der eigene Master hat momentan keinen Zugriff auf diesen Slave.  
Bit 7 des 1. (Stationsstatus-)Byte des Feldes *SlvDiag*.

*Datentyp:* VT\_BOOL

*Zugriffsrecht:* nur lesbar

## Syntax

DP2 : [<ProjektierterCPName>] Slave<Adresse>S1vDiagPrmFault

Dieses Bit wird vom DP-Slave gesetzt, falls das letzte Parametriertelegramm für diesen Slave fehlerhaft war (beispielsweise falsche Länge, falsche Ident-Number, ungültige Parameter).

Bit 6 des 1. (Stationsstatus-)Byte des Feldes *SlvDiag*.

*Datentyp:* VT\_BOOL

*Zugriffsrecht:* nur lesbar

## Syntax

DP2 : [<ProjektierterCPName>] Slave<Adresse>S1vDiagInvalidSlaveResponse

Dieses Bit wird gesetzt, sobald von einem angesprochenen DP-Slave eine unplausible Antwort empfangen wird.

Bit 5 des 1. (Stationsstatus-)Byte des Feldes *SlvDiag*.

*Datentyp:* VT\_BOOL

*Zugriffsrecht:* nur lesbar

## Syntax

DP2 : [<ProjektierterCPName>] Slave<Adresse>S1vDiagNotSupported

Dieses Bit wird gesetzt, sobald eine Funktion angefordert wurde, die von diesem Slave nicht unterstützt wird (beispielsweise wird der Betrieb im SYNC-Mode gefordert, vom Slave aber nicht unterstützt).

Bit 4 des 1. (Stationsstatus-)Byte des Feldes *SlvDiag*.

*Datentyp:* VT\_BOOL

*Zugriffsrecht:* nur lesbar

## Syntax

DP2 : [<ProjektierterCPName>] Slave<Adresse>S1vDiagExtDiag

Dieses Bit wird vom DP-Slave gesetzt. Wenn das Bit gesetzt ist, dann muss in dem Slave-spezifischen Diagnosebereich (*Ext\_Diag\_Data*) ein Diagnoseeintrag vorliegen.

Wenn das Bit nicht gesetzt ist, dann kann in dem Slave-spezifischen Diagnosebereich (*Ext\_Diag\_Data*) eine Statusmeldung vorliegen. Die Bedeutung dieser Statusmeldung ist applikationsspezifisch zu vereinbaren.

Bit 3 des 1. (Stationsstatus-)Byte des Feldes *SlvDiag*.

Datentyp: VT\_BOOL

Zugriffsrecht: Read-Only

## Syntax

DP2 : [<ProjektierterCPName>] Slave<Adresse>S1vDiagCfgFault

Dieses Bit wird gesetzt, sobald die vom DP-Master zuletzt gesendeten Konfigurationsdaten mit denjenigen, die der DP-Slave ermittelt hat, nicht übereinstimmen. Das bedeutet, es liegt ein Konfigurationfehler vor.

Bit 2 des 1. Stationsstatus-Byte des Feldes *S1vDiag*.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

DP2 : [<ProjektierterCPName>] Slave<Adresse>S1vDiagStationNotReady

Dieses Bit wird gesetzt, wenn der DP-Slave noch nicht für den Produktivdatenaustausch bereit ist.

Bit 1 des 1. (Stationsstatus-)Byte des Feldes *S1vDiag*.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

DP2 : [<ProjektierterCPName>] Slave<Adresse>S1vDiagStationNonExistent

Dieses Bit setzt der DP-Master, falls der DP-Slave nicht über den Bus erreichbar ist. Wenn dieses Bit gesetzt ist, dann enthalten die Diagnosebits den Zustand der letzten Diagnosemeldung oder den Initialwert. Der DP-Slave setzt dieses Bit fest auf Null.

Bit 0 des 1. (Stationsstatus-)Byte des Feldes *S1vDiag*.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

DP2 : [<ProjektierterCPName>] Slave<Adresse>S1vDiagDeactivated

Dieses Bit wird gesetzt, sobald der DP-Slave im lokalen Parametersatz als nicht aktiv gekennzeichnet und aus der zyklischen Bearbeitung herausgenommen wurde.

Bit 7 des 2. (Stationsstatus-)Byte des Feldes *S1vDiag*.

Bit 6 des 2. (Stationsstatus-)Byte des Feldes *S1vDiag* ist reserviert.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

DP2 : [<ProjektierterCPName>] Slave<Adresse>SlvDiagSyncMode

Dieses Bit wird vom DP-Slave gesetzt, sobald dieser DP-Slave das Sync-Steuerkommando erhalten hat.

Bit 5 des 2. (Stationsstatus-)Byte des Feldes *SlvDiag*.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

DP2 : [<ProjektierterCPName>] Slave<Adresse>SlvDiagFreezeMode

Dieses Bit wird vom DP-Slave gesetzt, sobald dieser DP-Slave das Freeze-Steuerkommando erhalten hat.

Bit 4 des 2. (Stationsstatus-)Byte des Feldes *SlvDiag*.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

DP2 : [<ProjektierterCPName>] Slave<Adresse>SlvDiagWDOn

Dieses Bit wird vom DP-Slave gesetzt. Wenn dieses Bit auf 1 gesetzt ist, dann ist die Ansprechüberwachung beim DP-Slave aktiviert.

Bit 3 des 2. (Stationsstatus-)Byte des Feldes *SlvDiag*.

Bit 2 des 2. (Stationsstatus-)Byte des Feldes *SlvDiag* ist fest auf 1 gesetzt.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

DP2 : [<ProjektierterCPName>] Slave<Adresse>SlvDiagStatDiag

Wenn der DP-Slave dieses Bit setzt, dann muss der DP-Master solange Diagnoseinformationen abholen, bis dieses Bit wieder gelöscht wird. Der DP-Slave setzt zum Beispiel dieses Bit, wenn er keine gültigen Nutzdaten zur Verfügung stellen kann.  
Bit 1 des 2. (Stationsstatus-)Byte des Feldes *SlvDiag*.

Datentyp: VT\_BOOL

Zugriffsrecht: nur lesbar

## Syntax

DP2 : [<ProjektierterCPName>] Slave<Adresse>SlvDiagPrmReq

Wenn der DP-Slave dieses Bit setzt, dann muss er neu parametriert und konfiguriert werden. Das Bit bleibt solange gesetzt, bis eine Parametrierung erfolgt ist.

Bit 0 des 2. (Stationsstatus-)Byte des Feldes *SlvDiag*.

**Anmerkung**

Wenn Bit 1 (*SlvDiagStatDiag*) und Bit 0 (*SlvDiagPrmReq*) gesetzt sind, dann hat Bit 0 die höhere Priorität.

**Datentyp:** VT\_BOOL

**Zugriffsrecht:** nur lesbar

**Syntax**

```
DP2 : [<ProjektierterCPName>] Slave<Adresse>SlvDiagExtDiagOverflow
```

Wenn dieses Bit gesetzt ist, dann liegen mehr Diagnoseinformationen vor, als in *SlvDiagExtDiagData* angegeben sind. Zum Beispiel setzt der DP-Slave dieses Bit, wenn mehr Kanaldiagnosen vorliegen, als der DP-Slave in seinen Sendepuffer eintragen kann. Oder der DP-Master setzt dieses Bit, wenn der DP-Slave mehr Diagnosedaten sendet, als der DP-Master in seinem Diagnosepuffer berücksichtigen kann.

Bit 7 des 3. (Stationsstatus-)Byte des Feldes *SlvDiag*.

Bit 6 bis 0 des 3. (Stationsstatus-)Byte des Feldes *SlvDiag* sind reserviert.

**Datentyp:** VT\_BOOL

**Zugriffsrecht:** nur lesbar

**Syntax**

```
DP2 : [<ProjektierterCPName>] Slave<Adresse>SlvDiagMasterAddr
```

In Byte 4 des Feldes *SlvDiag* (das *Diag.MasterAdd*-Byte) wird die Adresse des DP-Masters eingetragen, der diesen DP-Slave parametriert hat. Wenn der DP-Slave von keinem DP-Master parametriert ist, dann setzt der DP-Slave in dieses Byte die Adresse 255 ein. Byte 4 (*Diag.MasterAdd*-Byte) des Feldes *SlvDiag*.

**Datentyp:** VT\_UI1

**Zugriffsrecht:** nur lesbar

**Syntax**

```
DP2 : [<ProjektierterCPName>] Slave<Adresse>SlvDiagIdentNumber
```

In diesem Wort wird die Herstellerkennung für einen DP-Slave-Typ angegeben. Diese Kennung kann zum Einen für Prüfungszwecke und zum Anderen zur genauen Identifizierung herangezogen werden.

5. und 6. (IdentNumber -)Byte des Feldes *SlvDiag*.

**Datentyp:** VT\_UI2

**Zugriffsrecht:** nur lesbar

**Syntax**

```
DP2 : [<ProjektierterCPName>] Slave<Adresse>SlvDiagExtDiagData
```

In diesem Feld von Bytes (die *ExtDiagData*-Bytes) mit einer Länge von bis zu 26 Byte kann der DP-Slave seine spezifische Diagnose ablegen. Es wird eine Blockstruktur mit je einem

Header-Byte für die geräte- und kennungsbezogene Diagnose vorgeschrieben.  
7. bis 32. (*ExtDiagData*-)Byte des Feldes *SlvDiag*.

Datentyp: VT\_ARRAY

Zugriffsrecht: nur lesbar

OLE-Datentyp: VT\_UI1

## Syntax

```
DP2 : [<ProjektierterCPName>] Slave<Adresse>SlvCFGData
```

### Erklärung

#### DPMCL2-Items des Busteilnehmers - Diagnose-Items Slave

##### **SlvCFGData**

Konfigurationsdaten des DP-Slaves, die beim DP-Master hinterlegt sind.

Datentyp: VT\_ARRAY

Zugriffsrecht: nur lesbar

OLE-Datentyp: VT\_U I1

## Syntax

```
DP2 : [<ProjektierterCPName>] Slave<Adresse>SetSlaveAddress
```

### Erklärung

#### DPMCL2-Items des Busteilnehmers - Diagnose-Items Slave

##### **SetSlaveAddress**

Setzt eine neue PROFIBUS-Adresse für den Slave

Datentyp: VT\_ARRAY

Zugriffsrecht: nur Schreiben

OLE-Datentyp: VT\_VARIANT

Beim Schreiben des Items wird der entsprechende DPMCL2-Dienst ausgeführt. Der zu schreibende Wert enthält in einem Feld die zur Ausführung des Dienstes notwendigen Parameter:

Feldelement	Datentyp	Bedeutung
1	VT_UI1	Neu zu vergebende Slave-Adresse.
2	VT_BOOL	Flag, ob die DP-Slave Adresse zu einem späteren Zeitpunkt noch einmal geändert werden kann.
3	VT_I4	Gerätetyp (PROFIBUS Ident-Nummer) des Teilnehmers.
4	VT_ARRAY   VT_UI1	Anwenderspezifische Daten. Es kann ein leeres Array übergeben werden.

Nach erfolgreichem Schreiben der neuen Slave-Adresse müssen bei Bedarf für den Slave

neue Items angelegt werden. Die Items mit der alten Slave-Adresse (z. B. DPMCL2\_E/A Items) sprechen weiter den Slave unter der alten Adresse an und können deshalb nicht mehr erfolgreich gelesen werden.

#### 2.4.2.5 Syntax der Prozessvariablen für I/O Daten

Die englischen Abkürzungen I/O (input/output) entsprechen den deutschen Abkürzungen E/A (Eingang/Ausgang).

#### Syntax

Beim Zugriff auf die Eingänge im E/A-Bereich eines Slave muss die Slave-Variable nach folgender Syntax gebildet werden:

```
DP2:[<ProjektierteterCPName>]Slave<Adresse>_E{<Format>
<Offset{.Bit}>{,Anzahl}}
```

oder (englisch):

```
DP2:[<ProjektierteterCPName>]Slave<Adresse>_I{<Format>
<Offset{.Bit}>{,Anzahl}}
```

#### Erklärung

##### DPMCL2-Items des Busteilnehmers - E Items Slave

<Adresse>\_E oder englisch <Adresse>\_I  
 <Adresse>\_A oder englisch <Adresse>\_Q

Die Eingänge und die Ausgänge können vom OPC-Client des SIMATIC NET OPC-Server nur gelesen werden.

Die durch die Projektierung vergebenen Module sowie deren Datengröße (Byte bzw. Wort) und Konsistenz werden vom DP-Master Klasse 2 nicht berücksichtigt. Er unterscheidet sich dadurch von einem DP-Master Klasse 1. Zwar kann der DP-Master Klasse 2 die Konfiguration des Slave zur Laufzeit ermitteln, der Slave und der zugehörige DP-Master Klasse 1 kann aber seine Konfiguration zur Laufzeit ändern, ohne dass der DP-Master Klasse 2 darüber benachrichtigt wird. Die Konsequenz daraus sind inkonsistente Konfigurationsdaten, selbst bei einem zeitraubenden und netzbelastenden Pollen der Konfigurationsdaten.

Der Zugriff auf einen E/A-Bereich erfolgt durch Angabe der Slave-Nummer <Adresse> und des E/A-Bereichs, wobei \_E(bzw. \_I, englisch) den Eingabebereich kennzeichnet. Falls die optionale Formatangabe weggelassen wird, werden die Daten als Feld von Bytes in der Gesamtlänge der Eingänge geliefert.

Datentyp: VT\_ARRAY

Zugriffsrecht: nur lesbar

OLE-Datentyp: VT\_UI1

**<Format>**

Das Element *Format* legt fest, in welchem Format die Daten geliefert werden. Wird keine Formatangabe verwendet, wird das Format *Byte* verwendet. Durch die Formatangabe wird auch der Datentyp festgelegt.

Formatbezeichner	Beschreibung	OLE-Datentyp	Visual Basic-Datentyp
X	Bit	VT_BOOL	Boolean
BYTE oder B	Byte (unsigned 8)	VT_UI1	Byte
CHAR	Character (signed 8)	VT_I1	Integer
WORD oder W	Wort (unsigned 16)	VT_UI2	Long
INT	Integer (signed 16)	VT_I2	Integer
DWORD oder D	Doppelwort (unsigned 32)	VT_UI4	Double
DINT	Doppel-Integer (signed 32)	VT_I4	Long
REAL	Fließkommazahl	VT_R4	Single

**<Offset{.Bit}>**

Offset in Bytes des anzusprechenden Elements (Offset-Adressierung). Die Spezifizierung eines Bits ist nur bei dem Typ X zulässig. Der anzugebende Offset erfolgt in Bytes. Beispiel: X2.3 liefert das 3. Bit des 2. Bytes.

**<Anzahl>**

Anzahl der Elemente. Der Datentyp der Variable ist ein Feld mit Elementen (Datentyp VT\_ARRAY) des angegebenen Formats. Die Angabe der Anzahl ist beim Format X nicht möglich.

Wird dieser Namensteil weggelassen oder als Anzahl 1 angegeben, wird als Anzahl 1 angenommen und der Datentyp der Variable ist kein Feld.

## Syntax

Beim Zugriff auf die Ausgänge im E/A-Bereich eines Slave muss die Slave-Variable nach folgender Syntax gebildet werden:

```
DP2:[<ProjektierterCPName>]Slave<Adresse>_A{<Format>
<Offset{.Bit}>{,Anzahl}}
```

oder (englisch):

```
DP2:[<ProjektierterCPName>]Slave<Adresse>_Q{<Format>
<Offset{.Bit}>{,Anzahl}}
```

## Erklärung

### DPMCL2-Items des Busteilnehmers - A Items Slave

Der Zugriff auf einen E/A-Bereich erfolgt durch Angabe der Slave-Nummer <Adresse> und des E/A-Bereichs, wobei *\_A* (bzw. *\_Q*, englisch) den Ausgabebereich kennzeichnet. Falls die optionale Formatangabe weggelassen wird, werden die Daten als Feld von Bytes in der Gesamtlänge der Ausgänge geliefert.

Datentyp: VT\_ARRAY

Zugriffsrecht: Read-Only

OLE-Datentyp: VT\_UI1

Die Formatierung erfolgt wie bei den E-Items des Slave.

### 2.4.2.6 Syntax der Prozessvariablen für Datensätze

#### Syntax

```
DP2 : [<ProjektierterCPName>] Slave<Adresse>S<Slot>Data<Index>
{ ,<Länge>} { ,<Teilbereich>}
mit Teilbereich = <Teilbereich> = <Format><Offset>{ .<Bit>} { ,<Anzahl>}
```

#### DP2-Items des Busteilnehmers

##### Slave

Kennzeichen für den Zugriff auf einen Slave über das DP-Protokoll.

##### Adresse

PROFIBUS Stationsadresse des Busteilnehmers. Der Busteilnehmer muss nicht notwendigerweise am PROFIBUS angeschlossen sein, damit das Item angelegt und (ggf. mit einem Zugriffsfehler) verwendet werden kann.

Bereich: 0 bis 126

##### Hinweis

Der SIMATIC NET OPC-Server kann erst durch den Erfolg eines auszuführenden Slave-Dienstes erkennen, ob der adressierte Busteilnehmer tatsächlich DP-fähig ist. Bei kritischen Busteilnehmern, die auf solche Dienste möglicherweise fehlerhaft reagieren, liegt deshalb die Erzeugung entsprechender Items in der Verantwortung des Anwenders.

#### Erklärung

#### DPC2-Items des Busteilnehmers - DPC2-Read/Write

##### S

Kennzeichen für den Slot des Slaves; typischerweise ein Modul.

##### <Slot>

Slot im erweiterten Speicherbereich eines Slave. Slot und Index kennzeichnen einen Datensatz.

Bereich: 0 bis 255 (an der C-Schnittstelle: 0 bis 255)

##### Data

Kennzeichen für einen Datensatz.

##### <Index>

Index der Daten im Slave-Slot.

Bereich: 0 bis 255

##### <Länge>

Länge des Datensatzes.

Bereich: 1 bis 240

**<Teilbereich>**

Kennzeichen für einen Teilbereich.

**<Format>**

Das Element *Format* legt fest, in welchem Format die Daten geliefert werden. Durch die Formatangabe wird auch der Datentyp festgelegt.

Formatbezeichner	Beschreibung	OLE-Datentyp	Visual Basic-Datentyp
X	Bit	VT_BOOL	Boolean
BYTE oder B	Byte (unsigned 8)	VT_UI1	Byte
CHAR	Character (signed 8)	VT_I1	Integer
WORD oder W	Wort (unsigned 16)	VT_UI2	Long
INT	Integer (signed 16)	VT_I2	Integer
DWORD oder D	Doppelwort (unsigned 32)	VT_UI4	Double
DINT	Doppel-Integer (signed 32)	VT_I4	Long
REAL	Fließkommazahl	VT_R4	Single

**<Offset{.Bit}>**

Offset in Bytes des anzusprechenden Elements. Die Spezifizierung eines Bits ist nur bei dem Typ X zulässig.

Beispiel: X2.3 liefert das 3. Bit des 2. Bytes.

**<Anzahl>**

Anzahl der Elemente (bei Format X nicht zugelassen). Der Datentyp der Variable ist ein Feld mit Elementen (Datentyp VT\_ARRAY) des angegebenen Formats. Wird dieser Namensteil weggelassen oder als Anzahl 1 angegeben, wird als Anzahl 1 angenommen und der Datentyp der Variable ist kein Feld.

**Wert des Datensatzes**

Rückgabewert des Datensatzitems.

Der Datentyp ist VT\_ARRAY | VT\_UI1, falls kein Teilbereich angeben wird.

Lesbar; falls die Datensatzlänge angegeben wird, auch schreibbar.

---

**Hinweis**

Die durch die Parameter Anzahl und Format bestimmte Datenlänge darf die Größe des Datensatzes im Slave nicht überschreiten. Die Größe eines Datensatzes ist slave-abhängig und kann vom OPC-Server nicht kontrolliert werden.

Beim Lesezugriff auf einen Teilbereich des Datensatzes wird vom Partnergerät zunächst immer der gesamte Datensatz gelesen und anschließend der entsprechende Teilbereich ausgewertet.

Beim Schreiben eines Teilbereichs wird ebenfalls der gesamte Datensatz an das Partnergerät weitergegeben. Werden innerhalb eines Schreibauftrags mehrere Teilbereiche des Datensatzes beschrieben, wird der Datensatz erst an das Partnergerät geschrieben, nachdem alle Teilbereiche des Datensatzes aktualisiert wurden.

Bei Verwendung des OPC-Servers ist es daher sinnvoll, alle Items mit Teizugriffen auf einen bestimmten Datensatz innerhalb einer Gruppe zusammenzufassen und die gesamte Gruppe zu schreiben.

---

---

**Hinweis****Schreiben von Datensätzen an das Partnergerät**

Vermeiden Sie beim Schreiben von Datensätzen grundsätzlich Überschneidungen und Lücken, da nicht vorhersehbar ist, welcher Wert in diesem Fall geschrieben wird.

---

**Hinweis**

Ein geschriebener Wert eines OPC-DPC2-Datensatz-Items ist über dasselbe Item oder ein Item mit gleicher Adresse, Slot und Index, nicht identisch auslesbar. DPC2-Datensätze können sich gerätespezifisch zwischen den geschriebenen und gelesenen Werten unterscheiden.

---

**Syntax**

```
DP2:[<ProjektierterCPName>]Slave<Adresse>DTS<Slot>Data
<Index>{,<Länge>}{,<Teilbereich>}
mit <Teilbereich> = <Format><Offset>{.<Bit>}{,<Anzahl>}
```

**Erklärung****DPC2-Items des Busteilnehmers - DPC2-Data-Transport**

Die DPC2-DataTransport Items haben Lese/Schreibzugriff, falls die Länge angegeben wird (sonst nur Lesezugriff). Wird kein Teilbereich vorgegeben, ist der Datentyp:

Datentyp: VT\_ARRAY

OLE-Datentyp: VT\_UI1

Dieses Item realisiert protokollseitig einen Datenaustausch. Der Lesezugriff dieser Items erfolgt lokal und liefert den mit dem letzten Schreibzugriff im Austausch erhaltenen Datensatz.

**DTS**

Kennzeichen für Data-Transport.

**<Slot>**

Slot im erweiterten Speicherbereich eines Slave. Slot und Index kennzeichnen einen Datensatz.

Bereich: 0 bis 255

**<Offset>{.<Bit>}**

Offset in Bytes des anzusprechenden Elements. Die Spezifizierung eines Bits ist nur bei dem Typ X zulässig.

Beispiel: X2.3 liefert das 3. Bit des 2. Bytes.

**<Anzahl>**

Anzahl der Elemente (bei Format X nicht zugelassen). Der Datentyp der Variable ist ein Feld mit Elementen (Datentyp: VT\_ARRAY) des angegebenen Formats. Wird dieser Namensteil weggelassen oder als Anzahl 1 angegeben, wird als Anzahl 1 angenommen und der Datentyp der Variable ist kein Feld.

**Hinweis**

Beim Lesezugriff auf einen Teilbereich des Datensatzes wird der entsprechende Teilbereich des vom Partnergerät zuletzt erhaltenen Datensatzes ausgewertet.

Beim Schreiben eines Teilbereichs wird ebenfalls der gesamte Datensatz an das Partnergerät weitergegeben. Werden innerhalb eines Schreibauftrags mehrere Teilbereiche des Datensatzes beschrieben, wird der Datensatz erst an das Partnergerät geschrieben, nachdem alle Teilbereiche des Datensatzes aktualisiert wurden.

Bei Verwendung des OPC-Servers ist es daher sinnvoll, alle Items mit Teillizenzen auf einen bestimmten Datensatz innerhalb einer Gruppe zusammenzufassen und die gesamte Gruppe zu schreiben.

**Hinweis****Schreiben von Datensätzen an das Partnergerät**

Vermeiden Sie beim Schreiben von Datensätzen grundsätzlich Überschneidungen und Lücken, da nicht vorhersehbar ist, welcher Wert in diesem Fall geschrieben wird.

**2.4.2.7****Beispiele für Prozessvariablen für Datensätze**

Hier finden Sie Beispiele, die die Syntax von Variablennamen für DPC2-Datensätze verdeutlichen.

**Variablennamen für DPC2**

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

Zugriff auf das Doppelwort ab Offset 7 in einem Datensatz der Länge 120 Bytes in Slot 3, Index 2 des Slaves 5.

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

Zugriff auf ein Feld mit 4 Bytes ab Offset 8 in einem Datensatz der Länge 120 Bytes in Slot 3, Index 2 des Slaves 5.

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

Data Transport Zugriff auf das Doppelwort ab Offset 2 in einem Datensatz der Länge 10 Bytes in Slot 6, Index 2 des Slaves 5.

### **2.4.2.8 Syntax der DP2-spezifischen Informationsvariablen**

#### **Syntax**

DP2 : [<ProjektierterCPName>] &identify()

#### **Erklärung**

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

Liefert die Teilnehmeridentifikation des angegebenen Gerätes als Feld mit 4 Strings:

Datentyp: VT\_ARRAY

Beschreibung: 4 Array-Elemente (0-3)

Zugriffsrecht: nur lesbar

OLE-Datentyp: VT\_BSTR

Elemente des Rückgabewerts:

- Hersteller
- Controller
- Hardware-Version
- Software-Version

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

##### **OLE-Datentyp Visual Basic Typ**

VT\_ARRAY | VT\_BSTR String()

#### **Syntax**

DP2 : [<ProjektierterCPName>] lifelist()

#### **Erklärung**

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

Allgemeine Items

Liefert eine Liste mit allen angeschlossenen Busteilnehmern.

Datentyp: VT\_ARRAY

Beschreibung: 127 Array-Elemente (0-126)

Zugriffsrecht: nur lesbar

OLE-Datentyp: VT\_UI1

Jedes Arrayelement steht für eine PROFIBUS Stationsadresse. Die Werte der Array-Elemente bedeuten:

Element	Bedeutung
0x00	STATION_PASSIVE
0x10	STATION_NON_EXISTANT
0x20	STATION_ACTIVE_READY (bereit zur Aufnahme in den logischen Ring)
0x30	STATION_ACTIVE (bereits im logischen Ring)

#### Hinweis

Bei Verwendung der Lifelist als OPC-Item, z. B. DP2:[CP5614A2]lifelist(), ist Folgendes zu beachten:

Der FDL-Dienst "Lifelist" ist ein Dienst, der nur von einer Anwendung aufgerufen werden kann. Wird er als OPC-Item eingestellt, so steht er für andere Anwendungen nicht mehr zu Verfügung. Insbesondere kann im Konfigurationsprogramm "Kommunikations-Einstellungen" die Lifelist nicht mehr angezeigt werden.

Da der Lifelist-Aufruf zudem eine nicht unbedeutende Buslast erzeugt, die unter OPC zyklisch auftritt, sollte er nach Möglichkeit vermieden werden.

#### 2.4.2.9 Syntax der systemspezifischen Informationsvariablen

DP2 : [SYSTEM] &version()

##### &version()

Liefert eine Versionskennung für den DP2-OPC-Server, hier z.B. die Zeichenfolge  
*SIMATIC NET Core Server DP2 V 7.xxxx.yyyy.zzzz Copyright 2012*

Datentyp: VT\_BSTR

Zugriffsrecht: nur lesbar

#### 2.4.3 DP-Slave

#### Hinweis

Ein PROFIBUS-Kommunikationsprozessor als DP-Slave kann über OPC nur als DP-V0-Slave betrieben werden. Mit den genannten HARDNET-Baugruppen können somit über OPC keine Datensätze gelesen oder geschrieben werden.

### 2.4.3.1 Variabldienste zum Zugriff auf lokale Slave-Daten

Der OPC-Server für SIMATIC NET kann in einem PROFIBUS DP Netz neben der DP-Master-Funktion auch die Rolle des DP-Slaves übernehmen. Der OPC-Server verwaltet Speicherbereiche für die Ein- und Ausgänge dieses DP-Slaves und bildet diese auf OPC-Variablen ab. Ein OPC-Client kann die vom Master gesetzten Ausgänge lesen und in den Eingängen Werte setzen, die der DP-Master im nächsten Zyklus abholt.

DP-Slaves sind modular aufgebaut. Ein DP-Slave kann mehrere Module mit unterschiedlichen Ein-/Ausgangsbereichen enthalten. Die Zuordnung der Module geschieht durch die DP-Projektierung.

Der Variablenname bezeichnet einen Ein- oder Ausgangsbereich im Modul eines Slaves. Der Zugriff auf die Ein- und Ausgänge des Slaves erfolgt durch Angabe der Modul-Nummer und des Ein- oder Ausgangsbereichs.

Über Variablennamen können Statusinformation der Slaves und des DP-Masters abgefragt werden.

#### Hinweis

Der parallele Betrieb von DP-Master und DP-Slave mit der Baugruppe CP 5614, CP 5614 A2 oder CP 5614 FO ist nur möglich, wenn bei der Projektierung die Betriebsart "DP-Base" eingestellt wurde. Alternativ dazu können Sie auch den CP 5624 verwenden.

### 2.4.3.2 Syntax der Prozessvariablen für den DP-Slave

#### Syntax

##### Eingänge:

```
DP: [<Verbindungsname>] Slave{M<Nummer>}_E{<Format><Offset>
{.<Bit>}[,<Anzahl>]}
```

##### Ausgänge:

```
DP: [<Verbindungsname>] Slave{M<Nummer>}_A{<Format><Offset>
{.<Bit>}[,<Anzahl>]}
```

#### Erklärungen

##### DP

Protokoll für den Zugriff auf die Prozessvariable.

##### <Verbindungsname>

In der Projektierung angegebener Name der Kommunikationsbaugruppe.

##### Slave

Kennzeichen für den Zugriff auf einen Slave über das DP-Protokoll.

##### M

Kennzeichen für die Nummer des Moduls.

##### <Nummer>

Nummer des Moduls, das den Ein- bzw. Ausgangsbereich enthält.

**\_E**

Kennzeichen für einen Eingang.

**\_A**

Kennzeichen für einen Ausgang.

**<Format>**

Format, in dem die Daten geliefert werden.

Durch die Formatangabe wird auch der Datentyp festgelegt.

Über die Automation-Schnittstelle von OPC können alle angegebenen OLE-Datentypen gelesen werden. Jedoch bieten einige Entwicklungswerkzeuge (wie z.B. Visual Basic) nur eine eingeschränkte Menge von Datentypen an. Die folgende Tabelle listet deshalb den entsprechenden Visual Basic-Typ auf, in dem der Variablenwert dargestellt werden kann.

Formatbezeichner	Beschreibung	OLE-Datentyp	Visual Basic-Typ
X	Bit	VT_BOOL	Boolean
BYTE oder B	Byte(unsigned 8)	VT_UI1	Byte
CHAR	Character (signed 8)	VT_I1	Integer
WORD oder W	Wort (unsigned 16)	VT_UI2	Long
INT	Integer (signed 16)	VT_I2	Integer
DWORD oder D	Doppelwort (unsigned 32)	VT_UI4	Double
DINT	Doppel-Integer (signed 32)	VT_I4	Long
REAL	Fließkommazahl	VT_R4	Single

**<Offset>**

Byteadresse im Adressraum des Slaves, an der das Element liegt, das angesprochen werden soll.

**<Bit>**

Bitnummer in dem adressierten Byte. Der Bereich liegt zwischen 0 und 7.

Die Spezifizierung eines Bits ist nur beim Formatbezeichner X zulässig.

**<Anzahl>**

Anzahl der Elemente.

Der Datentyp (VT-ARRAY) der Variable ist ein Feld mit Elementen des angegebenen Formats.

Wird Anzahl weggelassen, wird als Anzahl 1 angenommen und der Datentyp der Variable ist kein Feld.

Den Parameter Anzahl dürfen Sie nicht beim Formatbezeichner X verwenden.

**2.4.3.3****Beispiele für Prozessvariablen für den DP-Slave**

Hier finden Sie einige Beispiele, die die Syntax von Variablennamen für DP-Slave-Variablen verdeutlichen.

**Eingänge**

DP:[CP 5611]SlaveM003\_EB0  
SlaveM003\_EB0

Eingangsbyte 0 (Offset 0) im Modul 3 des DP-Slaves.

*DP:[CP 5611]SlaveM003\_EB1,3*

SlaveM003\_EB1,3

Ein Feld mit 3 Bytes ab Eingangsbyte1 (Offset 1) im Modul 3 des DP-Slaves.

*DP:[CP 5624]SlaveM003\_EX0.0*

SlaveM003\_EX0.0

Eingangsbit 0 im Byte 0 im Modul 3 des DP-Slaves

*DP:[CP 5614]SlaveM003\_EB3,8*

SlaveM003\_EB3,8

Ein Feld mit 8 Eingangsbytes ab Offset 3 im Modul 3 des DP-Slaves.

## Ausgänge

*DP:[CP 5611]SlaveM003\_AW3*

SlaveM004\_AW3

Ein Ausgangswort an Adresse 3 im Modul 4 des DP-Slave.

*DP:[CP 5611]SlaveM003\_ADWORD2*

SlaveM003\_ADWORD2

Ein Ausgangsdoppelwort an Adresse 2 im Modul 3 des DP-Slave.

*DP:[CP 5624]SlaveM003\_AX3.7*

Slave\_AX3.7

Ein Ausgangsbit 7 im Byte 3 des DP-Slave.

*DP:[CP 5624]SlaveM001\_AW0,4*

SlaveM001\_AW0,4

Ein Feld mit 4 Ausgangsworten im Modul 1 des DP-Slave.

### 2.4.3.4 DP-Slave-spezifische Informationsvariablen

Für Diagnosezwecke beim DP-Slave gibt es einige vordefinierte Informationsvariablen.

### 2.4.3.5 Syntax der DP-Slave-spezifischen Informationsvariablen

## Syntax

Es gibt zwei Möglichkeiten:

DP: [<Verbindungsname>]<Diagnose-Item>

DP: [<Verbindungsname>]<Parameter-Item>

## Erklärungen

**DP**

Protokoll für den Zugriff auf die Prozessvariable.

**<Verbindungsname>**

Name, der in der Projektierung für die Kommunikationsbaugruppe festgelegt wurde.

**<Diagnose-Item>**

Vordefiniertes Item.

Es gibt folgende Möglichkeit:

**devicestate**

Zustand der Baugruppe, auf dem der DP-Slave abläuft.

Es gibt folgende Zustände:

- ONLINE
- OFFLINE

**<Parameter-Item>**

Vordefinierte Parameter-Items.

Es gibt folgende Möglichkeiten:

- *SlaveMiscReadSlvParCfgData*  
Konfigurationsdaten des Slaves
- *SlaveSlvState*  
Zustand des Slaves.

Es gibt folgende Zustände:

- DATA\_EXCHANGE
- NO\_DATA\_EXCHANGE

## 2.5 PROFIBUS-DP mit OPC UA

### Prozessvariablen für den DP-Master mit OPC UA

Der OPC-Server von SIMATIC NET für den DP-Master-Betrieb über OPC UA bietet für folgende Dienste Prozessvariablen an:

- Dienste für den Master Klasse 1  
Zugriff und Beobachtung von DP-Eingängen und Ausgängen
- Sync / Freeze  
Azyklisches Senden von Steuertelegrammen an Slave-Gruppen
- Fast Logic für
  - CP 5613 A2 und CP 5614 A2 (nur DP-Master)  
Automatische Überwachung von Slave-Daten
  - CP 5623 und CP5624 (nur DP-Master)  
Automatische Überwachung von Slave-Daten
- Diagnosevariablen  
Auswertung der statischen Diagnose

## **Prozessvariablen für den DP-Slave**

Der OPC-Server von SIMATIC NET für den DP-Slave-Betrieb über OPC UA bietet für folgende Dienste Prozessvariablen an:

- Variabldienste zum Zugriff auf lokale Slave-Daten  
Zugriff auf die Ein- und Ausgänge des Slave
- Diagnosevariablen  
Auswertung der statischen Diagnose des Slave

### **2.5.1           SIMATIC NET OPC-UA-Server für das DP-Protokoll**

#### **Einleitung**

Der folgende Abschnitt beschreibt eine Konfigurationsvariante für das DP-Protokoll, die parallel zu DP-COM-OPC-Data-Access-Server auch OPC UA unterstützt. Hierfür muss der DP-COM-OPC-Data-Access-Server als Outproc-OPC-Server eingerichtet werden.

Da das PROFIBUS DP-Protokoll ein Abbild der Ein- und Ausgangsdaten im DP-RAM des im PC gesteckten Kommunikationsprozessors bereithält, erfolgen die Zugriffe auf die Prozessdaten lokal innerhalb des PC. Damit können die Zugriffe besonders bei Verwendung der SIMATIC NET Baugruppen CP 5613 A2, CP 5613 A3, CP 5614 A2, CP 5614 A3, CP 5623 und CP 5624 über die Schnittstelle DP-Base sehr schnell erfolgen.

In einigen Anwendungsfällen, z.B. bei Verwendung von PC-basierten Steuerungen, sind sehr kurze Zeiten für den Zugriff auf Prozessdaten erforderlich.

## Konfiguration

Die Aktivierung des DP-OPC-UA-Servers erfolgt durch die Auswahl von "DP" und "OPC UA" im Konfigurationsprogramm "Kommunikations-Einstellungen" bei "OPC-Protokollauswahl":

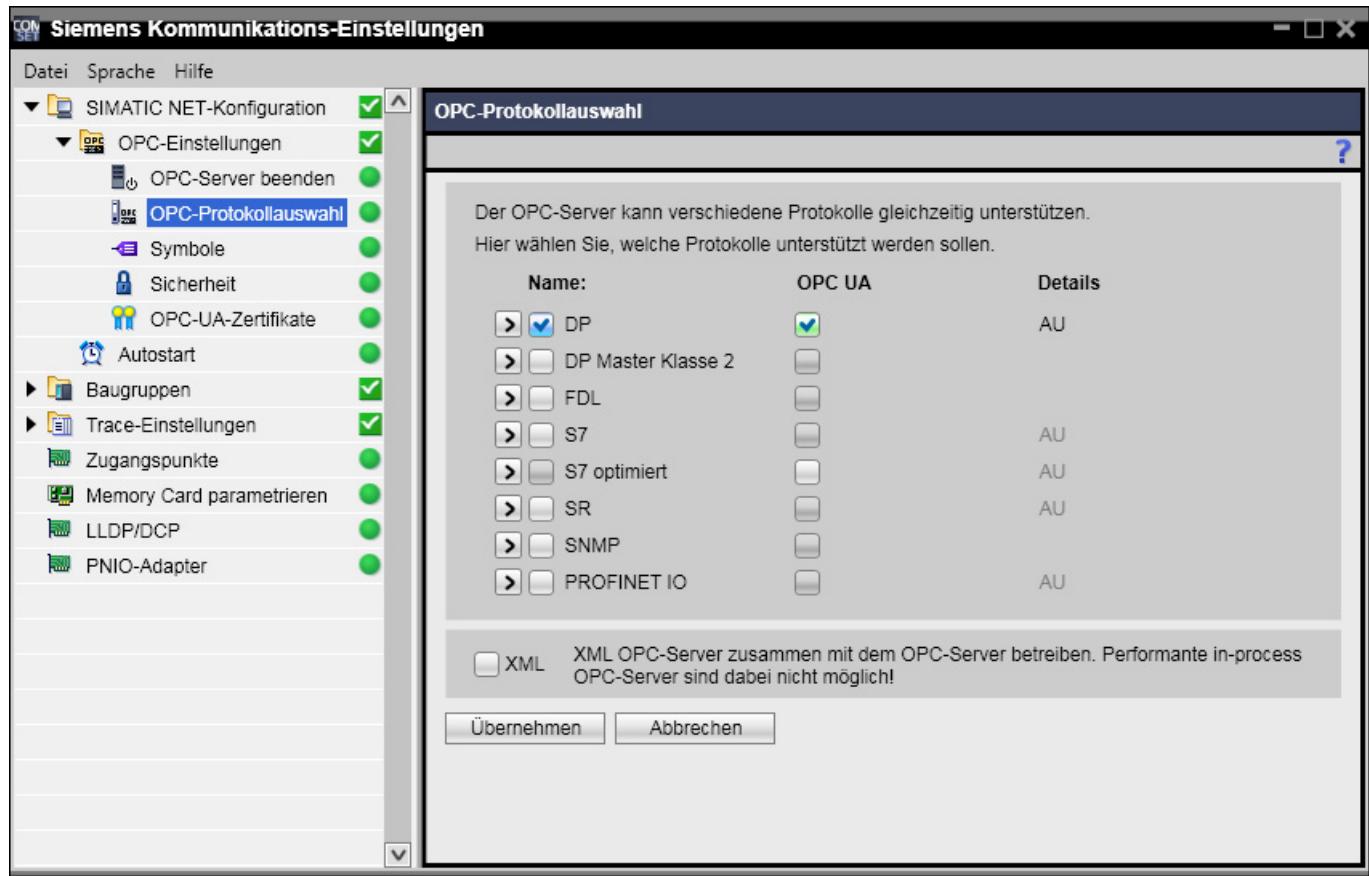


Bild 2-3 Fenster des Konfigurationsprogramms "Kommunikations-Einstellungen" zur Auswahl von OPC UA für das DP-Protokoll

## Vorteile / Nachteile

Bei Verwendung des DP-OPC-UA-Servers ist nur der Outproc-Betrieb des DP-OPC-Servers möglich. Der DP-OPC-UA-Server-Prozess muss zur Aufrechterhaltung der UA-Empfangsbereitschaft gestartet worden sein. Ein Beenden des DP-OPC-UA-Servers, auch nach Abmeldung aller OPC-UA-Clients, wird nicht ausgeführt. Wenn der DP-OPC-UA-Server-Prozess gestoppt wird, so ist die UA-Funktionalität nicht mehr verfügbar.

Gegenüber dem DP-COM-Server bestehen folgende Vorteile:

- Es ist keine COM/DCOM-Konfiguration mehr nötig.
- Performante, sichere Kommunikation

## 2.5.2 Unterstützung der DP-Dienste unter OPC UA

### DP-Master Klasse 1

Der DP-OPC-UA-Server unterstützt DP-Master Klasse 1. Der DP-Master Klasse 1 führt die zyklische Kommunikation zu den DP-Slaves aus. Die Kommunikation enthält zentrale Funktionen wie:

- Parametrierung und Konfigurierung der Slaves
- zyklischer Datentransfer zu den DP-Slaves
- Überwachen der DP-Slaves
- Bereitstellen von Diagnoseinformationen

Durch DPC1 kann ein zyklisch arbeitender Master zusätzlichen, azyklischen Datenverkehr durchführen.

### Übersicht des DP-Master Klasse 1 unter OPC UA

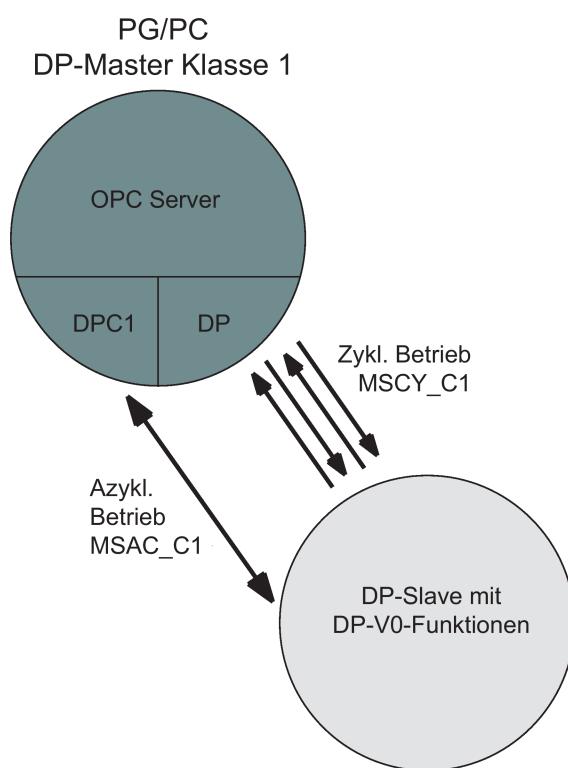


Bild 2-4 Die Protokollteile von DP-Master Klasse 1

MSCY_C1	Master-Slave, zyklischer C1-Betrieb
MSAC_C1	Master-Slave, azyklischer C1-Betrieb

## DP-Slave (DP-V0)

Der DP-OPC-UA-Server unterstützt die DP-Slave-Funktion DP-V0. Die Kommunikation enthält zentrale Funktionen wie:

- Parametrierung und Konfigurierung durch den DP-Master Klasse 1
- zyklischer Datentransfer zu den DP-Master Klasse 1
- Überwachen durch den DP-Master Klasse 1
- Bereitstellen von Diagnoseinformationen durch den DP-Master Klasse 1

Durch DPC1 kann ein zyklisch arbeitender Master zusätzlichen, azyklischen Datenverkehr durchführen.

## Übersicht des DP-OPC-UA-Slave (V0)

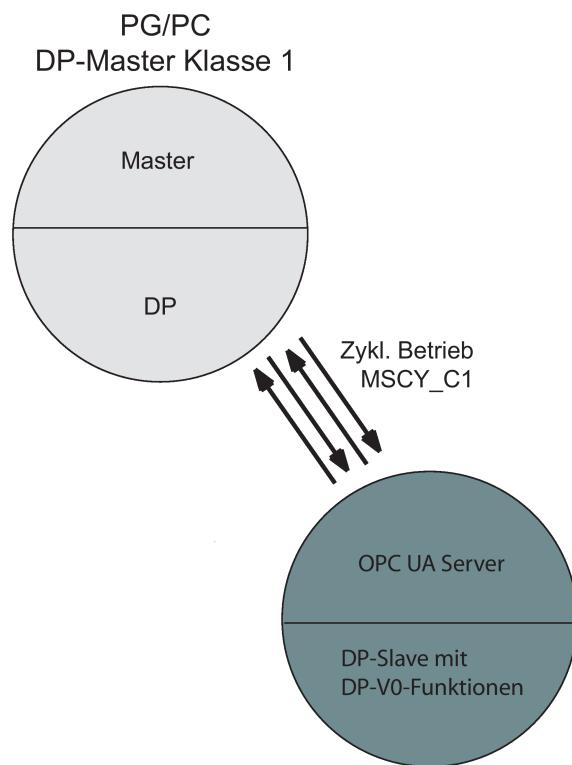


Bild 2-5 Die Protokollteile von DP-Slave (V0)

MSCY_C1	Master-Slave, zyklischer C1-Betrieb
MSAC_C1	Master-Slave, azyklischer C1-Betrieb

## **2.5.3 Wie wird der DP-OPC-UA-Server adressiert?**

### **Server-URL**

Für das native binäre TCP-Protokoll gibt es für den OPC-Client zwei Möglichkeiten der Server-Adressierung:

- Direkte Adressierung:
  - opc.tcp://<hostname>:55103
    - oder
  - opc.tcp://<IP-Adresse>:55103
    - oder
  - opc.tcp://localhost:55103

Der DP-OPC-UA-Server stellt den Port 55103.

- Der Discovery-Dienst ist ein Hilfsmittel zur Auflistung aller verfügbaren Server-URL. Verwenden Sie hierzu folgende Server-URL:
  - opc.tcp://<hostname>:4840
    - oder
  - opc.tcp://<IP-Adresse>:4840
    - oder
  - http://<hostname>:52601/UADiscovery/
    - oder
  - http://<IP-Adresse>:52601/UADiscovery/

Der Discovery-Server hat den Port 4840 (für TCP-Verbindungen) und den Port 52601 (für HTTP-Verbindungen).

### **IPv6-Adresse**

Zwischen der OPC-Client-Anwendung und dem OPC-Server kann der Datenaustausch auch über IPv6 Adressierung realisiert werden. Die Adresse muss in Klammern angegeben werden, z.B. [fe80:e499:b710:5975:73d8:14]

### **Endpunkte und Sicherheitsmodi**

Der SIMATIC NET DP-OPC-UA-Server unterstützt eine durch Verschlüsselung und Signierung gesicherte Kommunikation über das TCP-Protokoll. Diese sicheren Modi sind im Produktivbetrieb vorzuziehen.

Der Discovery-Dienst auf dem angesprochenen Host meldet die Endpunkte der Server, sowie deren Sicherheitsanforderungen und Protokollunterstützung.

Die Server-URL "opc.tcp://<hostname>:55103" des DP-OPC-UA-Servers bietet folgende Endpunkte:

- Endpunkt im Sicherheitsmodus "SignAndEncrypt":

Zur Kommunikation mit dem Server werden Signierung und Verschlüsselung gefordert. Die Kommunikation ist durch Zertifikataustausch und Passworteingabe geschützt.

Zusätzlich zum Sicherheitsmodus wird die Sicherheitsrichtlinie Basic128Rsa15 angezeigt.

- Endpunkt im Sicherheitsmodus "None":

In diesem Modus werden keine Sicherheitsfunktionen vom Server gefordert (Sicherheitsrichtlinie "None") und er ist für Test und Inbetriebnahme geeignet.

Weitere Details zu den Sicherheitsfunktionen finden Sie im Kapitel "OPC-UA-Schnittstelle programmieren (Seite 508)".

Die Sicherheitsrichtlinien "Basic128Rsa18" und "None" finden Sie in der UA-Spezifikation der OPC Foundation unter folgender Internet-Adresse:

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

Weitere Informationen finden Sie auf folgender Internetseite:

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

## Die OPC-UA-Discovery des OPC Scout V10

Der OPC Scout V10 ermöglicht die Nutzung des Discovery-Dienstes zur Übernahme von UA-Endpunkten (Server-URL) in den Navigationsbereich des OPC Scout V10.

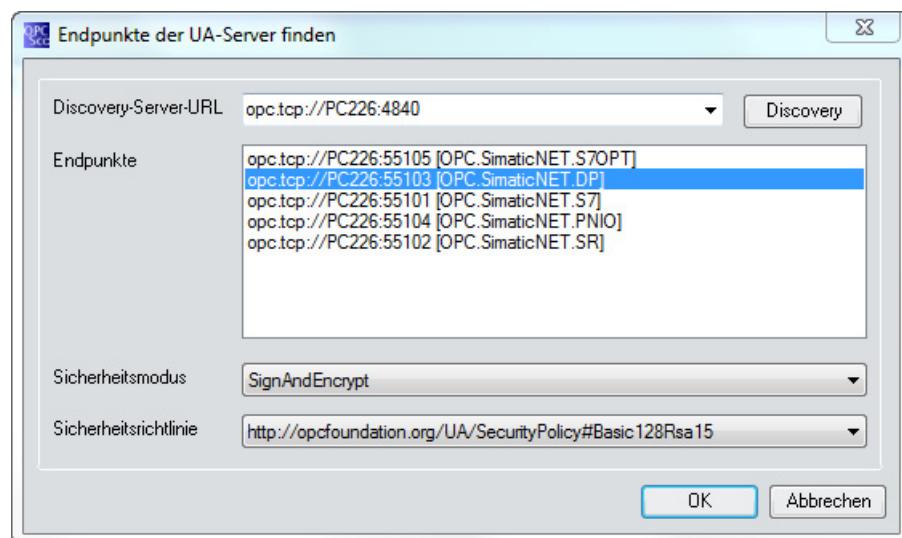


Bild 2-6 Das Dialogfeld "Endpunkte der UA-Server finden" des OPC Scout V10

Alle verfügbaren OPC-UA-Server (darunter auch der DP-OPC-UA-Server) können über den OPC-UA-Discovery-Dienst aufgelistet werden.

Der OPC Scout V10 kennt alle von SIMATIC NET unterstützten OPC-UA-Endpunkte. Der Discovery-Dienst auf dem angesprochenen Host meldet für diese Endpunkte die registrierten DP-OPC-UA-Server sowie deren Ports und Sicherheitsmodi.

Weitere Details finden Sie in der Online-Hilfe des OPC Scout V10.

## 2.5.4 Welche Namensräume bietet der DP-OPC-UA-Server an?

**Der DP-OPC-UA-Server bietet folgende Namensräume an:**

Tabelle 2- 1 Namensräume von DP-OPC-UA:

Namensraum-Index	"Bezeichner" (Namensraum-URI) / Kommentar
0	"http://opcfoundation.org/UA/" von der OPC Foundation spezifiziert
1	"urn:Siemens.Automation.SimaticNET.DP:(GUID)" Eindeutiger Bezeichner des DP-OPC-UA-Server.
2	"DPTYPES:" Definitionen für DP-spezifische Objekttypen.
3	"DP:" Bezeichner des DP-OPC-UA-Servers mit neuer vereinfachter Syntax (durchsuchbar und verwendbar mit OPC UA)
4	"DPCOM:" Bezeichner des Servers mit alter Syntax, DP-OPC-DA-kompatibel (mit OPC UA verwendbar, aber nicht durchsuchbar)

Die Namensraum-Indizes 0 und 1 sind reserviert und in Ihrer Bedeutung von der OPC Foundation spezifiziert.

Die Zuordnung der restlichen Namensraum-Indizes zu den Bezeichnern (Namenraum-URI) muss zu Beginn einer OPC-UA-Session vom Client unter Angabe des Bezeichners über die Datenvariable "NamespaceArray" ermittelt werden. Die Bezeichner "DPTYPES:", "DP:" und "DPCOM:" sind beim DP-OPC-UA-Server immer vorhanden.

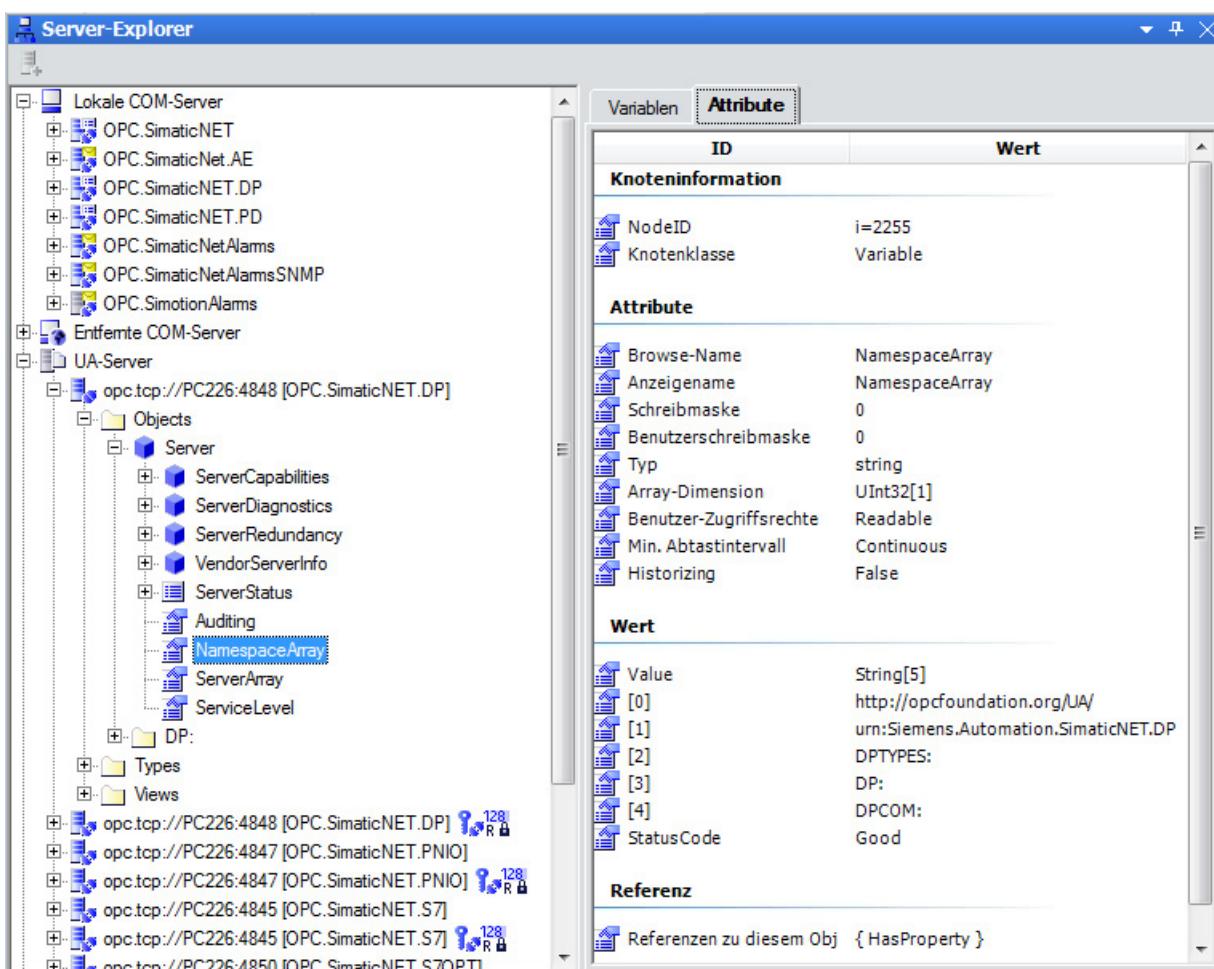


Bild 2-7 Anzeige der DP-OPC-UA-Namensräume mittels der Browse-Funktion des OPC Scout V10

## 2.5.5 Die Nodeld

### Identifikation einer DP-Prozessvariable

Die Nodeld identifiziert mit Hilfe des folgenden Tupels zur Laufzeit eine DP-Prozessvariable eindeutig:

- Namensraum-Index
- Bezeichner (Zeichenfolge, numerischer Wert)

## Beispiele

- Nodeld:
  - Namensraum-URI:  
 $DP:$   
 (= Namensraum-Index 3) für Siemens.Automation.SimaticNET.DP
  - Bezeichner:  
 $DPMasterName.slv17.q0.244$
  
- Nodeld:
  - Namensraum-URI:  
 $DPCOM:$   
 (= Namensraum-Index 4) für OPC.SimaticNET; die Syntax ist DP-OPC-DA-kompatibel
  - Bezeichner:  
 $DPMasterName.slv17.q0.244$

## Wie verhält sich der neue auf OPC UA angepasste Namensraum?

Die Welt der OPC-Data-Access-Items eines COM-Servers ist zum Lesen und Schreiben von Prozessvariablen in sich abgeschlossen. Daneben existiert unabhängig davon die Alarmwelt.

Dagegen ist die OPC-UA-Sicht auf Automatisierungsobjekte auch auf verschiedene Eigenschaften der Objekte bezogen. OPC UA greift nicht mehr alleine auf Items zu, sondern auf Objekte und deren Unterobjekte.

- Slavevariablen und Methoden sind beispielsweise Unterobjekte eines DP-Master-Objekts. Attribute und Properties definieren die Objekte näher.
- Ein OPC-Data-Access-Item für den Slave-Zugriff entspricht dabei einer OPC-UA-Datenvariablen.
- Ein OPC-Data-Access-Item für Fast Logic entspricht einer OPC-UA-Methode.

Den qualifizierten Bezeichnern der Nodelds kommt unter OPC UA eine größere Bedeutung als unter OPC Data Access zu. Jeder einzelne Zugriff auf ein Objekt, Unterobjekt, Property und Attribut erfolgt über dessen Nodeld.

Unter anderem für die Unterstützung durch lokale Sprachen, sieht OPC UA den Anzeigennamen vor. So kann ein und dasselbe Objekt beispielsweise in unterschiedlichen Sprachumgebungen, die der OPC-UA-Client vorgibt, unterschiedlich durchsucht werden, wobei aber jedes Mal die selbe Nodeld präsentiert wird. Der Anzeigename wird analog zur jeweiligen Nodeld gewählt. Die Texte des gesamten Namensraums sind in Englisch.

## Syntax der DP-OPC-UA-Datenobjekte

OPC UA definiert eine optimierte Syntax für den Zugriff auf die einzelnen Objekte. Die Nodelds aller OPC-UA-Objekte haben folgenden Aufbau:

$<boardobjekt>. <objekt>. <unterobjekt>. <property>$

Ein Unterobjekt kann weitere Unterobjekte beinhalten.

Der Zugriff auf eine nicht interpretierbare Nodeld wird mit einem Fehler zurückgewiesen. Die Groß- oder Kleinschreibung der Buchstaben "A-Z" wird bei allen Items ignoriert.

## Symbolische Objektdarstellung

Die OPC-UA-Spezifikation empfiehlt zur hierarchischen Beschreibung des Adressraums eine einheitliche Symboldarstellung. Folgende Symbole werden in diesem Dokument verwendet:

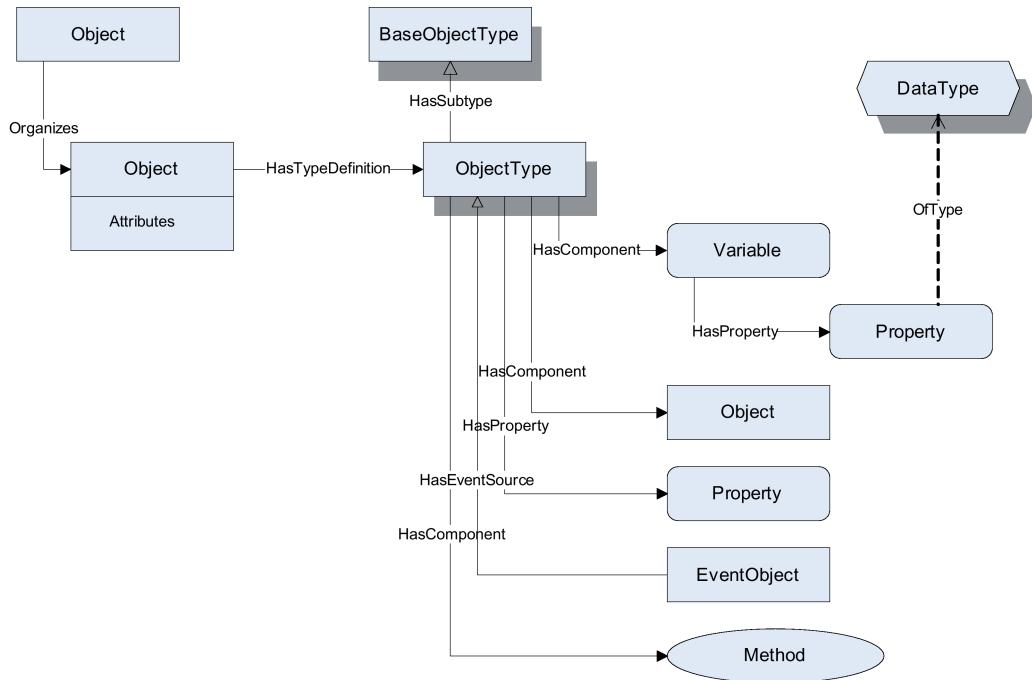


Bild 2-8 Symbole des OPC-UA-Adressraums

## 2.5.6 Board-Objekte für DP-Baugruppen

### 2.5.6.1 Übersicht der Board-Objekte für DP-Baugruppen

#### DP-Board-Typen

Alle protokollspezifischen Objekte sind immer einem Board zugeordnet, bei DP ist dies normalerweise die DP-Master-Baugruppe oder die DP-Slave-Baugruppe. Ausnahmen hiervon bildet das sogenannte Systemboard und die DP-Demo-Baugruppe.

- **DP-Master-Baugruppe**  
DP-Master-Baugruppen werden zum Datenaustausch mit DP-Slave-Baugruppen genutzt und im Allgemeinen über STEP 7 projektiert.
- **DP-Slave-Baugruppe**  
DP-Slave-Baugruppen werden zum Datenaustausch mit DP-Master-Baugruppen genutzt und im Allgemeinen über STEP 7 projektiert.
- **Die Demo-Baugruppe**  
Unter der Demo-Baugruppe mit dem Namen "DEMO" wird ein DP-Master mit mehreren einfachen Slaves simuliert.

## Demo-Baugruppe

<Boardobjekt>:= "DEMO"

Unter der Demo-Baugruppe mit dem Namen "DEMO" gibt es Objekte, die einen ähnlichen Namensraum wie DP-Master-Baugruppen beinhalten. Die Demo-Baugruppe ist zum Kennenlernen des SIMATIC NET OPC-Systems gedacht und kann über die Konfiguration hinzugeschaltet werden.

<Boardobjekt>:= "DEMO\_S"

Unter der Demo-Baugruppe mit dem Namen "DEMO\_S" gibt es Objekte, die einen ähnlichen Namensraum wie DP-Slave-Baugruppen beinhalten. Die Demo-Baugruppe ist zum Kennenlernen des SIMATIC NET OPC-Systems gedacht und kann über die Konfiguration hinzugeschaltet werden.

---

### Hinweis

Die Demo-Baugruppen mit den Namen "DEMO" und "DEMO\_S" dürfen nicht gleichzeitig mit einer DP-Master-Baugruppe oder DP-Slave-Baugruppe gleichen Namens eingesetzt werden. Eine mit diesem Namen projektierte DP-Master-Baugruppe oder DP-Slave-Baugruppe wird ignoriert, wenn eine Demo-Baugruppe hinzukonfiguriert wurde.

---

## Was sind DP-Boardobjekte?

Alle produktiven protokollspezifischen Objekte sind immer einem Board zugeordnet. Bei DP sind dies die DP-Master Baugruppen bzw. DP-Slave Baugruppen (Board). Ausnahmen hiervon bilden das Demo-Board und das DEMO\_S-Board.

### 2.5.6.2      Board-Namen

#### Der Board-Name einer DP-Baugruppe

Der Board-Name ist der in STEP 7 oder dem TIA Portal projektierte DP-Name zur Identifikation der Baugruppe. Dieser Name heißt bei STEP 7 "Lokale ID". Die Lokale ID ist innerhalb des OPC-Servers eindeutig.

#### Board-Typen

Der OPC-Server unterstützt folgende Boardtypen:

- DP-Master-Baugruppe
- DP-Slave-Baugruppe

## Welche Zeichen sind für DP-Board-Namen erlaubt?

Für <Boardname> sind Ziffern "0-9", alphabetische Zeichen in Groß- und Kleinschreibung "A-z" und Sonderzeichen " \_+()" erlaubt. Der Board-Name darf 24 Zeichen lang sein. Groß- und Kleinschreibung wird nicht unterschieden.

---

### Hinweis

Bei SIMATIC NET werden typischerweise die Baugruppennamen verwendet wie unten im Beispiel erwähnt.

---

Die Board-Namen "SYSTEM", "DEMO" und "DEMO\_S" sind reserviert und dürfen nicht verwendet werden.

## Beispiele für Board-Namen

Typische Beispiele sind:

- CP5613A2
- CP5624Slave
- DPMaster

### 2.5.6.3 Typ-Definition des DP-Master Boardobjekts

#### Typ-Definition des DP-Master Boardobjekts

Für die Objekte und Funktionalitäten, die über ein produktives DP-Master Board verwendbar sind, ist ein spezifischer OPC-UA-Objekttyp definiert:

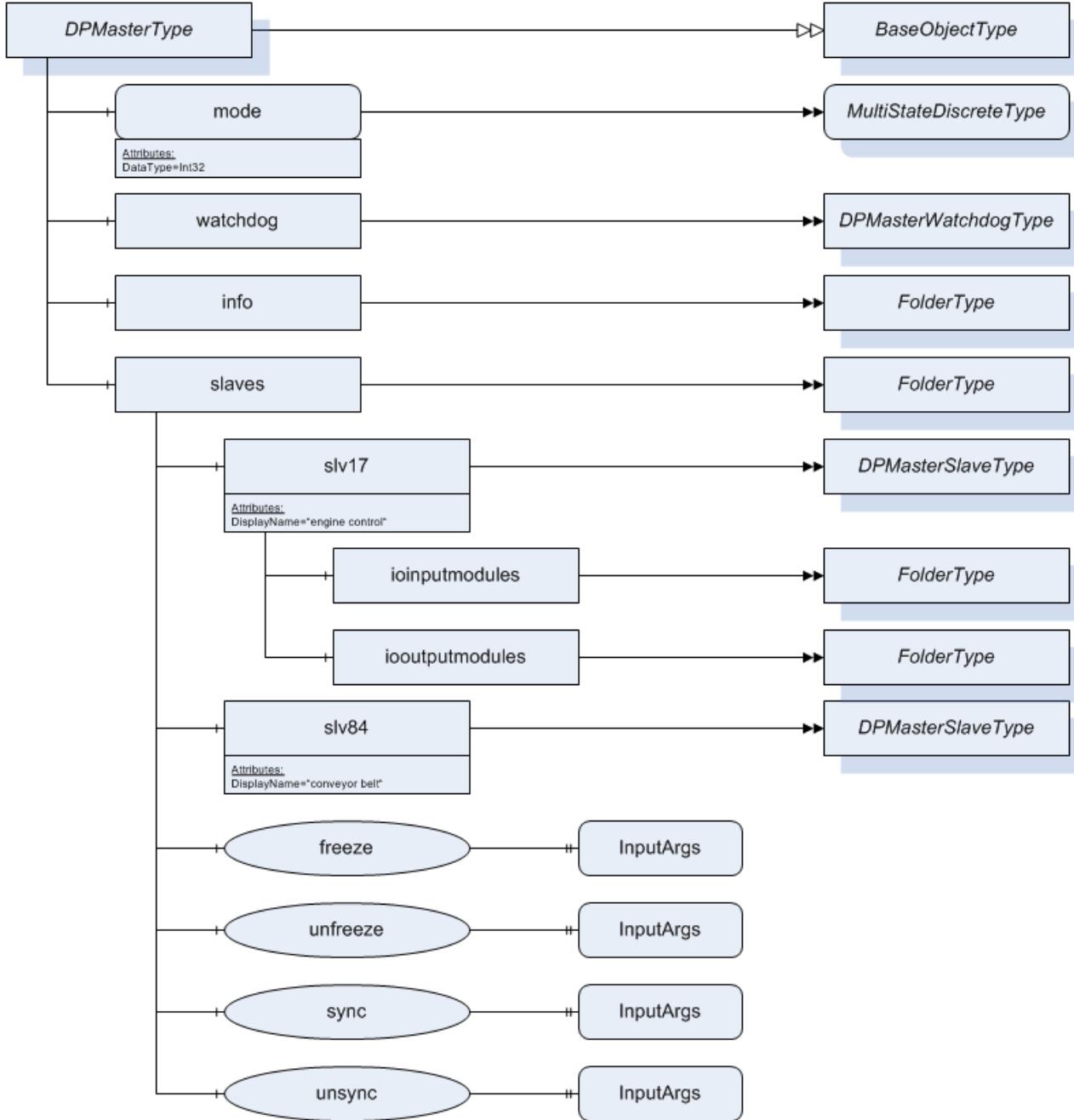


Bild 2-9 Der Typ des DP-Master Boardobjekts im Namensraum von OPC UA

Im OPC-UA-Namensraum für Objekte werden Instanzen dieses Typs angezeigt. Der Typ selbst kann unter "Typen" strukturiert ausgelesen werden.

#### 2.5.6.4 Typ-Definition des DP-Slave Boardobjekts

##### Typ-Definition des DP-Slave Boardobjekts

Für die Objekte und Funktionalitäten, die über ein produktives DP-Slave Board verwendbar sind, ist ein spezifischer OPC-UA-Objekttyp definiert:

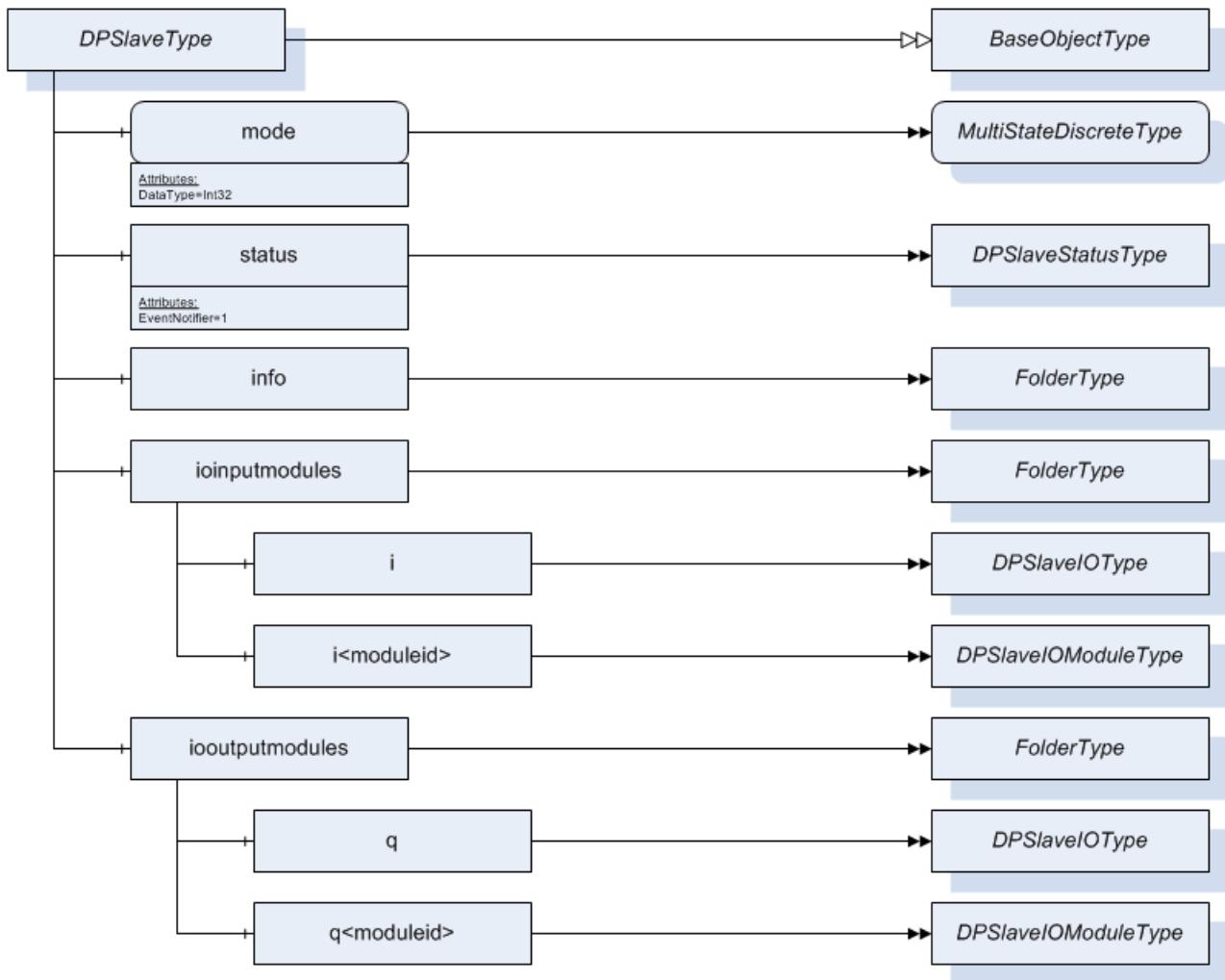


Bild 2-10 Der Typ des DP-Slave Boardobjekts im Namensraum von OPC UA

Im OPC-UA-Namensraum für Objekte werden Instanzen dieses Typs angezeigt. Der Typ selbst kann unter "Typen" strukturiert ausgelesen werden.

## **2.5.7 DP Master Klasse 1**

### **2.5.7.1 Prozessvariablen für Dienste des Master Klasse 1**

Mit den Diensten zum Zugriff auf zyklische Daten können Sie auf die Eingänge und Ausgänge der Slaves zugreifen und diese beobachten und steuern.

Der Zugriff erfolgt über:

- Slave-Nummer  
Diese Nummer entspricht der PROFIBUS-Adresse.
- Modulnummer  
DP-Slaves können mehrere Module mit unterschiedlichen Ein-/Ausgangsbereichen enthalten.
- Ein-/Ausgangsbereich

### **2.5.7.2 Syntax der Prozessvariablen für den Master Klasse 1**

#### **Syntax der Prozessvariablen**

Optimierte Syntax der Prozessvariablen der DP-OPC-UA-Node-ID, für das zyklische Lesen und Schreiben von IO-Datenvariablen:

Namensraum-URI: DP: (Namensraum-Index: 3)

#### **Syntax**

Es gibt folgende Möglichkeiten:

- <DPMaster>.<Slave>.i{<Nummer>} {.<Offset>,<DPTyp>{,<Anzahl>}} }
- <DPMaster>.<Slave>.q{<Nummer>} {.<Offset>,<DPTyp>{,<Anzahl>}} }

#### **Erklärungen**

##### **<DPMaster>**

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt. Beim DP-Protokoll ist der Verbindungsname der projektierte Name der Kommunikationsbaugruppe.

**<Slave>**

Symbolischer Slave-Name als Kennzeichen für den Zugriff auf einen DP-Slave. Der symbolische Slave-Name kann in STEP 7 / NCM PC (muss ab OPC-Server V8.2 projektiert sein) vergeben werden.

**Hinweis**

Wird in STEP 7 / NCM PC ein OPC-Server < V8.2 projektiert und kein symbolischer Slave-Name vergeben, dann wird als symbolischer Slave-Name der Bezeichner "slv" gefolgt von der projektierten Slave-Adresse verwendet.

Wird in STEP 7 / NCM PC ein OPC-Server >= V8.2 projektiert und kein symbolischer Slave-Name vergeben, dann wird die Bezeichnung des Slave als Slave-Name verwendet. Durch den Konsistenzcheck in STEP 7 / NCM PC wird sichergestellt, dass eindeutige Slave-Namen verwendet werden und der Slave-Name nicht leer ist.

**Hinweis****Zugelassene Zeichen für Slave-Namen**

Folgende Zeichen sind im Zeichenvorrat bei Slave-Namen zugelassen:

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

Folgende Zeichen sind im Zeichenvorrat bei Slave-Namen nicht zugelassen:

- Punkt ".."
- Doppelpunkt ":"
- Pipesymbol "|"
- Backslash "\"
- Eckige Klammern "[" und "]"
- Anführungszeichen """"
- Und-Zeichen "&"
- Fragezeichen "?"
- Spitze Klammern "<" und ">"
- Stern "\*"

Ausgeschlossen ist außerdem das Leerzeichen am Anfang und Ende des Slave-Namen.

**q**

Kennzeichen für einen Ausgang. Ausgänge sind les- und schreibbar.

**i**

Kennzeichen für einen Eingang. Eingänge sind nur lesbar.

**<Nummer>**

Nummer des Moduls, das den Ein- oder Ausgangsbereich enthält.

**<Offset> (Offset Null-basiert - erstes Element ist Null)**

Byteoffset im Adressraum des Slave, an dem das Element liegt, das angesprochen werden soll. Wird ein Modul spezifiziert, so gilt der Offset innerhalb des Moduls. Ohne Angabe des

Moduls bezieht sich der Offset auf den gesamten Ein-/Ausgabebereich des Slave. Der Byteoffset ist nullbasiert.

#### <DPTyp>

Ein DP-Datentyp wird im OPC-UA-Server in den entsprechenden OPC-UA-Datentyp umgewandelt. Die folgende Tabelle listet den Typ-Bezeichner und den entsprechenden OPC-Datentyp auf, in dem der Variablenwert dargestellt werden kann.

Datentyp	OPC-UA-Datentyp	Hinweis
x<Bitadresse>	Boolean	Bit (bool) Zusätzlich zum Byte-Offset im Bereich ist die <Bitadresse> im jeweiligen Byte anzugeben. Wertebereich 0...7
b	Byte	Byte (unsigned)
	ByteString	Wird als Defaultwert verwendet, falls kein <DPTyp> angegeben ist. OPC UA kennt kein "Byte[]", sondern verwendet hierzu den skalaren Datentyp "ByteString".
w	UInt16	Wort (unsigned)
dw	UInt32	Doppelwort (unsigned)
lw	UInt64	Langwort (unsigned)
c	SByte	Byte (signed)
i	Int16	Wort (signed)
di	Int32	Doppelwort (signed)
li	Int64	Langwort (signed)
r	Float	Fließkomma (4 Byte)
lr	Double	Fließkomma (8 Byte)
s<Stringlänge>	String	Es ist die für den String reservierte <Stringlänge> anzugeben. Wertebereich 1...254 Beim Schreiben können auch kürzere Strings geschrieben werden, wobei die übertragene Datenlänge immer die reservierte Stringlänge in Byte ist. Die nicht benötigten Bytes werden mit dem Wert 0 gefüllt.

#### <Anzahl>

Anzahl der Elemente. Der Datentyp der Variable ist ein Feld mit Elementen des angegebenen Formats. Die Angabe einer Anzahl von Feldelementen führt immer zur Bildung eines Feldes vom entsprechenden Typ, auch wenn nur ein einziges Feldelement adressiert wird.

#### 2.5.7.3 Beispiele für Prozessvariablen für den Master Klasse 1

Hier finden Sie Beispiele, die die Syntax von Variablennamen für DP-Variablen verdeutlichen.

## Eingänge

Namensraum-URI: DP: (Namensraum-Index: 3)

- CP 5623.Förderband.i3.0  
bezeichnet Eingangsbyte 0 des Moduls 3 von Slave "Förderband"
- CP 5623.Motorsteuerung.i3.1,b,3  
bezeichnet Feld mit 3 Bytes ab Eingangsbyte 1 des Moduls 3 von Slave "Motorsteuerung"
- CP 5623.Schweißroboter.i3.2,dw  
bezeichnet Doppelwort ab Eingangsbyte 2 des Moduls 3 von Slave "Schweißroboter"
- CP 5623.slv4.i3.0,r  
bezeichnet Fließkommazahl ab Eingangsbyte 0 von Slave 4, Modul 3
- CP 5623.slv4.i.0,b,8  
bezeichnet die ersten 8 Bytes des gesamten Eingangsbereichs von Slave 4 über alle Module hinweg

## Ausgänge

Namensraum-URI: DP: (Namensraum-Index: 3)

- CP 5623.Förderband.q7.1  
bezeichnet Ausgangsbyte 1 des Moduls 7 von Slave "Förderband"
- CP 5623.Motorsteuerung.q.7,2,x5  
bezeichnet Bit 5 im Ausgangsbyte 2 des Slave "Motorsteuerung", Modul 7
- CP 5623.slv4.q.0,w,8  
bezeichnet Feld mit 8 Wörtern aus dem Ausgangsbereich von Slave 4 über alle Module hinweg

### 2.5.7.4 DPC1-Dienste

Mit den DPC1-Diensten können Sie auf Datensätze der DPC1-Slaves zugreifen. Die Übertragung der Datensätze erfolgt azyklisch.

Die Bedeutung der Datensätze ist durch den Hersteller des Slave festgelegt. Sie können beispielsweise für Konfigurationsdaten eines Antriebs genutzt werden.

Der OPC-Server kann beim Anmelden einer DPC1-Variablen nur die Syntax auf Korrektheit überprüfen, aber nicht, ob auf Grund der Projektierung des DPC1-Slave die Variable im Partnergerät gültig und die Größe des Datensatzes ausreichend ist.

### 2.5.7.5 Syntax der Prozessvariablen für DPC1-Dienste

#### Syntax der Prozessvariablen

Optimierte Syntax der Prozessvariablen der DP-OPC-UA-Node-ID für Datensatz-Datenvariablen:

Namensraum-URI: DP: (Namensraum-Index: 3)

## Klassische Syntax

Les- und schreibbare Datensätze (ohne Längenangabe keine schreibbaren Datensätze):

- <DPMaster>.<Slave>.s<Slot>.dr<Index>{,<Länge>} { .<Offset>{,<DPTyp>{,<Anzahl>}} }

Lesbare Datensätze:

- <DPMaster>.<Slave>.s<Slot>.dr<Index>{ .<Offset>{,<DPTyp>{,<Anzahl>}} }

## Erklärungen

### <DPMaster>

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt. Beim DP-Protokoll ist der Verbindungsname der projektierte Name der Kommunikationsbaugruppe.

### <Slave>

Symbolischer Slave-Name als Kennzeichen für den Zugriff auf einen DP-Slave. Der symbolische Slave-Name kann in STEP 7 / NCM PC (muss ab OPC-Server V8.2 projektiert sein) vergeben werden.

---

### Hinweis

Wird in STEP 7 / NCM PC ein OPC-Server < V8.2 projektiert und kein symbolischer Slave-Name vergeben, dann wird als symbolischer Slave-Name der Bezeichner "slv", gefolgt von der projektierten Slave-Adresse, verwendet.

Wird in STEP 7 / NCM PC ein OPC-Server >= V8.2 projektiert und kein symbolischer Slave-Name vergeben, dann wird die Bezeichnung des Slave als Slave-Name verwendet. Durch den Konsistenzcheck in STEP 7 / NCM PC wird sichergestellt, dass eindeutige Slave-Namen verwendet werden und der Slave-Name nicht leer ist.

---

---

**Hinweis****Zugelassene Zeichen für Slave-Namen**

Folgende Zeichen sind im Zeichenvorrat bei Slave-Namen zugelassen:

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

Folgende Zeichen sind im Zeichenvorrat bei Slave-Namen nicht zugelassen:

- Punkt ".."
- Doppelpunkt ":"
- Pipesymbol "|"
- Backslash "\"
- Eckige Klammern "[" und "]"
- Anführungszeichen """"
- Und-Zeichen "&"
- Fragezeichen "?"
- Spitze Klammern "<" und ">"
- Stern "\*"'

Ausgeschlossen ist außerdem das Leerzeichen am Anfang und Ende des Slave-Namen.

---

**s**

Kennzeichen für den Slot des Slave

**<Slot>**

Slot im erweiterten Speicherbereich eines Slaves für azyklische Dienste. Slot und Index kennzeichnen einen Datensatz.

**dr**

Kennzeichen für den Zugriff auf einen Datensatz.

**<Index>**

Index innerhalb eines Slots im erweiterten Speicherbereich eines Slaves für azyklische Dienste. Slot und Index kennzeichnen einen Datensatz.

**<Länge>**

Länge des Datensatzes. Bereich zwischen 1 und 240.

**<Offset>**

Byteadresse im Datensatz für das Element, das angesprochen werden soll.

**<DPTyp>**

Datentyp.

Der Datentyp wird im OPC-UA-Server in den entsprechenden OLE-Datentyp umgewandelt.

Datentyp	OPC-UA-Datentyp	Hinweis
x<Bitadresse>	Boolean	Bit (bool) Zusätzlich zum Byte-Offset im Bereich ist die <Bitadresse> im jeweiligen Byte anzugeben. Wertebereich 0...7
b	Byte	Byte (unsigned)
	ByteString	Wird als Defaultwert verwendet, falls kein <DPTyp> angegeben ist. OPC UA kennt kein "Byte[]", sondern verwendet hierzu den skalaren Datentyp "ByteString".
w	UInt16	Wort (unsigned)
dw	UInt32	Doppelwort (unsigned)
lw	UInt64	Langwort (unsigned)
c	SByte	Byte (signed)
i	Int16	Wort (signed)
di	Int32	Doppelwort (signed)
li	Int64	Langwort (signed)
r	Float	Fließkomma (4 Byte)
lr	Double	Fließkomma (8 Byte)
s<Stringlänge>	String	Es ist die für den String reservierte <Stringlänge> anzugeben. Wertebereich 1...254 Beim Schreiben können auch kürzere Strings geschrieben werden, wobei die übertragene Datenlänge immer die reservierte Stringlänge in Byte ist. Die nicht benötigten Bytes werden mit dem Wert 0 gefüllt.

**<Anzahl>**

Anzahl der Elemente. Der Datentyp der Variable ist ein Feld mit Elementen des angegebenen Formats. Die Angabe einer Anzahl von Feldelementen führt immer zur Bildung eines Feldes vom entsprechenden Typ, auch wenn nur ein einziges Feldelement adressiert wird.

#### 2.5.7.6 Beispiele für Prozessvariablen für DPC1-Dienste

Hier finden Sie Beispiele, die die Syntax von Variablennamen für DPC1-Dienste verdeutlichen.

## Variablennamen für DPC1

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

bezeichnet Zugriff auf das Doppelwort ab Offset 7 in einem Datensatz der Länge 120 Bytes in Slot 3, Index 2 des Slave "Motorsteuerung".

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

bezeichnet Zugriff auf ein Feld mit 4 Bytes ab Offset 8 in einem Datensatz der Länge 120 Bytes in Slot 3, Index 2 des Slave 5.

### 2.5.7.7 Fast Logic für CP 5613/CP 5614/CP 5623/CP 5624 (nur Master)

Der CP 5613/CP 5623 und der DP-Master-Teil des CP 5614/CP 5624 unterstützen die Eigenschaft Fast Logic. Dadurch können Sie den CP so parametrieren, dass er als Reaktion auf Datenänderung beim gleichen oder bei anderen Slaves Werte schreibt. Die Anwenderapplikation wird dabei über die Datenänderung informiert.

Der CP 5613/CP 5614 und CP 5623/CP 5624 stellen 6 Fast Logic Auslöser bereit, die über OPC-UA-Datenvariablen und -Methoden konfiguriert und ausgewertet werden können.

## Vorteile von Fast Logic

Die Verwendung von Fast Logic hat folgende Vorteile:

- Der OPC-Server und der OPC-Client sind entlastet.
- Die Datenübertragung findet schneller statt, weil sie unabhängig von der auf dem PC laufenden Software in der Hardware des CPs abläuft.

---

### Hinweis

Nachdem ein Fast Logic-Trigger ausgelöst wurde, wird er anschließend automatisch deaktiviert. Sie müssen den Trigger dann erneut über die Methode "fl<fastlogicid>.on" aktivieren.

Fast Logic funktioniert nur dann korrekt, wenn der DP-Master im Zustand OPERATE ist und die beteiligten Slaves im Zustand "READY" sind. Deshalb sollte ein Fast Logic Trigger von dem DP-Anwendungsprogramm erst aktiviert werden, wenn das Anwendungsprogramm den DP-Master in den Zustand "OPERATE" versetzt hat und die beteiligten Slaves im Zustand "READY" sind.

Solange Fast Logic-Trigger aktiv sind, darf kein DP-Anwendungsprogramm schreibend auf die Ausgangsbytes zugreifen, die über Fast Logic mit Eingangsbytes verknüpft sind.

---

### 2.5.7.8 Syntax der Steuervariablen für Fast Logic

#### Syntax der OPC-UA-Datenvariablen

Namensraum-URI: DP: (Namensraum-Index: 3)

<DPMaster>.fl<fastlogicid>.state

<DPMaster>.fl<fastlogicid>.fastlogicid

## Syntax der OPC-UA-Methoden

Namensraum-URI: DP: (Namensraum-Index: 3)

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

## Erklärung

### **<DPMaster>**

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt. Beim DP-Protokoll ist der Verbindungsname der projektierte Name der Kommunikationsbaugruppe.

### **fl**

Kennzeichen für Fast Logic.

### **<fastlogicid>**

Nummer des verwendeten Fast Logic Triggers. Wert zwischen 1 und 4.

### **state**

Gibt den Fast Logic Status zurück.

Rückgabewerte:

- CLEARED (0)  
Trigger <fastlogicid> ist nicht aktiviert.
- ACTIVATED (1)  
Trigger <fastlogicid> ist aktiviert.
- TRIGGERED (2)  
Trigger <fastlogicid> hat die Überwachung ausgeführt.

### **fastlogicid**

Gibt die ID des verwendeten Fast Logic Triggers zurück.

### **activate**

Fast Logic Trigger aktivieren und aufrufen.

Die OPC-UA-Methode "activate" wird beim Aufruf als Parameter gesendet.

Wenn ein Parameterblock gesendet wird, wird die Fast Logic-Eigenschaft für den in der OPC-UA-Methode angegebenen Trigger <fastlogicid> festgelegt. Der Parameterblock, der gesendet wird, ist ein Feld mit 8 Bytes und hat folgenden Aufbau:

- slave\_addr\_in\_byte  
Adresse des Slaves, dessen Eingänge für den Trigger selektiert werden
- index\_in\_byte  
Offset des Eingangsbytes des Triggers
- cmp\_value\_in\_byte  
Vergleichswert für das Eingangsbyte
- mask\_in\_byte  
Einzelne Bits im Eingangsbyte können maskiert werden, so dass sie beim Vergleich nicht berücksichtigt werden. Maskiert wird durch eine "1" im entsprechenden Bit, das heißt für mask\_in\_byte==0x00 werden alle Bits von cmp\_value\_in\_byte für den Vergleich

herangezogen. Der Trigger wird ausgelöst, wenn alle nicht maskierten Bits im selektierten Eingangsbyte gleich den Bits in cmp\_value\_in\_byte sind.

- slave\_addr\_out\_byte  
Selektiert den Slave, dessen Ausgangsbyte beim Eintreffen der Trigger-Bedingung verändert werden soll
- index\_out\_byte  
Offset des Ausgangsbytes
- value\_out\_byte  
Wert, der in das Ausgangsbyte geschrieben werden soll
- mask\_out\_byte  
Einzelne Bits im Ausgangsbyte können maskiert werden, so dass sie beim Eintreffen der Trigger-Bedingung nicht verändert werden. Maskiert wird durch eine "1" im entsprechenden Bit, das heißt für mask\_out\_byte==0x00 werden alle Bits von value\_out\_byte in das selektierte Ausgangsbyte geschrieben.

#### **clear**

Fast Logic Trigger abschalten.

Die OPC-UA-Methode "off" kann nur geschrieben werden. Sie besitzt keine Argumente.

## **Beispiele**

OPC-UA-Variable: CP 5614.fl4.state

OPC-UA-Methode: CP 5623.fl3.clear

### **2.5.7.9 Steuertelegramme Sync und Freeze**

Für besondere Anwendungsfälle stehen vier Steuerkommandos zur Verfügung, die mit einem Steuertelegramm versendet werden. In Standardanwendungsfällen werden diese nicht benötigt.

Ein Steuertelegramm ist ein Telegramm, das der Master an einen einzelnen Slave, eine oder mehrere Gruppen oder an alle Slaves sendet. Diese Telegramme werden von den angesprochenen Slaves nicht quittiert.

Steuertelegramme dienen der Übertragung von Steuerkommandos (sogenannte Global Controls Commands) an die ausgewählten Slaves zum Zwecke der Synchronisation. Dabei enthält ein Steuerkommando drei Komponenten:

- Kennung, ob ein oder mehrere DP-Slaves adressiert werden
- Identifikation der Slave-Gruppe
- Steuerkommando(s)

## **Gruppenbildung**

Bei der Projektierung kann einem Slave eine Gruppenidentifikation zugewiesen werden, d. h. es ist möglich, mehrere Slave in einer Gruppe zusammenzufassen.

Welche der Slave zu einer Gruppe gehören, wird beim Erstellen der Datenbasis festgelegt. Dabei kann optional für jeden DP-Slave eine Gruppennummer vergeben werden. Diese

Gruppennummer wird dem DP-Slave in der Parametrierphase bekannt gemacht. Insgesamt können bis zu acht Gruppen gebildet werden.

### **2.5.7.10 Syntax der Steuervariablen für Sync und Freeze**

#### **Syntax der OPC-UA-Methoden**

Namensraum-URI: DP: (Namensraum-Index: 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>
```

#### **Erklärung**

##### **<DPMaster>**

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt. Beim DP-Protokoll ist der Verbindungsname der projektierte Name der Kommunikationsbaugruppe.

##### **<Slave>**

Symbolischer Slave-Name als Kennzeichen für den Zugriff auf einen DP-Slave. Der symbolische Slave-Name kann in STEP 7 / NCM PC (muss ab OPC-Server V8.2 projektiert sein) vergeben werden.

---

##### **Hinweis**

Wird in STEP 7 / NCM PC ein OPC-Server < V8.2 projektiert und kein symbolischer Slave-Name vergeben, dann wird als symbolischer Slave-Name der Bezeichner "slv" gefolgt von der projektierten Slave-Adresse verwendet.

Wird in STEP 7 / NCM PC ein OPC-Server >= V8.2 projektiert und kein symbolischer Slave-Name vergeben, dann wird die Bezeichnung des Slave als Slave-Name verwendet. Durch den Konsistenzcheck in STEP 7 / NCM PC wird sichergestellt, dass eindeutige Slave-Namen verwendet werden und der Slave-Name nicht leer ist.

---

---

**Hinweis****Zugelassene Zeichen für Slave-Namen**

Folgende Zeichen sind im Zeichenvorrat bei Slave-Namen zugelassen:

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

Folgende Zeichen sind im Zeichenvorrat bei Slave-Namen nicht zugelassen:

- Punkt ".."
- Doppelpunkt ":"
- Pipesymbol "|"
- Backslash "\"
- Eckige Klammern "[" und "]"
- Anführungszeichen """"
- Und-Zeichen "&"
- Fragezeichen "?"
- Spitze Klammern "<" und ">"
- Stern "\*"

Ausgeschlossen ist außerdem das Leerzeichen am Anfang und Ende des Slave-Namen.

---

**q**

Kennzeichen für einen Ausgang.

**sync()**

Mit dem Aufruf der OPC-UA-Methode "sync" werden die aktuellen Zustände aller DP-Slave-Ausgänge der ausgewählten Gruppen eingefroren. Alle danach vom DP-Master gesendeten Ausgangs-Daten werden zunächst nicht von den DP-Slaves übernommen.

Der Parameterblock, der geschrieben wird, ist ein Byte mit folgendem Aufbau:

- slavegroups

Jedes Bit des Byte repräsentiert eine von 8 projektierten Slave-Gruppen. Bit 0 repräsentiert die erste Slave-Gruppe.

Wenn Sie den Sync-Aufruf wiederholt senden, dann werden die gerade aktuellen Ausgangsdaten des Master an die Ausgänge der Slaves übermittelt und die Ausgänge erneut eingefroren.

**unsync()**

Mit dem Aufruf der OPC-UA-Methode "unsync" wird das Einfrieren der DP-Slave-Ausgänge aufgehoben. Nach Erhalt des unsync-Aufrufs übernehmen die DP-Slaves alle vom DP-Master aus gesendeten Ausgangs-Daten wieder zyklisch.

Der Parameterblock, der geschrieben wird, ist ein Byte mit folgendem Aufbau:

- slavegroups

Jedes Bit des Byte repräsentiert eine von 8 projektierten Slave-Gruppen. Bit 0 repräsentiert die erste Slave-Gruppe.

**i**

Kennzeichen für einen Eingang.

**freeze()**

Mit dem Aufruf der OPC-UA-Methode "freeze" werden die aktuellen Zustände aller DP-Slave-Eingänge der ausgewählten Gruppen eingefroren. Bei nachfolgenden Lesezyklen erhält der DP-Master diese eingefrorenen Eingangsdaten.

Der Parameterblock, der gesendet wird, ist ein Byte mit folgendem Aufbau:

- slavegroups

Jedes Bit des Byte repräsentiert eine von 8 projektierten Slave-Gruppen. Bit 0 repräsentiert die erste Slave-Gruppe.

Wenn Sie den Freeze-Aufruf wiederholt senden, dann werden die gerade aktuellen Eingangsdaten des Master an die Eingänge der Slaves übermittelt und die Eingänge erneut eingefroren.

**unfreeze()**

Mit dem Aufruf der OPC-UA-Methode "unfreeze" wird das Einfrieren der DP-Slave-Eingänge aufgehoben. Nach Erhalt des unfreeze-Aufrufs stellen die DP-Slaves die aktuellen Eingangsdaten dem DP-Master wieder zur Verfügung.

Der Parameterblock, der gesendet wird, ist ein Byte mit folgendem Aufbau:

- slavegroups

Jedes Bit des Byte repräsentiert eine von 8 projektierten Slave-Gruppen. Bit 0 repräsentiert die erste Slave-Gruppe.

**state**

Kennzeichen für eine Slave-Status-Methode oder -Variable.

**<Status>**

Status	Beschreibung
activate()	Aktiviert den Slave
deactivate()	Deaktiviert den Slave
restart()	Startet den Slave neu
state	Variable, die den Zustand des Slave anzeigt (EnumString).

**Beispiele**

CP 5614.q.sync()

CP 5614.Motorsteuerung.state.state

**2.5.7.11 DP-spezifische Informationsvariablen****DP-spezifische Datenvariablen für Informationen**

Es gibt DP-Master-spezifische Datenvariablen, mit denen Sie Informationen über den Zustand, Eigenschaften bzw. die Art des DP-Master abfragen können.

Folgende Informationen können ermittelt werden:

- Betriebszustand des DP-Master (mode)
- Baugruppeninformationen des DP-Master (info)

- Watchdog des DP-Master (watchdog)
- Slave-Parameter, die der DP-Master von seinen Slaves kennt.

## Syntax der DP-spezifischen Informationsvariablen

Nodeld:

*Namensraum-Index: 3*

*<Masterobjekt>.<Informationsparameter>*

## Betriebszustand des DP-Master

### Syntax für den Betriebszustand des DP-Master (mode)

Namensraum-URI: DP: (Namensraum-Index: 3)

*<DPMaster>.<Informationsparameter>*

## Erklärungen

### <DPMaster>

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt. Beim DP-Protokoll ist der Verbindungsname der projektierte Name der Kommunikationsbaugruppe.

### <Informationsparameter>

Baugruppeninformation	Beschreibung	
mode	Aktuelle Betriebsart des DP-Master. Die aktuelle Betriebsart kann sowohl gelesen als auch geschrieben werden. Das Einstellen der Betriebsart durch Schreiben eines der unten genannten Werte ist nur in Abhängigkeit von der DP-Anwendungsumgebung möglich. Datenvariable vom OPC-UA-Typ "MultistateDiscreteType", Lesen und Schreiben "mode" kann beispielsweise folgende Werte zurückgeben:	
	OFFLINE (0)	Keine Kommunikation zwischen Master und Slave.
	STOP (1)	Bis auf Diagnosedaten keine Kommunikation zwischen Master und Slave.
	CLEAR (2)	Parametrier- und Konfigurierphase
	AUTOCLEAR (3)	Autoclear-Phase, der DP-Master erreicht nicht mehr alle Slaves
	OPERATE (4)	Produktivphase

### Beispiel:

CP 5614.mode

## Baugruppeninformationen des DP-Master

### Syntax für die Baugruppeninformationen des DP-Masters (info)

Namensraum-URI: DP: (Namensraum-Index: 3)

<DPMaster>. <Baugruppeninformation>

## Erklärungen

### <DPMaster>

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt. Beim DP-Protokoll ist der Verbindungsname der projektierte Name der Kommunikationsbaugruppe.

### <Baugruppeninformation>

Baugruppeninformation	Beschreibung																											
identnumber	Ident-Nummer der Zertifizierung Datenvariable vom OPC-UA-Typ "UInt16", nur lesbar																											
hardware	Hardware Datenvariable vom UA-Typ "MultistateDiscreteType", nur lesbar. <table border="1"><tr><td>0</td><td>SOFTNET</td></tr><tr><td>1</td><td>CP 5613 (elektrischer PROFIBUS-Port)</td></tr><tr><td>2</td><td>CP 5613 FO (optischer PROFIBUS-Port)</td></tr><tr><td>3</td><td>CP 5614 (elektrischer PROFIBUS-Port)</td></tr><tr><td>4</td><td>CP 5614 FO (optischer PROFIBUS-Port)</td></tr><tr><td>5</td><td>CP 5614 FO (optischer PROFIBUS-Port) mit externer Spannungsversorgung verbunden</td></tr><tr><td>6</td><td>CP 5613 FO (optischer PROFIBUS-Port) mit externer Spannungsversorgung verbunden</td></tr><tr><td>7</td><td>CP 5613 A2 (elektrischer PROFIBUS-Port)</td></tr><tr><td>8</td><td>CP 5614 A2 (elektrischer PROFIBUS-Port)</td></tr><tr><td>9</td><td>CP 5623 (elektrischer PROFIBUS-Port)</td></tr><tr><td>10</td><td>CP 5624 (elektrischer PROFIBUS-Port)</td></tr><tr><td>11</td><td>CP 5613 A3 (elektrischer PROFIBUS-Port)</td></tr><tr><td>12</td><td>CP 5614 A3 (elektrischer PROFIBUS-Port)</td></tr></table>		0	SOFTNET	1	CP 5613 (elektrischer PROFIBUS-Port)	2	CP 5613 FO (optischer PROFIBUS-Port)	3	CP 5614 (elektrischer PROFIBUS-Port)	4	CP 5614 FO (optischer PROFIBUS-Port)	5	CP 5614 FO (optischer PROFIBUS-Port) mit externer Spannungsversorgung verbunden	6	CP 5613 FO (optischer PROFIBUS-Port) mit externer Spannungsversorgung verbunden	7	CP 5613 A2 (elektrischer PROFIBUS-Port)	8	CP 5614 A2 (elektrischer PROFIBUS-Port)	9	CP 5623 (elektrischer PROFIBUS-Port)	10	CP 5624 (elektrischer PROFIBUS-Port)	11	CP 5613 A3 (elektrischer PROFIBUS-Port)	12	CP 5614 A3 (elektrischer PROFIBUS-Port)
0	SOFTNET																											
1	CP 5613 (elektrischer PROFIBUS-Port)																											
2	CP 5613 FO (optischer PROFIBUS-Port)																											
3	CP 5614 (elektrischer PROFIBUS-Port)																											
4	CP 5614 FO (optischer PROFIBUS-Port)																											
5	CP 5614 FO (optischer PROFIBUS-Port) mit externer Spannungsversorgung verbunden																											
6	CP 5613 FO (optischer PROFIBUS-Port) mit externer Spannungsversorgung verbunden																											
7	CP 5613 A2 (elektrischer PROFIBUS-Port)																											
8	CP 5614 A2 (elektrischer PROFIBUS-Port)																											
9	CP 5623 (elektrischer PROFIBUS-Port)																											
10	CP 5624 (elektrischer PROFIBUS-Port)																											
11	CP 5613 A3 (elektrischer PROFIBUS-Port)																											
12	CP 5614 A3 (elektrischer PROFIBUS-Port)																											
hardwareversion	Hardware-Ausgabestand Datenvariable vom OPC-UA-Typ "Byte", nur lesbar																											
firmwareversion	Firmware-Ausgabestand Datenvariable vom OPC-UA-Typ "UInt16", nur lesbar																											
buspar	Busparameter des Masters lesen Datenvariable vom OPC-UA-Typ "ByteString", nur lesbar																											

### Beispiel:

CP 5614.hardware

## Watchdog des DP-Master

### Syntax für den Watchdog des DP-Master

Namensraum-URI: DP: (Namensraum-Index: 3)

<DPMaster>.watchdog.<Watchdog-Variable>

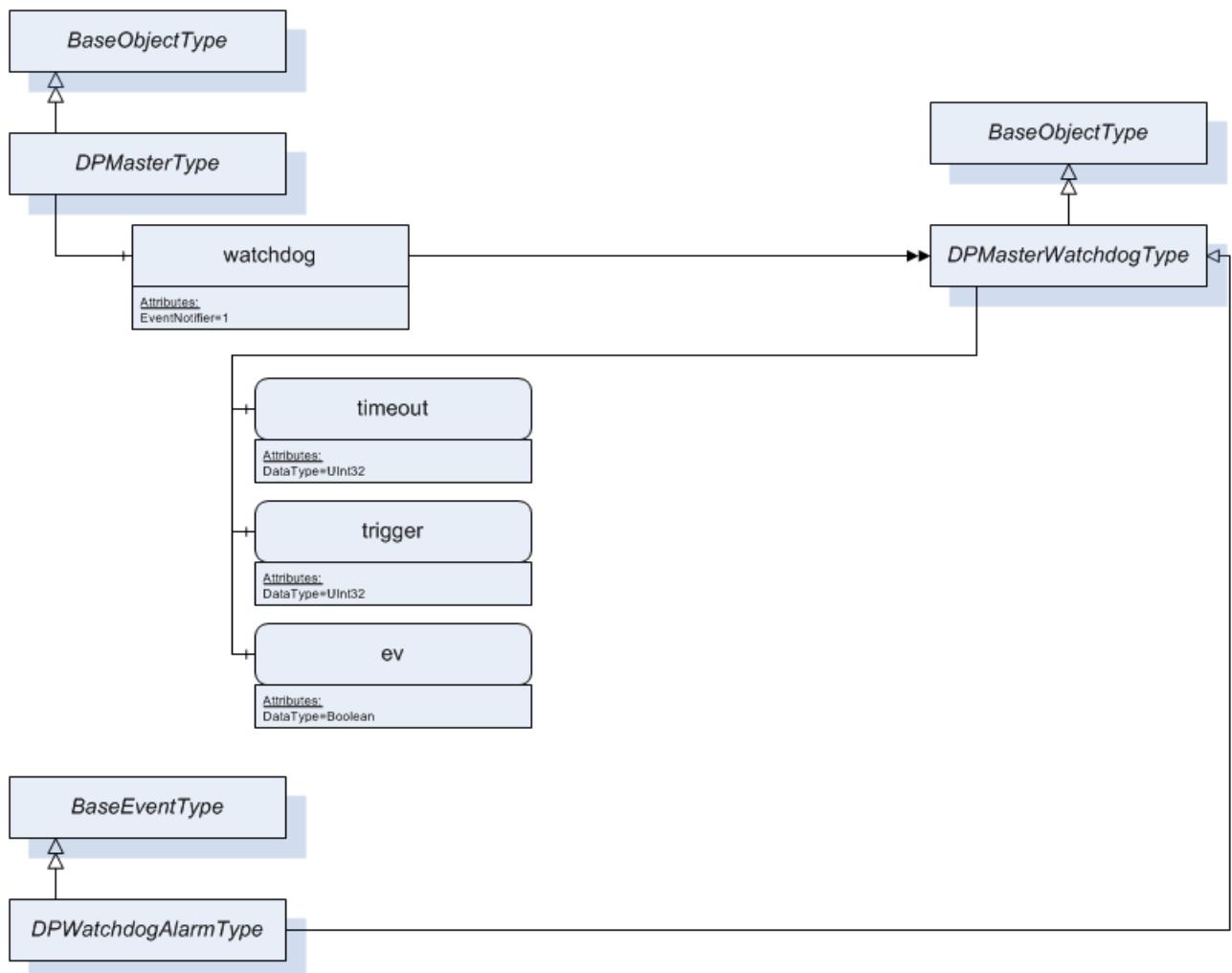


Bild 2-11 Watchdog auf dem DP-Master

watchdog.timeout	Watchdogtimeout lesen bzw. setzen, rücksetzen oder abschalten. Der OPC-UA-Server rundet einen geschriebenen Wert selbstständig auf den nächsthöheren Wert auf. DatenvARIABLE vom OPC-UA-Typ "UInt32", Lesen und Schreiben, die Einheit ist Millisekunden.	
0		Watchdog abschalten

	Wertebereich: zwischen 20 und einschließlich 102.000 ms	Watchdogtimeout setzen. Granularität bei Softnet DP, z.B. CP 5621: 400 ms Granularität bei Hardnet DP, z.B. CP 5623: 10 ms
watchdog.trigger	Watchdog-Funktionsfähigkeit des OPC-UA-Servers überwachen. Datenvariable vom OPC-UA-Typ "UInt32", nur lesbar	
watchdog.ev	Auslösen des Watchdogs überwachen Datenvariable vom OPC-UA-Typ "Boolean", nur lesbar	
	False	Normalbetrieb
	True	DP-Protokollstack hat den Watchdog ausgelöst.

**Beispiel:**

CP 5614.watchdog.ev

CP 5614.watchdog.timeout

**DP-Slave an einem DP-Master****Syntax für DP-Slave-Informationsvariablen an einem DP-Master**

Namensraum-URI: DP: (Namensraum-Index: 3)

&lt;DPMaster&gt;.&lt;Slave&gt;.&lt;Informationsvariablen&gt;

**Erklärungen****<DPMaster>**

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt. Beim DP-Protokoll ist der Verbindungsname der projektierte Name der Kommunikationsbaugruppe.

**<Slave>**

Symbolischer Slave-Name als Kennzeichen für den Zugriff auf einen DP-Slave. Der symbolische Slave-Name kann in STEP 7 / NCM PC vergeben werden.

**Hinweis**

Wird in STEP 7 / NCM PC ein OPC-Server < V8.2 projektiert und kein symbolischer Slave-Name vergeben, dann wird als symbolischer Slave-Name der Bezeichner "slv" gefolgt von der projektierten Slave-Adresse verwendet.

Wird in STEP 7 / NCM PC ein OPC-Server >= V8.2 projektiert und kein symbolischer Slave-Name vergeben, dann wird die Bezeichnung des Slave als Slave-Name verwendet. Durch den Konsistenzcheck in STEP 7 / NCM PC wird sichergestellt, dass eindeutige Slave-Namen verwendet werden und der Slave-Name nicht leer ist.

**<Adresse>**

PROFIBUS-Adresse des DP-Slave.

Bereich: 0 ... 126.

**<Informationsvariablen>**

Informationsvariablen	Beschreibung				
type	DP-Slave-Typ ermitteln. Datenvvariable vom UA-Typ "MultistateDiscreteType", nur lesbar.				
	0	NORM	Norm-DP-Slave		
	1	ET200_U	Nicht Norm-Slave: ET 200 U		
	2	ET200K_B	Nicht Norm-Slave: ET 200 K/B		
	3	ET200_SPM	Nicht Norm-Slave: Allgemeine SPM-Station		
dpv	DP-Protokollversion des DP-Slave ermitteln. Datenvvariable vom UA-Typ "MultistateDiscreteType", nur lesbar.				
	0	DP-V0			
	1	DP-V1			
	2	DP-V2			
parcfgdata	Mit dieser Informationsvariablen kann der Aufbau der IO-Module des DP-Slave vom CP erfragt werden. Die Parameter stammen aus der Projektierung. Datenvvariable vom UA-Typ "ByteString", nur lesbar				
parprmdata	Mit dieser Informationsvariablen können die Parametrierdaten des DP-Slave erfragt werden. Datenvvariable vom UA-Typ "ByteString", nur lesbar				
paruserdata	Mit dieser Informationsvariablen können die Anwender-Parametrierdaten des DP-Slave erfragt werden. Datenvvariable vom UA-Typ "ByteString", nur lesbar				
partype	Mit dieser Informationsvariablen können die allgemeinen Slave-Parameter wie z. B. SI-Flag, Slave-Typ und Oktett-String erfragt werden. Datenvvariable vom UA-Typ "ByteString", nur lesbar				
paraddtab	Mit dieser Informationsvariablen können die unterschiedlichen Komponenten der Slave-Parameter eines DP-Slaves auslesen. Datenvvariable vom UA-Typ "ByteString", nur lesbar				

**Beispiele:**

CP 5611.Motorsteuerung.type

CP 5614.slv17.dpv

**Syntax für DP-Slave- IO-Informationsvariablen an einem DP-Master**

Namensraum-URI: DP: (Namensraum-Index: 3)

Es gibt folgende Möglichkeiten:

- <DPMaster>.<Slave>.q.<SlaveEigenschaften>
- <DPMaster>.<Slave>.i.<SlaveEigenschaften>

- <DPMaster>.<Slave>.q<ModuleID>.<SlaveModuleEigenschaften>
- <DPMaster>.<Slave>.i<ModuleID>.<SlaveModuleEigenschaften>

## Erklärungen

### <DPMaster>

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt. Beim DP-Protokoll ist der Verbindungsname der projektierte Name der Kommunikationsbaugruppe.

### <Slave>

Symbolischer Slave-Name als Kennzeichen für den Zugriff auf einen DP-Slave. Der symbolische Slave-Name kann in STEP 7 / NCM PC vergeben werden.

---

### Hinweis

Wird in STEP 7 / NCM PC ein OPC-Server < V8.2 projektiert und kein symbolischer Slave-Name vergeben, dann wird als symbolischer Slave-Name der Bezeichner "slv" gefolgt von der projektierten Slave-Adresse verwendet.

Wird in STEP 7 / NCM PC ein OPC-Server >= V8.2 projektiert und kein symbolischer Slave-Name vergeben, dann wird die Bezeichnung des Slave als Slave-Name verwendet. Durch den Konsistenzcheck in STEP 7 / NCM PC wird sichergestellt, dass eindeutige Slave-Namen verwendet werden und der Slave-Name nicht leer ist.

---

**q**

Kennzeichen für einen Ausgang. Ausgänge sind les- und schreibbar

**i**

Kennzeichen für einen Eingang. Eingänge sind nur lesbar.

### <ModuleID>

ID des Moduls, das den Ein- oder Ausgangsbereich enthält.

### <SlaveEigenschaften>

IO-Informationsvariablen	Beschreibung	
length	Modullänge in Bytes aus den DP-Slave-Konfigurationsdaten Property vom Typ UInt16, nur lesbar	
modultype	Modultyp aus den DP-Slave-Konfigurationsdaten. DatenvARIABLE vom UA-Typ "MultistateDiscreteType", nur lesbar.	
	0	UNKNOWN
	1	I (Eingang)
	2	Q (Ausgang)
	3	IQ (Eingang/Ausgang)
	Der Wert "3" kommt nur bei DP-Slave-Modulen vor. In diesem Fall gibt es sowohl ein Input- als auch ein Output-Modul mit gleicher "moduleid".	
slaveid	PROFIBUS-Adresse des zugehörigen DP-Slave Property vom Typ "Byte", nur lesbar	

## &lt;SlaveModuleEigenschaften&gt;

IO-Informationsvariablen	Beschreibung	
length	Modullänge in Bytes aus den DP-Slave-Konfigurationsdaten Property vom Typ UInt16, nur lesbar	
modultype	Modultyp aus den DP-Slave-Konfigurationsdaten. Datenvvariable vom UA-Typ "MultistateDiscreteType", nur lesbar.	
	0	UNKNOWN
	1	I (Eingang)
	2	Q (Ausgang)
	3	IQ (Eingang/Ausgang)
	Der Wert "3" kommt nur bei DP-Slave-Modulen vor. In diesem Fall gibt es sowohl ein Input- als auch ein Output-Modul mit gleicher "moduleid".	
slaveid	PROFIBUS-Adresse des zugehörigen DP-Slave Property vom Typ "Byte", nur lesbar	
moduleid	Moduladresse aus den DP-Slave-Konfigurationsdaten. Property vom Typ "Byte", nur lesbar	
slot	Slot bzw. Baugruppenkennung dieses Moduls. Property vom Typ "Byte", nur lesbar	
dataunitlength	Länge der Dateneinheit eines Moduls aus den DP-Slave-Konfigurationsdaten. Datenvvariable vom UA-Typ "MultistateDiscreteType", nur lesbar	
	0	UNKNOWN
	1	BYTE
	2	WORD
consistence	Konsistenz über alle Dateneinheiten eines Moduls aus den DP-Slave-Konfigurationsdaten. Property vom Typ "Boolean", nur lesbar	

**Beispiele:**

CP 5614.Förderband.q.length

CP 5614.slv17.q.modultype

CP 5614.slv17.q0.length

**Syntax für DP-Slave-Diagnosevariablen an einem DP-Master**

Namensraum-URI: DP: (Namensraum-Index: 3)

&lt;DPMaster&gt;.&lt;Slave&gt;.diagnosis.data{.&lt;Diagnosevariablen&gt;}

**Erklärungen****<DPMaster>**

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt. Beim DP-Protokoll ist der Verbindungsname der projektierte Name der Kommunikationsbaugruppe.

**<Slave>**

Symbolischer Slave-Name als Kennzeichen für den Zugriff auf einen DP-Slave. Der symbolische Slave-Name kann in STEP 7 / NCM PC vergeben werden.

**Hinweis**

Wird in STEP 7 / NCM PC ein OPC-Server < V8.2 projektiert und kein symbolischer Slave-Name vergeben, dann wird als symbolischer Slave-Name der Bezeichner "slv" gefolgt von der projektierten Slave-Adresse verwendet.

Wird in STEP 7 / NCM PC ein OPC-Server >= V8.2 projektiert und kein symbolischer Slave-Name vergeben, dann wird die Bezeichnung des Slave als Slave-Name verwendet. Durch den Konsistenzcheck in STEP 7 / NCM PC wird sichergestellt, dass eindeutige Slave-Namen verwendet werden und der Slave-Name nicht leer ist.

**diagnosis**

Diagnosedaten des DP-Slave.

**data**

Enthält die letzten Diagnosedaten eines DP-Slave. Die ersten 6 Bytes enthalten die Standarddiagnose des DP-Slave. Datenvariable vom Typ "ByteString", die nur lesbar ist.

**<Diagnosevariablen>**

Diagnosevariablen	Beschreibung
masterlock	Bit7 aus Byte1 der Diagnosedaten. Der DP-Slave ist bereits von einem anderen DP-Master parametriert worden, d.h. der eigene DP-Master hat momentan keinen Zugriff auf diesen DP-Slave. Datenvariable vom Typ "Boolean", nur lesbar
prmfault	Bit6 aus Byte1 der Diagnosedaten. Dieses Bit wird vom DP-Slave gesetzt, falls das letzte Parametriertelegramm fehlerhaft war (beispielsweise falsche Länge, falsche Ident-Nummer, ungültige Parameter). Datenvariable vom Typ "Boolean", nur lesbar
invalidslaveresponse	Bit5 aus Byte1 der Diagnosedaten. Dieses Bit wird gesetzt, sobald von einem angesprochenen DP-Slave eine un-plausible Antwort empfangen wird. Datenvariable vom Typ "Boolean", nur lesbar
notsupported	Bit4 aus Byte1 der Diagnosedaten. Dieses Bit wird gesetzt, sobald eine Funktion angefordert wurde, die von diesem DP-Slave nicht unterstützt wird (beispielsweise wird ein Betrieb im SYNC-Mode gefordert, vom DP-Slave aber nicht unterstützt). Datenvariable vom Typ "Boolean", nur lesbar
extdiag	Bit3 aus Byte1 der Diagnosedaten. Dieses Bit wird vom DP-Slave gesetzt. Ist das Bit gesetzt, muss in dem Slave-spezifischen Diagnosebereich (Ext_Diag_Data, Byte 7-32) ein Diagnoseeintrag vorliegen. Ist das Bit nicht gesetzt, kann in dem Slave-spezifischen Diagnosebereich (Ext_Diag_Data, Byte 7-32) eine Statusmeldung vorliegen. Die Bedeutung dieser Statusmeldung ist applikationsspezifisch zu vereinbaren. Datenvariable vom Typ "Boolean", nur lesbar

Diagnosevariablen	Beschreibung
cfgfault	Bit2 aus Byte1 der Diagnosedaten. Dieses Bit wird gesetzt, sobald die vom DP-Master zuletzt gesendeten Konfigurationsdaten mit denjenigen, die der DP-Slave ermittelt hat, nicht übereinstimmen, d. h. es liegt ein Konfigurationsfehler vor. Datenvvariable vom Typ "Boolean", nur lesbar
stationnotready	Bit1 aus Byte1 der Diagnosedaten. Dieses Bit wird gesetzt, wenn der DP-Slave noch nicht für den Produktivdatenaustausch bereit ist. Datenvvariable vom Typ "Boolean", nur lesbar
stationnonexistent	Bit0 aus Byte1 der Diagnosedaten. Dieses Bit setzt der DP-Master, falls der DP-Slave nicht über den Bus erreichbar ist. Ist dieses Bit gesetzt, so enthalten die Diagnosebits den Zustand der letzten Diagnosemeldung oder den Initialwert. Der DP-Slave setzt dieses Bit fest auf Null. Datenvvariable vom Typ "Boolean", nur lesbar
deactivated	Bit7 aus Byte2 der Diagnosedaten. Dieses Bit wird gesetzt, sobald der DP-Slave im lokalen Parametersatz als nicht aktiv gekennzeichnet und aus der zyklischen Bearbeitung herausgenommen wurde. Datenvvariable vom Typ "Boolean", nur lesbar Hinweis: Bit6 ist reserviert.
syncmode	Bit5 aus Byte2 der Diagnosedaten. Dieses Bit wird vom DP-Slave gesetzt, sobald dieser DP-Slave das Sync-Steuerkommando erhalten hat. Datenvvariable vom Typ "Boolean", nur lesbar
freezemode	Bit4 aus Byte2 der Diagnosedaten. Dieses Bit wird vom DP-Slave gesetzt, sobald dieser DP-Slave das Freeze-Steuerkommando erhalten hat. Datenvvariable vom Typ "Boolean", nur lesbar
wdon	Bit3 aus Byte2 der Diagnosedaten. Dieses Bit wird vom DP-Slave gesetzt. Ist dieses Bit auf 1 gesetzt, so ist die Ansprechüberwachung beim DP-Slave aktiviert. Datenvvariable vom Typ "Boolean", nur lesbar
statdiag	Bit1 aus Byte2 der Diagnosedaten. Setzt der DP-Slave dieses Bit, muss der DP-Master solange Diagnoseinformationen empfangen, bis dieses Bit wieder gelöscht wird. Der DP-Slave setzt zum Beispiel dieses Bit, wenn er keine gültigen Nutzdaten zur Verfügung stellen kann. Datenvvariable vom Typ "Boolean", nur lesbar Hinweis: Bit 2: Dieses Bit wird vom DP-Slave fest auf 1 gesetzt.
prmreq	Bit0 aus Byte2 der Diagnosedaten. Setzt der DP-Slave dieses Bit, muss er neu parametriert und konfiguriert werden. Das Bit bleibt solange gesetzt, bis eine Parametrierung erfolgt ist. Datenvvariable vom Typ "Boolean", nur lesbar. Hinweis: Sind Bit 1 und Bit 0 gleichzeitig gesetzt, so hat Bit 0 die höhere Priorität.

Diagnosevariablen	Beschreibung
extdiagoverflow	Bit7 aus Byte3 der Diagnosedaten. Ist dieses Bit gesetzt, liegen mehr Diagnoseinformationen vor, als in "Ext_Diag_Data" angegeben sind. Zum Beispiel setzt der DP-Slave dieses Bit, wenn mehr Kanaldiagnosen vorliegen, als der DP-Slave in seinen Sendepuffer eintragen kann; oder der DP-Master setzt dieses Bit, wenn der DP-Slave mehr Diagnosedaten sendet, als der DP-Master in seinem Diagnosepuffer berücksichtigen kann. Datenvariable vom Typ "Boolean", nur lesbar
masteraddr	Byte4 der Diagnosedaten. In das Byte 4, dem "Diag. Master_Add"-Byte, wird die Adresse des DP-Masters eingetragen, der diesen DP-Slave parametriert hat. Ist der DP-Slave von keinem DP-Master parametriert, so setzt der DP-Slave in dieses Byte die Adresse 255 ein. Datenvariable vom Typ "Byte", nur lesbar Hinweis: Bit6 bis Bit0 sind reserviert.
identnumber	Byte5 und 6 der Diagnosedaten. In die Bytes 5 und 6, den "Ident_Number"-Bytes, wird die Herstellerkennung für einen DP-Slave-Typ vergeben. Diese Kennung kann zum einen für Prüfungs-zwecke und zum anderen zur genauen Identifizierung herangezogen werden. Datenvariable vom Typ "Word", nur lesbar
extdiagdata	Byte7-32 der Diagnosedaten. Ab Byte 7, den "Ext_Diag_Data"-Bytes, kann der DP-Slave seine spezifische Diagnose ablegen. Es wird eine Blockstruktur mit je einem Headerbyte für die geräte- und kennungsbezogene Diagnose vorgeschrieben. Datenvariable vom Typ "ByteString", nur lesbar

**Beispiele:**

CP 5611.Motorsteuerung.diagnosis.data

CP 5611.slv17.diagnosis.data.stationnotready

CP 5611.slv17.diagnosis.data.identnumber

**Beispiele für DP-spezifische Informationsvariablen**

Hier finden Sie Beispiele, welche die Syntax der Namen von DP-spezifischen Informationsvariablen verdeutlichen.

**Informationen über die Hardware eines DP-Masters**

Nodeld:

- Namensraum-URI:  
*DP: (Namensraum-Index: 3)*
- Bezeichner:  
*CP 5611 A2.hardware* vom Typ "int"

Möglicher Rückgabewert: SOFTNET, CP5613,CP5614 usw.

## Watchdog eines DP-Masters

Nodeld:

- Namensraum-URI:  
*DP: (Namensraum-Index: 3)*
- Bezeichner:
  - *CP 5624.watchdog.timeout*  
Möglicher Rückgabewert: 0 = Watchdog wird deaktiviert
  - *CP 5624.watchdog.trigger*  
Möglicher Rückgabewert: 2000 = Watchdog-Funktionsfähigkeit des DP-OPC-UA-Servers
  - *CP 5624.watchdog.ev*  
Möglicher Rückgabewert: True = Watchdog wurde aktiviert

## 2.5.8 DP-Slave

---

### Hinweis

Ein PROFIBUS-Kommunikationsprozessor als DP-Slave kann über OPC nur als DP-V0-Slave betrieben werden. Mit den genannten HARDNET-Baugruppen können somit über OPC keine Datensätze gelesen oder geschrieben werden.

---

### 2.5.8.1 Variabldienste zum Zugriff auf lokale DP-Slave-Daten

Der OPC-Server für SIMATIC NET kann in einem PROFIBUS DP Netz neben der DP-Master-Funktion auch die Rolle des DP-Slaves übernehmen. Der OPC-Server verwaltet Speicherbereiche für die Ein- und Ausgänge dieses DP-Slaves und bildet diese auf OPC-Variablen ab. Ein OPC-Client kann die vom Master gesetzten Ausgänge lesen und in den Eingängen Werte setzen, die der DP-Master im nächsten Zyklus abholt.

DP-Slaves sind modular aufgebaut. Ein DP-Slave kann mehrere Module mit unterschiedlichen Ein-/Ausgangsbereichen enthalten. Die Zuordnung der Module geschieht durch die DP-Projektierung.

Der Variablenname bezeichnet einen Ein- oder Ausgangsbereich im Modul eines Slave. Der Zugriff auf die Ein- und Ausgänge des Slaves erfolgt durch Angabe der Modul-Nummer und des Ein- oder Ausgangsbereichs.

Über Variablennamen können Statusinformation der Slaves und des DP-Master abgefragt werden.

## 2.5.8.2 Syntax der Prozessvariablen für den DP-Slave

### Syntax der Prozessvariablen

Optimierte Syntax der Prozessvariablen der DP-OPC-UA-Node-ID, für das Lesen und Schreiben von IO-Datenvariablen:

Namensraum-URI: DP: (Namensraum-Index: 3)

### Klassische Syntax

Es gibt zwei Möglichkeiten:

- <DPSlave>.i{ {<Nummer>} .<Offset>{ ,<DPTyp>{ ,<Anzahl>}} }
- <DPSlave>.q{ {<Nummer>} .<Offset>{ ,<DPTyp>{ ,<Anzahl>}} }

### Erklärungen

#### <DPSlave>

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt. Beim DP-Protokoll ist der Verbindungsname der projektierte Name der Kommunikationsbaugruppe.

#### q

Kennzeichen für einen Ausgang. Ausgänge sind nur lesbar.

#### i

Kennzeichen für einen Eingang. Eingänge sind les- und schreibbar.

#### <Nummer>

Nummer des Moduls, das den Ein- oder Ausgangsbereich enthält.

#### <Offset>

Byteoffset im Adressraum des Slave, an dem das Element liegt, das angesprochen werden soll. Wird ein Modul spezifiziert, so gilt der Offset innerhalb des Moduls. Ohne Angabe des Moduls bezieht sich der Offset auf den gesamten Ein-/Ausgabebereich des Slave.

#### <DPTyp>

Ein DP-Datentyp wird im OPC-UA-Server in den entsprechenden OPC-UA-Datentyp umgewandelt. Die folgende Tabelle listet den Typ-Bezeichner und den entsprechenden OPC-Datentyp auf, in dem der Variablenwert dargestellt werden kann.

Datentyp	OPC UA Datentyp	Hinweis
x<Bitadresse>	Boolean	Bit (bool) Zusätzlich zum Byte-Offset im Bereich ist die <Bitadresse> im jeweiligen Byte anzugeben. Wertebereich 0...7
b	Byte	Byte (unsigned) Wird als Defaultwert verwendet, falls kein <DPTyp> angegeben ist.
	ByteString	OPC UA kennt kein "Byte[]", sondern verwendet hierzu den skalaren Datentyp "ByteString".
w	UInt16	Wort (unsigned)

Datentyp	OPC UA Datentyp	Hinweis
dw	UInt32	Doppelwort (unsigned)
lw	UInt64	Langwort (unsigned)
c	SByte	Byte (signed)
i	Int16	Wort (signed)
di	Int32	Doppelwort (signed)
li	Int64	Langwort (signed)
r	Float	Fließkomma (4 Byte)
lr	Double	Fließkomma (8 Byte)
s<Stringlänge>	String	<p>Es ist die für den String reservierte &lt;Stringlänge&gt; anzugeben. Wertebereich 1...254</p> <p>Beim Schreiben können auch kürzere Strings geschrieben werden, wobei die übertragene Datenlänge immer die reservierte Stringlänge in Byte zuzüglich 2 Byte ist. Die nicht benötigten Bytes werden mit dem Wert 0 gefüllt.</p> <p>Das Lesen und Schreiben von Strings und String-Arrays wird intern auf das Lesen und Schreiben von Byte-Arrays abgebildet.</p>

#### <Anzahl>

Anzahl der Elemente. Der Datentyp der Variable ist ein Feld mit Elementen des angegebenen Formats. Die Angabe einer Anzahl von Feldelementen führt immer zur Bildung eines Feldes vom entsprechenden Typ, auch wenn nur ein einziges Feldelement adressiert wird.

### 2.5.8.3 Beispiele für Prozessvariablen für den DP-Slave

Hier finden Sie einige Beispiele.

#### Eingänge

- CP 5611.i3.0,b  
bezeichnet Eingangsbyte 0 (Offset 0) im Modul 3 des DP-Slave
- CP 5611.i3.1,b,3  
bezeichnet ein Feld mit 3 Bytes ab Eingangsbyte1 (Offset 1) im Modul 3 des DP-Slave
- CP 5614.i3.0,x0  
Eingangsbit 0 im Byte 0 im Modul 3 des DP-Slave
- CP 5614.i3.3,b,8  
bezeichnet ein Feld mit 8 Eingangsbytes ab Offset 3 im Modul 3 des DP-Slave

#### Ausgänge

- CP 5611.q4.3,w  
bezeichnet ein Ausgangswort an Adresse 3 im Modul 4 des DP-Slave
- CP 5611.q3.2,dw  
bezeichnet ein Ausgangsdoppelwort an Adresse 2 im Modul 3 des DP-Slave

- CP 5611,q0.3,x7  
bezeichnet das Ausgangsbit 7 im Byte 3 des DP-Slave
- CP 5614.q1.0,w,4  
bezeichnet ein Feld mit 4 Ausgangsworten im Modul 1 des DP-Slave

#### 2.5.8.4 Status des DP-Slave

##### Syntax für den Status des DP-Slave

Namensraum-URI: DP: (Namensraum-Index: 3)

<DPslave>.<Statusvariablen>

##### Erklärungen

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

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt. Beim DP-Protokoll ist der Verbindungsname der projektierte Name der Kommunikationsbaugruppe.

###### **state**

Gibt den Status des DP-Slave an.

###### **<Statusvariablen>**

Statusvariablen	Beschreibung		
state.state	Status des DP-Slave ermitteln. Datenvariable vom UA-Typ "MultistateDiscreteType", nur lesbar		
	0	NO_DATA_EXCHANGE	Der DP-Master hat den DP-Slave noch nicht konfiguriert und parametriert.
	1	DATA_EXCHANGE	Der DP-Slave befindet sich im zyklischen Datenaustausch mit dem DP-Master.

#### 2.5.8.5 Betriebszustand des DP-Slave

##### Syntax für den Betriebszustand des DP-Slave

<DPslave>.<mode>

##### Erklärungen

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

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt. Beim DP-Protokoll ist der Verbindungsname der projektierte Name der Kommunikationsbaugruppe.

**mode**

	<b>Beschreibung</b>		
mode	Betriebszustand des Slaves ermitteln oder setzen. Datenvariable vom UA-Typ "MultistateDiscreteType", Lesen und Schreiben		
	0	OFFLINE	Der DP-Slave reagiert nicht am Bus.
	1	ONLINE	Der DP-Slave ist bereit, die Parametrier- und Konfiguriertelegramme des DP-Masters zu empfangen und letztlich, nach erfolgreicher Parametrierung und Konfigurierung, in den Zustand "DATA_EXCHANGE" zu wechseln.

**2.5.8.6 DP-Slave-spezifische Informationsvariablen**

Für Diagnosezwecke beim DP-Slave gibt es einige vordefinierte Informationsvariablen.

**2.5.8.7 Syntax der DP-Slave-spezifischen Informationsvariablen****Syntax für DP-Slave-Informationsvariablen an einem DP-Slave**

Namensraum-URI: DP: (Namensraum-Index: 3)

<DPSlave>.<Informationsvariablen>

**Erklärungen****<DPSlave>**

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt. Beim DP-Protokoll ist der Verbindungsname der projektierte Name der Kommunikationsbaugruppe.

**<Informationsvariablen>**

<b>Informationsvariablen</b>	<b>Beschreibung</b>		
type	Slave-Typ ermitteln. Datenvariable vom UA-Typ "MultistateDiscreteType", nur lesbar		
	0	NORM	Norm-DP-Slave
	Der vom OPC-Server betriebene DP-Slave ist immer ein Norm-DP Slave.		
dpv	DP-Protokollversion des DP-Slave ermitteln. Datenvariable vom UA-Typ "Byte", nur lesbar		
	0	DP-V0	
	1	DP-V1	
	2	DP-V2	
	Immer 0, der vom OPC-Server betriebene DP-Slave unterstützt nur DP-V0.		
parcfgdata	Konfigurierdaten des DP-Slave. Die Konfigurierdaten enthalten die Angaben über den Aufbau der IO-Module des DP-Slave und werden vom OPC-UA-Server dazu benutzt, die zum DP-Slave gehörigen IO-Modulobjekte aufzubauen. Datenvariable vom UA-Typ "ByteString", nur lesbar		

Informationsvariablen	Beschreibung	
paruserdata	Anwender-Parametrierdaten des DP-Slave. Für diese werden in der Regel keine Daten geliefert. Datenvariable vom UA-Typ "ByteString", nur lesbar	
hardware	Hardware. Datenvariable vom UA-Typ "MultistateDiscreteType", nur lesbar	
	0	SOFTNET
	1	CP 5613 (elektrischer PROFIBUS-Port)
	2	CP 5613 FO (optischer PROFIBUS-Port)
	3	CP 5614 (elektrischer PROFIBUS-Port)
	4	CP 5614 FO (optischer PROFIBUS-Port)
	5	CP 5614 FO (optischer PROFIBUS-Port) mit externer Spannungsversorgung
	6	CP 5613 FO (optischer PROFIBUS-Port) mit externer Spannungsversorgung
	7	CP 5613 A2 (elektrischer PROFIBUS-Port)
	8	CP 5614 A2 (elektrischer PROFIBUS-Port)
	9	CP 5623 (elektrischer PROFIBUS-Port)
	10	CP 5624 (elektrischer PROFIBUS-Port)
	11	CP 5613 A3 (elektrischer PROFIBUS-Port)
	12	CP 5614 A3 (elektrischer PROFIBUS-Port)
hardwareversion	Hardware-Ausgabestand. Datenvariable vom UA-Typ "Byte", nur lesbar	
firmwareversion	Firmware-Ausgabestand. Datenvariable vom UA-Typ "UInt16", nur lesbar	
identnumber	Ident-Nummer der Zertifizierung. Datenvariable vom UA-Typ "UInt16", nur lesbar	

**Beispiel:**

CP 5611.hardware

**Syntax für DP-Slave-IO-Informationsvariablen an einem DP-Slave**

Es gibt 4 Möglichkeiten:

- <DPslave>.q.<SlaveEigenschaften>
- <DPslave>.i.<SlaveEigenschaften>
- <DPslave>.q<Nummer>.<SlaveModuleEigenschaften>
- <DPslave>.i<Nummer>.<SlaveModuleEigenschaften>

## Erklärungen

### <DPSlave>

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt. Beim DP-Protokoll ist der Verbindungsname der projektierte Name der Kommunikationsbaugruppe.

**q**

Kennzeichen für einen Ausgang. Ausgänge sind nur lesbar.

**i**

Kennzeichen für einen Eingang. Eingänge sind les- und schreibbar.

### <Nummer>

Nummer des Moduls, das den Ein- oder Ausgangsbereich enthält.

### <SlaveEigenschaften>

IO-Informationsvariablen	Beschreibung	
length	Modullänge in Bytes aus den DP-Slave-Konfigurationsdaten Property vom Typ UInt16, nur lesbar	
modultype	Modultyp aus den DP-Slave-Konfigurationsdaten. DatenvARIABLE vom UA-Typ "MultistateDiscreteType", nur lesbar.	
	0	UNKNOWN
	1	I (Eingang)
	2	Q (Ausgang)
	3	IQ (Eingang/Ausgang)
	Der Wert "3" kommt nur bei DP-Slave-Modulen vor. In diesem Fall gibt es sowohl ein Input- als auch ein Output-Modul mit gleicher "moduleid".	
slaveid	PROFIBUS-Adresse des zugehörigen DP-Slave Property vom Typ "Byte", nur lesbar	

### <SlaveModuleEigenschaften>

IO-Informationsvariablen	Beschreibung	
length	Modullänge in Bytes aus den DP-Slave-Konfigurationsdaten Property vom Typ UInt16, nur lesbar	
modultype	Modultyp aus den DP-Slave-Konfigurationsdaten. DatenvARIABLE vom UA-Typ "MultistateDiscreteType", nur lesbar.	
	0	UNKNOWN
	1	I (Eingang)
	2	Q (Ausgang)
	3	IQ (Eingang/Ausgang)
	Der Wert "3" kommt nur bei DP-Slave-Modulen vor. In diesem Fall gibt es sowohl ein Input- als auch ein Output-Modul mit gleicher "moduleid".	
slaveid	PROFIBUS-Adresse des zugehörigen DP-Slave Property vom Typ "Byte", nur lesbar	

IO-Informationsvariablen	Beschreibung	
moduleid	Moduladresse aus den DP-Slave-Konfigurationsdaten. Property vom Typ "Byte", nur lesbar	
slot	Slot bzw. Baugruppenkennung dieses Moduls. Property vom Typ "Byte", nur lesbar	
dataunitlength	Länge der Dateneinheit eines Moduls aus den DP-Slave-Konfigurationsdaten. Datenvariable vom UA-Typ "MultistateDiscreteType", nur lesbar	
	0	UNKNOWN
	1	BYTE
	2	WORD
consistence	Konsistenz über alle Dateneinheiten eines Moduls aus den DP-Slave-Konfigurationsdaten. Property vom Typ "Boolean", nur lesbar	

**Beispiele:**

CP 5614\_s.q.length

CP 5614\_s.q.moduletype

CP 5614\_s.q0.length

**2.5.9 DP-OPC-UA-Template-Datenvariablen****2.5.9.1 DP-OPC-UA-Template-Datenvariablen**

Sie haben mit den Prozessvariablen für das DP-OPC-UA-Protokoll flexible Einstellungsmöglichkeiten, um die Prozessdaten Ihrer Anlage in den gewünschten Datenformaten zu erhalten.

Die Vielfalt der Adressierungsmöglichkeiten lässt sich allerdings nicht in einen vollständig durchsuchbaren Namensraum fassen. Bereits ein Datenbaustein mit der Länge eines einzelnen Bytes besitzt etwa 40 verschiedene Datenformatoptionen – angefangen vom Byte, SByte, Felder mit einem Element davon, einzelne Bits, Felder von Bits mit bis zu 8 Feldelementen an unterschiedlichen Bitoffsets beginnend.

Der OPC-UA-Server unterstützt den Anwender deshalb mit den Template-Datenvariablen im DP-Namensraum. In einem für einen OPC-UA-Client typischen Texteingabefeld können diese Templates durch Ändern einiger weniger Zeichen in gültige ItemIDs verwandelt werden.

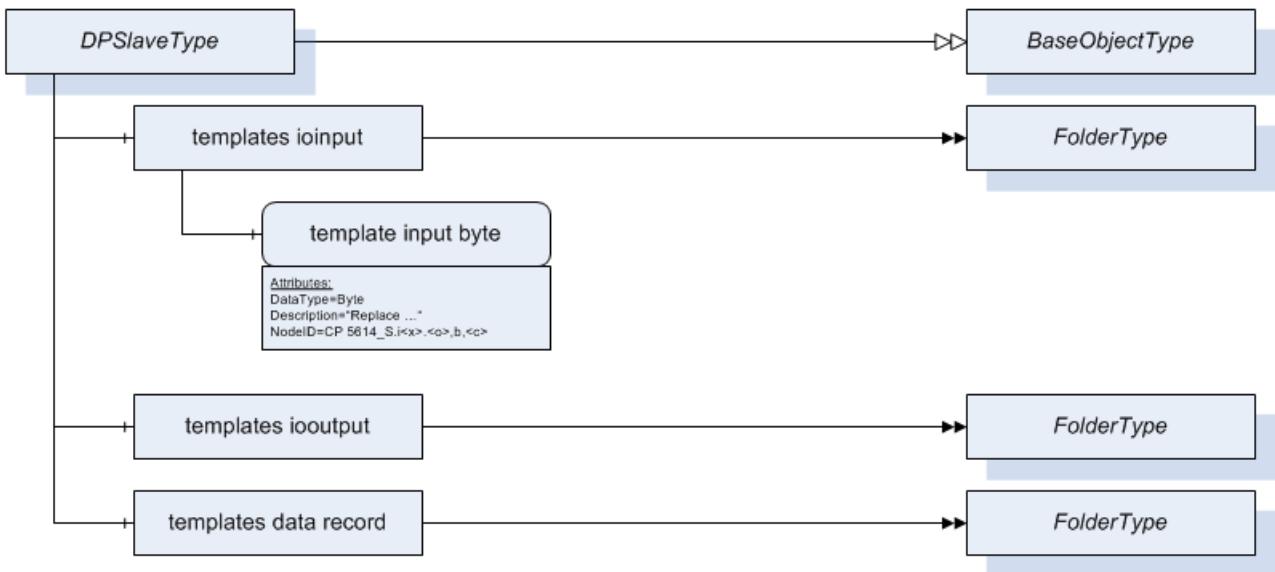


Bild 2-12 Template-Datenvariablen bei DP-OPC-UA

**Hinweis**

Die Verwendbarkeit von DP-OPC-UA-Template-Datenvariablen kann im Konfigurationsprogramm "Kommunikations-Einstellungen" unter "OPC-Protokollauswahl" > Klicken des Pfeilsymbols bei "DP" aktiviert und deaktiviert werden.

**Template-Datenvariablen innerhalb der Browse-Hierarchie**

Die Template-Datenvariablen sind neben den ihnen entsprechenden Ordnern in der Namensraum-Darstellung eingesortiert, so dass sie bei Bedarf leicht genutzt werden können.

**Syntax der Template-Datenvariablen**

Es gibt zwei Möglichkeiten:

- <DPMaster>.<Slave>.i<x>.<o>,<DPTypTemplate>,<c>
- <DPMaster>.<Slave>.q<x>.<o>,<DPTypTemplate>,<c>

**Erklärungen****<DPMaster>**

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt. Beim DP-Protokoll ist der Verbindungsname der projektierte Name der Kommunikationsbaugruppe.

**<Slave>**

Symbolischer Slave-Name als Kennzeichen für den Zugriff auf einen DP-Slave. Der symbolische Slave-Name kann in STEP 7 / NCM PC vergeben werden.

**Hinweis**

Wird in STEP 7 / NCM PC ein OPC-Server < V8.2 projektiert und kein symbolischer Slave-Name vergeben, dann wird als symbolischer Slave-Name der Bezeichner "slv" gefolgt von der projektierten Slave-Adresse verwendet.

Wird in STEP 7 / NCM PC ein OPC-Server >= V8.2 projektiert und kein symbolischer Slave-Name vergeben, dann wird die Bezeichnung des Slave als Slave-Name verwendet. Durch den Konsistenzcheck in STEP 7 / NCM PC wird sichergestellt, dass eindeutige Slave-Namen verwendet werden und der Slave-Name nicht leer ist.

**i**

Kennzeichen für einen Eingang. Eingänge sind les- und schreibbar.

**q**

Kennzeichen für einen Ausgang. Ausgänge sind nur lesbar.

**<x>**

Platzhalter für Nummer des Moduls, das den Ein- oder Ausgangsbereich enthält.

**<o>**

Platzhalter für Byteoffset im Adressraum des DP-Slave.

**<DPTypTemplate>**

Ein DP-Template-Datentyp wird im OPC-UA-Server in den entsprechenden OPC-UA-Datentyp umgewandelt. Die folgende Tabelle listet den Typ-Bezeichner und den entsprechenden OPC-Datentyp auf, in dem der Variablenwert dargestellt werden kann.

Datentyp	OPC-UA-Datentyp	Hinweis
x<bit>	Boolean	Platzhalter für den Bitoffset (0...7)
b	Byte	Platzhalter für Byte (unsigned)
w	UInt16	Platzhalter für Wort (unsigned)
dw	UInt32	Platzhalter für Doppelwort (unsigned)
lw	UInt64	Platzhalter für Langwort (unsigned)
c	SByte	Platzhalter für Byte (signed)
i	Int16	Platzhalter für Wort (signed)
di	Int32	Platzhalter für Doppelwort (signed)
li	Int64	Platzhalter für Langwort (signed)
r	Float	Platzhalter für Fließkomma (4 Byte)
lr	Double	Platzhalter für Fließkomma (8 Byte)
s<sl>	String	Platzhalter für die Länge des Strings

**<c>**

Platzhalter für Anzahl der Elemente.

**Beispiele:**

Nodeld	BrowseName	Beschreibung
CP 5624.slv<v>.i<x>.<o>,x<bit>,<c>	template input bit array	<v> PROFIBUS-Adresse des DP-Slave <x> Adresse des Moduls <o> Offset innerhalb des Moduls <bit> bitoffset (0 ... 7) <c> Größe des Feldes
CP 5624.slv<v>.i<x>.<o>,b,<c>	template input byte array	<v> PROFIBUS-Adresse des DP-Slave <x> Adresse des Moduls <o> Offset innerhalb des Moduls <c> Größe des Feldes
CP 5624.slv<v>.i<x>.<o>,w,<c>	template input word array	<v> PROFIBUS-Adresse des DP-Slave <x> Adresse des Moduls <o> Offset innerhalb des Moduls <c> Größe des Feldes
CP 5624.slv<v>.i<x>.<o>,s<sl>,<c>	template input string array	<v> PROFIBUS-Adresse des DP-Slave <x> Adresse des Moduls <o> Offset innerhalb des Moduls <sl> Länge des String <c> Größe des Feldes
CP 5624.slv<v>.q<x>.<o>,x<bit>,<c>	template output bit array	<v> PROFIBUS-Adresse des DP-Slave <x> Adresse des Moduls <o> Offset innerhalb des Moduls <bit> bitoffset (0 ... 7) <c> Größe des Feldes
CP 5624.slv<v>.dr<dr>,<l>.<o>,b	template data record byte	<v> PROFIBUS-Adresse des DP-Slave <x> Adresse des Moduls <dr> Datenaufzeichnungsnummer <l> Länge der Datei <o> Offset innerhalb der Datei
CP 5624_S.i<x>.<o>,b	template input byte	<x> Adresse des Moduls <o> Offset innerhalb des Moduls

## 2.6 S7-Kommunikation

Das S7-Protokoll für PROFIBUS und Industrial Ethernet dient der Kommunikation von Systemkomponenten innerhalb des Automatisierungssystems SIMATIC S7 und der Kommunikation von SIMATIC S7-Systemkomponenten mit Programmiergeräten und PCs.

## **Eigenschaften der S7-Kommunikation mit OPC**

Der OPC-Server von SIMATIC NET hat folgende Eigenschaften:

- **Variabeldienste**  
Zugriff und Beobachtung von S7-Variablen.
- **Blockorientierte Dienste**  
Programmgesteuerte Übertragung größerer Datenblöcke.
- **Serverfunktionalität**  
Der PC kann als Server für Datenblöcke und -bausteine eingesetzt werden.
- **Bausteindienste**  
Übertragung eines ladbaren Datenbereichs von und zu S7.
- **S7-Passwortfunktionen**  
Setzen eines Passworts zum Zugriff auf geschützte Bausteine.
- **Ereignisse**  
Verarbeitung von S7-Meldungen (S7-Alarms) als OPC Alarms & Events.

### **2.6.1 Performanter SIMATIC NET OPC-Server für das S7-Protokoll**

#### **Einleitung**

Der folgende Abschnitt beschreibt eine Konfigurationsvariante für das S7-Protokoll, die höhere Performanceanforderungen erfüllt. Hierfür werden alle unterlagerten S7-Protokollbibliotheken und der COM-Server als Inproc-Server in den Outproc-OPC-Server geladen. Die Protokollbearbeitung läuft im Prozess des OPC-Servers ab, weitere Laufzeiten für Prozesswechsel und Multiprotokollbetrieb fallen weg. Der Prozesswechsel zwischen OPC-Client und OPC-Server ist allerdings noch vorhanden.

## Konfiguration

Die Aktivierung dieser performanten Variante erfolgt implizit dadurch, dass im Konfigurationsprogramm "Kommunikations-Einstellungen" das Protokoll "S7" als einziges Protokoll ausgewählt wird (bei Auswahl weiterer Protokolle oder der OPC-UA-Schnittstelle entfällt der beschriebene Performance-Vorteil):

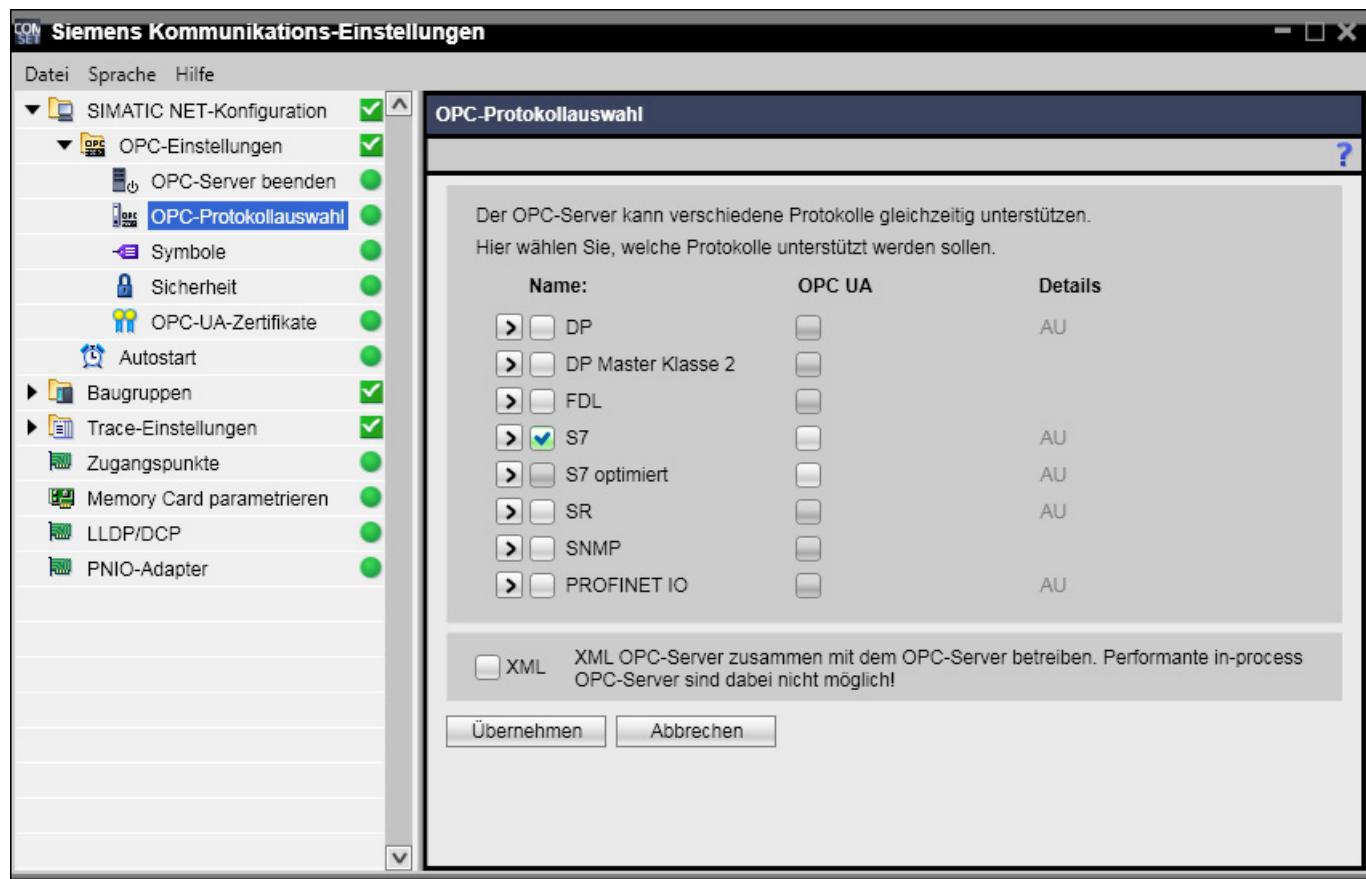


Bild 2-13 Fenster des Konfigurationsprogramms "Kommunikations-Einstellungen" zur Auswahl des S7-Protokolls

"Symbolik" darf zusätzlich ausgewählt werden.

### Hinweis

Für die Erstellung von Symbolen mit STEP 7 oder dem Symbol-Editor ist die Verwendung folgender Zeichen erlaubt: A-Z, a-z, 0-9, \_, -, ^, !, #, \$, %, &, ', /, (, ), <, >, =, ?, ~, +, \*, ',', :, |, @, [ , ], { , }, ". Zusätzlich sollten Sie bei der Erstellung von Symbolen mit STEP 7 darauf achten, dass es bei der Array-Auflösung zu Problemen kommen kann, wenn Ihre Symboldatei gleichzeitig Symbole der Form <Symbolname> und <Symbolname>[<Index>] enthält.

## Vorteile / Nachteile

Die Verwendung des performanten S7-OPC-Server hat jedoch den Nachteil, dass nur der Einzelprotokollbetrieb von S7 möglich ist. Dem stehen folgende Vorteile gegenüber:

- Höhere Performance als beim Multiprotokollbetrieb.
- Einfache Konfiguration.
- Mehrere Clients können den Server zur gleichen Zeit nutzen.
- Die Stabilität des OPC-Servers ist nicht von den Clients abhängig.

## Hinweise

### Hinweis

Achten Sie darauf, dass die Anzahl der projektierten S7-Verbindungen eines OPC-Servers die Anzahl gleichzeitig betreibbarer S7-Verbindungen nicht übersteigt, wenn in den Verbindungseigenschaften der OPC-Server als passiver Verbindungspartner konfiguriert ist (sogenannte Überprojektierung).

Hintergrund:

Wurden in der PC-Station mehr Verbindungen projektiert als zur selben Zeit zulässig sind, kann der OPC-Server nicht für alle Verbindungen einen Verbindungsrequest ausführen und der Verbindungsaufbau kann nicht für jede S7-Verbindung durchgeführt werden.

### Hinweis

S7-Verbindungen können zum Aufbau "bei Bedarf" in SIMATIC STEP 7/NCM PC projektiert werden.

Die Kommunikation der OPC-Funktionen wird dann erst bei Bedarf aufgebaut; somit kann der Verbindungsaufbau länger dauern und OPC in der Anlaufphase Fehler melden.

## 2.6.2 Protokoll-ID

Die Protokoll-ID für das S7-Protokoll lautet "S7".

## 2.6.3 Verbindungsnamen

### STEP 7-Verbindungen

Der Verbindungsname ist der in STEP 7 projektierte Name zur Identifikation der Verbindung. Dieser Name heißt bei STEP 7 "Lokale ID". Die Lokale ID ist innerhalb des OPC-Servers eindeutig.

Der OPC-Server unterstützt folgende Verbindungstypen:

- S7-Verbindung
- S7-Verbindung hochverfügbar

---

#### Hinweis

Zur Kompatibilität mit Version 2.2 und früher beachten Sie folgendes:

In älteren Versionen bestanden die Verbindungsinformationen aus den Angaben der S7-Verbindung, der VFDs und des Kommunikationsprozessors. Diese Elemente werden in der ItemID der Variablen weiterhin akzeptiert, sind jedoch nicht mehr im Namensraum vorhanden. Falls in einer Konfiguration mehrere VFDs vom OPC-Server verwendet wurden, können diese in STEP 7 ausschließlich aus Gründen der Kompatibilität projektiert werden.

---

### Welche Zeichen sind für S7-Verbindungsnamen erlaubt?

Für den <Verbindungsname> sind Ziffern "0-9", alphabetische Zeichen in Groß- und Kleinschreibung "A-z" und Sonderzeichen "\_-+()" erlaubt. Der Verbindungsname darf 24 Zeichen lang sein. Groß- und Kleinschreibung wird nicht unterschieden.

Weitere sichtbare Zeichen sind nicht erlaubt.

Die Verbindungsnamen "SYSTEM" bzw. der Verbindungsname "@LOCALSERVER" sind reserviert und dürfen nicht verwendet werden.

### Beispiele für Verbindungsnamen

Typische Beispiele sind:

- S7-Verbindung\_1
- S7-OPC-Verbindung

### Verbindungsprojekte

Es gibt folgende S7-Verbindungs-Objekte:

- Produktive S7-Verbindungen  
S7-Verbindungen werden zum Datenaustausch zwischen Automatisierungsgeräten genutzt und im Allgemeinen über STEP 7 projektiert.
- Die DEMO-Verbindung  
Sie dient ausschließlich zum Test.
- Die @LOCALSERVER-Verbindung  
Sie stellt die lokalen S7-Datenbausteine für die S7-Serverfunktionalität zur Verfügung.

## 2.6.4 Variabldienste

Variabldienste ermöglichen den Zugriff und die Beobachtung von S7-Variablen im Automatisierungsgerät. Die Adressierung der S7-Variablen erfolgt symbolisch. Die Art des Zugriffs orientiert sich an der Notation der S7-Werkzeuge.

### Objekte im Automatisierungsgerät

Der OPC-Server unterstützt folgende Objekte:

- Datenbausteine
- Instanzdatenbausteine
- Eingänge
- Ausgänge
- Peripherieeingänge
- Peripherieausgänge
- Merker
- Timer
- Zähler

Nicht jedes S7-Automatisierungsgerät unterstützt alle Objekttypen.

### 2.6.4.1 Syntax der Prozessvariablen für S7-Variabldienste

#### Syntax

Es gibt drei Möglichkeiten:

```
S7: [<Verbindungsname>] DB<Nr>, {<Typ>}<Adresse>{, <Anzahl>}
```

```
S7: [<Verbindungsname>] DI<Nr>, {<Typ>}<Adresse>{, <Anzahl>}
```

```
S7: [<Verbindungsname>]<Objekt>{<Typ>}<Adresse>{, <Anzahl>}
```

#### Erklärungen

##### S7

S7-Protokoll für den Zugriff auf die Prozessvariable.

##### <Verbindungsname>

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

##### DB

Datenbaustein. Kennzeichen für eine S7-Variable aus einem Datenbaustein.

##### DI

Instanzdatenbaustein. Kennzeichen für eine S7-Variable aus einem Instanzdatenbaustein.

**<Nr>**

Nummer des Datenbausteins oder Instanzdatenbausteins.

**<Objekt>**

Angabe des Bausteins/Bereichstyps in der S7.

Mögliche Werte sind:

<b>E</b>	Eingang
<b>A</b>	Ausgang
<b>PE</b>	Peripherieeingang
<b>PA</b>	Peripherieausgang
<b>M</b>	Merker
<b>T, TBCD, TDA</b>	Timer
<b>Z</b>	Zähler

**Zeitraster und Wertebereich von S7-Timer-Variablen (Typ T):**

Der Wertebereich von OPC-Prozessvariablen für S7 vom Typ Timer (T) ist dezimal in ms kodiert. Aus dem Wertebereich ergibt sich (für das Schreiben) das Zeitraster entsprechend der folgenden Tabelle:

Wertebereich	Zeitraster	Kommentar
1.000.000 ms bis 9.990.000 ms	10 s	Die Werte müssen Vielfache von 10.000 ms betragen.
100.000 ms bis 999.000 ms	1 s	Die Werte müssen Vielfache von 1.000 ms betragen.
10.000 ms bis 99.900 ms	100 ms	Die Werte müssen Vielfache von 100 ms betragen.
10 ms bis 9.990 ms	10 ms	Die Werte müssen Vielfache von 10 ms betragen. Ein feineres Raster ist nicht möglich. 0 ms sind erlaubt, aber ohne Funktion.

Die Anzahl der rechtsstehenden Nullen bestimmt das Zeitraster, die führenden Ziffern von 1 bis 999 multipliziert mit dem Zeitraster bestimmen die Laufdauer.

**Beispiele:**

Der Wert 90.000 ms hat ein Zeitraster von 100 ms.

Der Wert 123.000 ms hat ein Zeitraster 1 s.

Der Wert 150 ms hat ein Zeitraster von 10 ms.

Der OPC-Datentyp der S7-Timer-Variable vom Typ T ist ein Wort (signed, VT\_UI4).

**Zeitraster und Wertebereich von S7-Timer-Variablen (Typ TBCD):**

Der Wertebereich von OPC-Prozessvariablen für S7 vom Typ TBCD ist BCD-kodiert. Aus dem Wertebereich ergibt sich (für das Schreiben) das Zeitraster entsprechend der folgenden Tabelle:

<b>Bit-Nr.</b>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>Bedeutungssymbol</b>	0	0	x	x	z	z	z	z	z	z	z	z	z	z	z	z
<b>Erläuterung:</b>																
Bedeutungssymbol "0"	nicht relevant															

Bedeutungssymbol "x"	Angabe des Zeitrasters														
	Bit 13 und 12				Zeitraster in Sekunden										
	00				0,01										
	01				0,1										
	10				1										
	11				10										
Bedeutungssymbol "z"	BCD-kodierter Zeitwert (0...999)														

**Beispiel:**

Bit-Nr.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Wert	0	0	0	1	0	0	0	0	0	1	1	1	0	1	0	1

Bit 0-11 legen die Zahl 075 fest. Bit 12 und 13 legen das Zeitraster 0,1 fest.

$$75 * 0,1 = 0,75 \text{ Sekunden}$$

Der OPC-Datentyp der S7-Timer-Variablen vom Typ TBCD ist ein Wort (unsigned, VT\_UI2).

**Zeitraster und Wertebereich von S7-Timer-Variablen (Typ TDA):**

Mit der einfachen Timer-Variablen vom Typ T lassen sich Timer zwar einfach bedienen, es sind aber nicht alle möglichen Kombinationen von Zeitraster und Wertebereich einstellbar, da die Wertebereiche überlappen dürfen.

Für diesen Fall kann die S7-Timer-Variablen vom Typ TDA (Decimal Array) verwendet werden.

**<Objekt>: TDA**

Datentyp: Feld von zwei Worten {Zeitraster in ms VT\_UI2 / Zeitwert VT\_UI2}.

Wertebereich:

Zeitraster in ms: 10, 100, 1000, 10000.

Zeitwert: 0...999; 0 ist erlaubt, aber ohne Funktion.

Zeitbereiche: 10 ms: 0...9990ms; 100 ms: 0...99900ms; 1000 ms: 0...999000ms;  
10.000 ms: 0...9990000ms

Bei dem Objekt TDA ist keine Eingabe der <Anzahl> möglich.

**Beispiel:**

Beschreiben des Timers TDA3 mit Wert {100|50} initialisiert den Timer 3 mit dem Wert  $50 * 100\text{ms} = 5000 \text{ ms}$ , und dieser taktet 50-mal in 100 ms Stufen herunter.

Mit dem Typ T ist diese Einstellung nicht möglich.

**Wertebereich von S7-Zähler-Variablen (Typ Z):**

Der Wertebereich von OPC-Prozessvariablen für S7 vom Typ Zähler (Z) ist 0 bis 999, dezimal kodiert.

Der OPC-Datentyp der S7-Zähler-Variablen vom Typ Z ist ein Wort (unsigned, VT\_UI2).

**<Typ>**

S7-Datentyp.

Ein S7-Datentyp wird im OPC-Server in den entsprechenden OLE-Datentyp umgewandelt. Über die Automation-Schnittstelle von OPC können alle angegebenen OLE-Datentypen gelesen werden. Einige Entwicklungswerkzeuge (wie z. B. Visual Basic) bieten jedoch nur eine eingeschränkte Menge von Datentypen an. Die folgende Tabelle listet den entsprechenden Visual Basic-Typ auf, in dem der Variablenwert dargestellt werden kann.

Bei den Objekten *T*, *TDA* und *Z* ist die Angabe eines Typs nicht zulässig.

S7-Datentyp	Beschreibung	OLE-Datentyp	Visual Basic-Typ
X	Bit (bool). Sie müssen die Bit-Nummer (0...7) angeben.	VT_BOOL	Boolean
B	Byte (unsigned)	VT_UI1	Byte
W	Wort (unsigned)	VT_UI2	Long
D	Doppelwort (unsigned)	VT_UI4	Double
CHAR	Byte (signed)	VT_I1	Integer
INT	Wort (signed)	VT_I2	Integer
DINT	Doppelwort (signed)	VT_I4	Long
REAL	Fließkomma	VT_R4	Single
STRING	Zeichenfolge. Sie müssen die String-Länge angeben. Der String muss auf der S7 mit gültigen Werten initialisiert sein.	VT_BSTR	String
DT	Datum und Uhrzeit, 8 Byte BCD-Format (S7-CPU DATE-AND-TIME)	VT_DATE	Date
DATE	Datum und Uhrzeit, 8 Byte Die Uhrzeit ist immer 00:00:00, Wertebereich ab 01.01.1990. Abbildung des CPU Datentyps DATE (unsigned, 16 Bit).	VT_DATE	Date
TIME	Zeitwert (signed), IEC-Format, in ms	VT_I4	Long
TOD	Tageszeit (unsigned), 0...86399999 ms	VT_UI4	Double
S5TIMEBCD	Abbildung des CPU Datentyps S5TIME (unsigned, 16 Bit) mit eingeschränktem Wertebereich, 0...9990000 ms	VT_UI2	Long

**Zeitraster und Wertebereich für den S7-Datentyp S5TIMEBCD:**

Der Wertebereich der Zeit-Variable vom Datentyp S5TIMEBCD ist BCD-kodiert. Der Wertebereich ergibt sich entsprechend der folgenden Tabelle:

Bit-Nr.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bedeutungssymbol	0	0	x	x	z	z	z	z	z	z	z	z	z	z	z	z
<b>Erläuterung:</b>																
Bedeutungssymbol "0"	nicht relevant															

Bedeutungssymbol "X"	Angabe des Zeitrasters													
	Bit 13 und 12		Zeitraster in Sekunden											
	00		0,01											
	01		0,1											
	10		1											
	11		10											
Bedeutungssymbol "z"	BCD-kodierter Zeitwert (0...999)													

**Beispiel:**

Bit-Nr.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Wert	0	0	0	1	0	0	0	0	0	1	1	1	0	1	0	1

Bit 0-11 legen die Zahl 075 fest. Bit 12 und 13 legen das Zeitraster 0,1 fest.

$$75 * 0,1 = 0,75 \text{ Sekunden}$$

Der OPC-Datentyp der Zeit-Variable vom Datentyp S5TIMEBCD ist ein Wort (unsigned, VT\_UI2). Beim Schreiben ist der Wertebereich entsprechend eingeschränkt.

**<Adresse>**

Adresse der ersten Variablen, die angesprochen werden soll. Mögliche Werte sind:

- Byte-Offset
- Byte-Offset.Bit (nur für Datentyp X)  
Der Wertebereich der Bit-Adresse erlaubt dann 0...7.
- Byte-Offset.String-Länge (nur für Datentyp String, String-Länge 1 Byte bis 254 Byte)

Der Wertebereich für den Byte-Offset ist 0...65534. Geräte- und Typabhängig kann der tatsächlich verwendbare Wert der Adresse geringer sein.

**<Anzahl>**

Anzahl der Variablen eines Typs, die ab dem im Parameter *Adresse* angegebenen Offset angesprochen werden sollen. Der Wertebereich ist von der Projektierung abhängig.

Beim Datentyp X ist die Eingabe der Anzahl für Schreibzugriff nur in Vielfachen von 8 möglich. Die Bit-Adresse muss dann Null sein.

Beim Datentyp X ist die Eingabe der Anzahl für Lesezugriff nicht eingeschränkt.

**Beispiele:**

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

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

**2.6.4.2****Beispiele für Prozessvariablen für S7-Variablen-Dienste**

Hier finden Sie Beispiele, die die Syntax von Variablennamen für Variablen-Dienste verdeutlichen.

## Parameter Datenbaustein

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

DB5,B12

bezeichnet Datenbyte 12 im Datenbaustein 5.

## Parameter Instanzdatenbaustein

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

DI5,W10,9

bezeichnet 9 Datenwörter ab Byteadresse 10 im Instanzdatenbaustein 5.

## Parameter Objekte

*S7:[S7-Verbindung-1]EB0*

EB0

bezeichnet das Eingangsbyte 0.

### 2.6.4.3 Beispiele optimal strukturierter Items

Für die Variabeldienste S7 bietet der OPC-Server einen Optimierungsalgorithmus:

- OPC-Items, die gleichzeitig gelesen oder beobachtet werden, sollten im Namensraum des Partnergeräts aufeinanderfolgend angeordnet sein. Kleinere Lücken zwischen den relevanten Teilen werden zwar verarbeitet, verschlechtern jedoch den Datendurchsatz.
- OPC-Items, die gleichzeitig geschrieben werden, müssen im Namensraum aufeinanderfolgend angeordnet sein. Wenn der Schreibzugriff optimal erfolgen soll, dürfen keine Lücken vorhanden sein.

Wenn Ihr OPC-Client nicht auf Verwendung einzelner OPC-Items für die Prozessvariablen angewiesen ist, können Sie alternativ direkt Felder zum Zugriff auf die relevanten Daten verwenden und die Elemente der gelesenen Felder einzelnen Prozessvariablen zuweisen.

## Organisation eines Datenbausteins bei Lesezugriff

DB10,W10

DB10,B12

DB10,B13

DB10,DW14

DB10,W20

Wird über das Kommunikationssystem als ein lesender Feldzugriff auf DB10,B10,12 ausgeführt.

Trotz der Lücke an Byte 18 und 19 erfolgt die Umwandlung in einen einzigen Leseauftrag. Die unnötig gelesenen Daten werden verworfen. Anstelle von 5 einzelnen Variablenanforderungen wird nur eine Anforderung über das Netz transportiert.

### **Organisation eines Datenbausteins bei Schreibzugriff**

DB10,W10

DB10,B12

DB10,B13

DB10,DW14

DB10,W20

Wird über das Kommunikationssystem als zwei schreibende Zugriffe auf das Feld DB10,B10,8 und die einzelne Variable DB10,W20 ausgeführt.

### **2.6.5 Blockorientierte Dienste**

Blockorientierte Dienste ermöglichen eine programmgesteuerte Übertragung größerer Datenblöcke. Die Übertragung wird durch Variablen realisiert:

- Variablen, die Datenblöcke empfangen
- Variablen, die Datenblöcke senden

Die Datenmenge beträgt bei dem Datentransfer bis zu 65534 Byte, unabhängig von der Größe der PDU. Die Segmentierung der Daten wird von den Funktionen selbst übernommen.

---

#### **Hinweis**

Blockorientierte Dienste können nur bei zweiseitigen Verbindungen verwendet werden. Die erstellte Verbindungsprojektierung muss in das S7-Automatisierungsgerät bzw. auf die betreffenden PCs geladen werden.

---

## Beispiel für die Verwendung von Blockdiensten

Die folgende Abbildung zeigt, wie ein S7-400 Gerät ein Datenpaket an eine PC-Station mit S7-OPC-Server sendet.

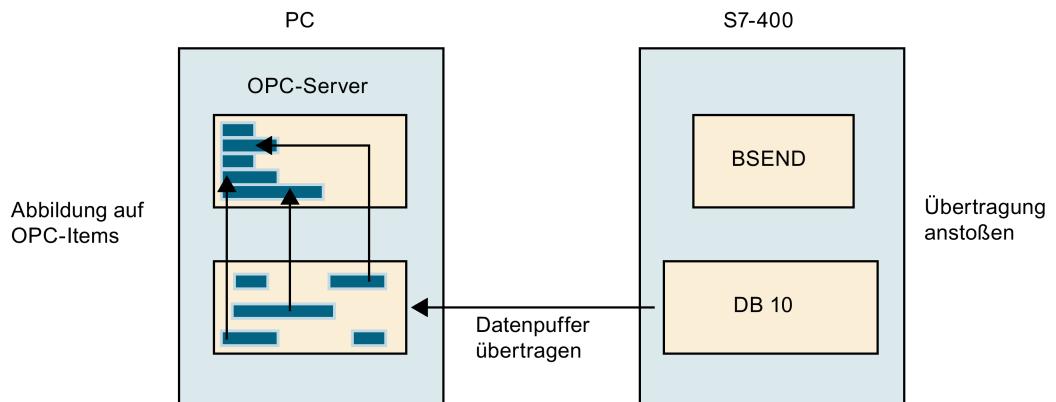


Bild 2-14 Senden eines Datenpakets von einem S7-400 Gerät an eine PC-Station mit S7-OPC-Server

Damit der OPC-Server Daten empfangen kann, werden im PC ein oder mehrere OPC-Items des Typs *BRCV* in eine aktive Gruppe eingefügt.

Im S7-400 Gerät stößt ein Anwendungsprogramm den Funktionsbaustein *BSEND* an. *BSEND* startet die Übertragung des gesamten Datenbereichs als Puffer zum PC.

Im PC werden die empfangenen Daten an den OPC-Server übergeben. Der OPC-Server bildet nun die Teilbereiche des Datenblocks auf die entsprechenden OPC-Items ab. Wenn diese OPC-Items beobachtet werden, sendet der OPC-Server bei Änderung der Werte einen Rückruf an den OPC-Client.

### 2.6.5.1 Syntax der Prozessvariablen für Blockorientierte Dienste

#### Syntax

Es gibt zwei Möglichkeiten:

```
S7: [<Verbindungsname>] BRCV,<RID>{ , {<Typ>}<Adresse>{ , <Anzahl>} }
```

```
S7: [<Verbindungsname>] BSEND<Länge>,<RID>{ , {<Typ>}<Adresse>{ , <Anzahl>} }
```

#### Erklärungen

##### S7

S7-Protokoll für den Zugriff auf die Prozessvariable.

**<Verbindungsname>**

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

**BRCV**

BRCV enthält den zuletzt vom Partner empfangenen Datenblock. Der Inhalt und die Länge der Empfangsdaten wird durch den sendenden Partner vorgegeben.

Die Variable ist nur lesbar.

Setzen Sie diese Variable zur Beobachtung ein. Damit wird der Empfang eines Datenblocks durch OPC-Server an eine OPC-Client-Anwendung gemeldet.

Ein expliziter Auftrag zum Lesen dieser Variablen kehrt zurück, wenn eine verbindungs-spezifische und in der Projektierung angegebene Fehlerwartezeit abgelaufen ist oder ein Datenblock empfangen wurde.

OLE-Datentyp	Visual Basic Typ
VT_ARRAY   VT_UI1	Byte()

**BSEND**

BSEND enthält den Sendedatenblock zur Übertragung an ein Partnergerät.

Der Datenblock wird erst dann an das Partnergerät übertragen, wenn die Variable geschrieben wird. Bei einem Lesezugriff wird der Inhalt des zuletzt erfolgreich übertragenen Datenblocks geliefert.

OLE-Datentyp	Visual Basic Typ
VT_ARRAY   VT_UI1	Byte()

**<Länge>**

Länge des Datenblocks in Bytes, der gesendet werden soll.

**<RID>**

ID des Adressierungsparameters. Sie ist für ein Bausteinpaar (*BSEND/BRCV*) festgelegt und innerhalb einer Verbindung eindeutig definiert.

Sie können mehrere *BSEND*-Bausteine über eine Verbindung senden bzw. mehrere *BRCV*-Bausteine empfangen, aber immer mit einer unterschiedlichen ID. Gleiche IDs können für weitere Verbindungen verwendet werden. Der Wertebereich bei BSEND/BRCV ist 0...4294967295.

**<Typ>**

S7-Datentyp.

Ein S7-Datentyp wird im OPC-Server in den entsprechenden OLE-Datentyp umgewandelt.

Über die Automation-Schnittstelle von OPC können alle angegebenen OLE-Datentypen gelesen werden. Jedoch bieten einige Entwicklungswerkzeuge (wie z.B. Visual Basic) nur eine eingeschränkte Menge von Datentypen an. Die folgende Tabelle listet deshalb den entsprechenden Visual Basic-Typ auf, in dem der Variablenwert dargestellt werden kann.

Format-bezeichner	Beschreibung	OLE-Datentyp	Visual Basic-Typ
X	Bit (bool). Sie müssen die Bit-Nummer angeben.	VT_BOOL	Boolean
BYTE oder B	Byte (unsigned)	VT_UI1	Byte
WORD oder W	Wort (unsigned)	VT_UI2	Long
DWORD oder D	Doppelwort (unsigned)	VT_UI4	Double
CHAR	Byte (signed)	VT_I1	Integer
INT	Wort (signed)	VT_I2	Integer
DINT	Doppelwort (signed)	VT_I4	Long
REAL	Fließkomma	VT_R4	Single
STRING	Zeichenfolge. Sie müssen die Stringlänge angeben	VT_BSTR	String
DT	Datum und Uhrzeit, 8 Byte BCD-Format	VT_DATE	Date
TIME	Zeitwert (signed), IEC-Format, in ms	VT_I4	Long
TOD	Tageszeit (unsigned), 0...86399999 ms	VT_UI4	Double

#### <Adresse>

Adresse der ersten Variablen, die angesprochen werden soll.

Mögliche Werte sind:

Byte-Nummer

Byte-Nummer.Bit (nur für Format X)

Der Wertebereich für die Byte-Nummer ist 0...65534. Geräte- und Typabhängig kann der tatsächlich verwendbare Wert der Adresse geringer sein.

#### <Anzahl>

Anzahl der Variablen eines Typs, die ab der im Parameter *Adresse* angegebenen Adresse angesprochen werden sollen (Wertebereich 0...65535).

---

#### Hinweis

Das Lesen (BRCV) und Schreiben (BSEND) einzelner Bits (Format X) ist möglich. Weiterhin ist das Lesen und Schreiben von Feldern einzelner Bits ohne Einschränkung der Beginn-Bit-Adresse und beliebiger Feldlänge innerhalb des Datenblockes möglich.

**Beispiel:** *S7:[S7-OPC-1]BRCV,1,X10.4,78*

#### Hinweis

Bitte beachten Sie folgende Hinweise:

- Durch die optionale Angabe von Typ, Adresse und Anzahl können Sie strukturiert auf Teilbereiche von Datenblöcken zugreifen.
- Variablen für Sendedaten oder Empfangsdaten mit unterschiedlicher Länge oder unterschiedlicher RID oder unterschiedlichem Verbindungsnamen verfügen über unabhängige Speicherbereiche.
- Der Sendedatenpuffer wird allokiert und mit Null initialisiert, wenn ein Item für einen unabhängigen Speicherbereich angelegt wird. Ein Schreibauftrag auf ein BSEND-Item wird in einen internen Schreibpuffer geschrieben und übertragen.
- Die Übertragung der Datenblöcke erfolgt azyklisch. Parallele Netzaufträge für die gleichen Daten sind möglich. Es wird immer der vollständige Sendedatenblock übertragen. Dies gilt auch bei Subelementzugriff oder wenn mehrere Clients gleichzeitig dieses Item beschreiben.  
Ein paralleles Schreiben des gleichen Sendedatenblocks oder von Teilenbereichen des Sendedatenblocks durch mehrere Clients kann zu Inkonsistenzen führen. Es wird deshalb empfohlen,
  - immer den vollständigen Datenblock zu lesen oder zu schreiben,
  - in NCM S7 ist die "Maximale Anzahl paralleler Netzaufträge" auf "Eins" zu setzen, was jedoch die Übertragungsleistung reduziert.

#### 2.6.5.2

#### Beispiele für Prozessvariablen für Blockorientierte Dienste

Hier finden Sie Beispiele, die die Syntax von Variablennamen für blockorientierte Dienste verdeutlichen.

#### Datenblock empfangen im gesamten Puffer

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

BRCV,1

Ein Datenblock wird im Empfangspuffer mit der RID 1 empfangen. Der vollständige Puffer wird auf ein Feld von Bytes abgebildet.

#### Teilzugriff auf empfangenen Datenblock

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

BRCV,1,W2,4

Aus dem empfangenen Datenblock wird der Inhalt ab Offset 2 auf ein Feld aus 4 Wörtern abgebildet. Insgesamt werden also 8 Bytes aus dem Datenblock betrachtet.

#### Doppelwort übertragen

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

**BSEND16,1,D2**

In einem Datenblock der Länge 16 mit der RID 1 wird ab Offset 2 ein Doppelwort adressiert. Wird ein Schreibbefehl auf die Variable angewendet, so wird der geschriebene Wert an der angegebenen Position im Block eingetragen und der Datenblock wird gesendet.

**Fließkommazahlen übertragen**

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

**BSEND32,1,REAL20,2**

In einem Datenblock der Länge 32 mit der RID 1 wird ab Offset 20 ein Feld mit Fließkommazahlen adressiert. Wird ein Schreibbefehl auf die Variable angewendet, so wird der geschriebene Wert an der angegebenen Position im Block eingetragen und der Datenblock wird gesendet.

## 2.6.6 S7-spezifische Informationsvariablen

Es gibt S7-spezifische Variablen, mit denen Sie Informationen über die S7-Kommunikation und die aufgebauten Verbindungen abfragen können.

Folgende Informationen können ermittelt werden:

- Attribut eines virtuellen Geräts (VFD)
- Status einer S7-Verbindung
- Status eines virtuellen Geräts
- Konfigurierte Parameter und Laufzeitparameter der S7-Verbindung zur Diagnose

### 2.6.6.1 Syntax der S7-spezifischen Informationsvariablen

**S7-spezifische Informationsvariablen**

**Syntax:**

*S7 : [<Verbindungsname>] <Informationsvariable>*

**Erklärungen****S7**

S7-Protokoll für den Zugriff auf die Prozessvariable.

**<Verbindungsname>**

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

**<Informationsparameter>**

Mögliche Werte sind:

**&identify()**

Herstellerattribute eines Kommunikationspartners.

Rückgabewerte (als Elemente eines String-Feldes):

- *Hersteller*
- *Modell*
- *Revision*

OLE-Datentyp	Visual Basic Typ
VT_ARRAY of VT_BSTR	String()

#### **&vfdstate()**

Zustand eines virtuellen Geräts.

Datentyp: *VT\_ARRAY of VT\_VARIANT* mit folgenden Komponenten:

logischer Status

Dienste, die unterstützt werden.

Rückgabewert:

- *S7\_STATE\_CHANGES\_ALLOWED*  
alle Dienste sind zulässig

OLE-Datentyp	Visual Basic Typ
VT_BSTR	String

physikalischer Status

Einsatzbereitschaft des realen Geräts

Rückgabewerte:

- *S7\_OPERATIONAL*  
das reale Gerät ist einsetzbar
- *S7\_NEEDS COMMISSIONING*  
das reale Gerät ist erst nach dem Abschluss lokaler Eingriffe einsetzbar

OLE-Datentyp	Visual Basic Typ
VT_BSTR	String

Detailinformationen über den lokalen VFD-Status.

Der Status wird als Octett-String zurückgegeben. Weitere Informationen zur Bedeutung des Rückgabewertes müssen der Dokumentation des Partnergerätes entnommen werden.

OLE-Datentyp	Visual Basic Typ
VT_ARRAY of VT_BSTR	String()

#### **&statepath()**

Zustand einer Kommunikationsverbindung zu einem Partnergerät.

Rückgabewerte:

- *DOWN*  
Verbindung ist nicht aufgebaut
- *UP*  
Verbindung ist aufgebaut
- *RECOVERY*  
Verbindung wird aufgebaut
- *ESTABLISH*  
(für zukünftige Erweiterungen reserviert)

OLE-Datentyp	Visual Basic Typ
VT_BSTR	String

#### **&statepathval()**

Zustand einer Kommunikationsverbindung zu einem Partnergerät.

Rückgabewerte:

- *1*  
Verbindung ist nicht aufgebaut
- *2*  
Verbindung ist aufgebaut
- *3*  
Verbindung wird aufgebaut
- *4*  
(für zukünftige Erweiterungen reserviert)

OLE-Datentyp	Visual Basic Typ
VT_UI1	Byte

### **Beispiele für S7-spezifische Informationsvariablen und Rückgabewerte**

Hier finden Sie Beispiele, die die Syntax von S7-spezifischen Informationsvariablennamen verdeutlichen.

### **Informationen über die Herstellerattribute eines virtuellen Geräts**

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

### &identify()

kann beispielsweise folgende Werte zurückgeben:

- Hersteller: *SIEMENS AG*
- Modell des virtuellen Geräts: *6ES7413-1AE0-0AB0*
- Revision: *V1.0*

## Zustand eines Geräts

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

### &vfdstate()

kann beispielsweise folgende Werte zurückgeben:

- Logischer Status: *S7\_STATE\_CHANGES\_ALLOWED*
- Alle Dienste sind zulässig.
- Physikalischer Status: *S7\_OPERATIONAL*
- Das reale Gerät ist einsatzbereit.
- Detailinformationen: *02.00.00*
- Detailinformationen zum lokalen VFD-Status.

## Zustand einer Kommunikationsverbindung als String

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

### &statepath()

kann beispielsweise folgenden Wert zurückgeben:

- Verbindungsstatus: *RECOVERY*
- *Die Verbindung wird momentan aufgebaut.*

## Zustand einer Kommunikationsverbindung als Zahl

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

### &statepathval()

kann beispielsweise folgenden Wert zurückgeben:

- Verbindungsstatus: *2*
- *Die Verbindung ist aufgebaut.*

### 2.6.6.2

## S7-spezifische Diagnosevariablen

Zur Überprüfung der Projektierungsparameter und aktueller Laufzeitparameter zu einer S7-Verbindung werden folgende Diagnosevariablen zur Verfügung gestellt. Im OPC Scout V10 gibt es hierfür eine separate Diagnoseansicht, zusätzlich werden die Variablen im Namensraum im Ordner "Configuration" unter der jeweiligen S7-Verbindung angezeigt.

## Syntax

S7: [<Verbindungsname>] <Diagnosevariable>

Diagnosevariable	Beschreibung
&vfd()	Name des OPC-Servers, dem die Verbindung zugeordnet ist. Üblicherweise hat dieses bei NCM-projektierten Verbindungen den Text "OPC Server". Datentyp VT_BSTR, nur lesbar.
&cp()	Name der Schnittstellenparametrierung, dem die Verbindung zugeordnet ist. Datentyp VT_BSTR, nur lesbar.
&remoteaddress()	Adresse des Verbindungspartners. Datentyp VT_BSTR, nur lesbar. Die Adresse des Verbindungspartners ist ein Datenpuffer mit einer vom Verbindungs-typ abhängigen Datenlänge. Für die übersichtlichere Auswertung durch den Anwender wird der Datenpuffer formatiert in einem String dargestellt. Profibus-Adresse Format: "ddd" (1-3 dezimale Ziffern) IP-Adresse (ISOonTCP) Format: "ddd.ddd.ddd.ddd" (je 1-3 dezimale Ziffern) MAC-Adresse (ISO) Format: "xx-xx-xx-xx-xx-xx" (je 2 hexadezimale Ziffern)
&localsap()	Lokaler SAP der Verbindung. Datentyp VT_BSTR, nur lesbar. Der lokale SAP des Verbindungspartners ist ein Datenpuffer mit einer vom Verbin-dungstyp abhängigen Datenlänge. Für die übersichtlichere Auswertung durch den Anwender wird der Datenpuffer formatiert in einem String dargestellt. Format: "xx.xx " (je 2 hexadezimale Ziffern)
&remotesap()	Remoter SAP der Verbindung. Datentyp VT_BSTR, nur lesbar. Der remote SAP des Verbindungspartners ist ein Datenpuffer mit einer vom Verbin-dungstyp abhängigen Datenlänge. Für die übersichtlichere Auswertung durch den Anwender wird der Datenpuffer formatiert in einem String dargestellt. Format: "xx.xx " (je 2 hexadezimale Ziffern)
&connect()	Art des Verbindungsaufbaus. Datentyp VT_UI4, nur lesbar. 0   Passiv, Verbindung wird permanent aufrecht erhalten. 1   Aktiv, Verbindungsaufbau wird erst bei Bedarf hergestellt, Verbindungsabbau ohne Benutzung nach Wartezeit. 2   Aktiv, Verbindung wird permanent aufrecht erhalten.
&abortconnectionafter()	Automatischer Verbindungsabbau. Verzögerungszeit für den automatischen Verbindungsabbau: Der OPC-Server baut die Verbindung nach dieser Zeit selbstständig wieder ab, sofern in dieser Zeit kein erneu-ter Variablenzugriff erfolgt. Auf diese Weise können bei Zugriffen auf Variablen in sehr großen Zeitabständen die Anzahl der benötigten Verbindungen reduziert werden. Datentyp VT_UI4, nur lesbar. 0: kein Abbau >0: Leerlaufzeit bis zum Abbau in ms

Diagnosevariable	Beschreibung
&optimizes7read()	Lesezugriffe für S7-Bausteinzugriff werden optimiert. Datentyp VT_BOOL, nur lesbar. "True": Optimierung "False": keine Optimierung Optimierung bedeutet: Mehrere Zugriffsanträge auf einzelne Variablen werden intern in einen einzigen Feldzugriff auf den Kommunikationspartner umgewandelt.
&optimizes7write()	Schreibzugriffe für S7-Bausteinzugriff werden optimiert. Datentyp VT_BOOL, nur lesbar. "True": Optimierung "False": keine Optimierung Optimierung bedeutet: Mehrere Zugriffsanträge auf einzelne Variablen werden intern in einen einzigen Feldzugriff auf den Kommunikationspartner umgewandelt.
&autopasswordreset()	Automatisches Rücksetzen des S7-Passworts zum Bausteinzugriff Datentyp VT_BOOL, nur lesbar. "True": Rücksetzen aktiviert "False": Rücksetzen deaktiviert In einer S7 bleibt eine Freischaltung der Domäindienste durch Passwort bis zu einem expliziten Rücksetzen aktiviert. Ein automatisches Rücksetzen des Passworts bei Verbindungsaufbau sorgt besonders im Zusammenwirken mit dem automatischen Verbindungsabbau nach unbenuztter Zeit dafür, dass das Passwort nicht unnötig lange freigegeben wird.
&fastconnectionstate-returnenable()	Schnelle Rückgabe eines Schreib/Lesezugriffs bei unterbrochener Verbindung. Datentyp VT_BOOL, nur lesbar. "True": aktiviert "False": deaktiviert
&connecttimeout()	Verbindungsaufbau-Timeout Datentyp VT_UI4, nur lesbar 0: kein Timeout >0: Timeout in ms
&timeout()	Auftrags-Timeout für Produktivaufräge in der S7-Kommunikation. Datentyp VT_UI4, nur lesbar 0: kein Timeout >0: Timeout in ms
&receivecredit()	Maximale Anzahl paralleler Netzaufträge, Empfangsrichtung Vorschlagswert für Verbindungsaufbau Datentyp VT_UI2, Nur lesbar >=1, Vorschlagswert für Verbindungsaufbau
&tradedreceivecredit()	Anzahl paralleler Protokollaufräge, Empfangsrichtung ausgehandelt nach Verbindungsaufbau. Datentyp VT_UI2, nur lesbar Ist die Verbindung unterbrochen, so ist die Qualität dieser Variablen "BAD".

Diagnosevariable	Beschreibung
&sendcredit()	Maximale Anzahl paralleler Netzaufträge, Senderichtung Vorschlagswert für Verbindungsaufbau Datentyp VT_UI2, nur lesbar >=1, Vorschlagswert für Verbindungsaufbau Wird gemeinsam mit &receivecredit() über die Projektierung eingestellt.
&tradedsendcredit()	Anzahl paralleler Protokollaufräge, Senderichtung ausgehandelt nach Verbindungs- aufbau. Datentyp VT_UI2, nur lesbar Ist die Verbindung unterbrochen, so ist die Qualität dieser Variablen "BAD".
&pdusize()	Größe der Protokoll-PDU Vorschlagswert für Verbindungsaufbau Datentyp VT_UI2, Nur lesbar >=1, Vorschlagswert für Verbindungsaufbau
&tradedpdusize()	Größe der Protokoll-PDU, ausgehandelt nach Verbindungs- aufbau. Datentyp VT_UI2, nur lesbar Ist die Verbindung unterbrochen, so ist die Qualität dieser Variablen "BAD".
&defaultalarmseverity()	Vorgegebene Alarmseverity für unprojektierte Alarmereignisse. Datentyp VT_UI2, nur lesbar 1: niederprior ... 1000: hochprior In der Projektierung gibt es eine Einstellmöglichkeit für Vorgabe-Priorität von Meldungen für S7-Meldungen und S7-Diagnosemeldungen.
&defaultdiagnosiseverity()	Vorgegebene Alarmseverity für unprojektierte Diagnoseereignisse. Datentyp VT_UI2, Nur lesbar 1: niederprior ... 1000: hochprior In der Projektierung gibt es eine Einstellmöglichkeit für Vorgabe-Priorität von Meldungen für S7-Meldungen und S7-Diagnosemeldungen.
&events()	Anmeldung von Alarmen und Events beim Verbindungspartner. Datentyp VT_UI4, nur lesbar. Die einzelnen Werte können kombiniert werden.
	0x000000 01      SCAN-Item (nicht mehr unterstützt)
	0x000000 02      Einfache Meldungen (nicht mehr unterstützt)
	0x000000 04      Einfache symbolbezogen Meldungen (nicht mehr unterstützt)
	0x000000 08      Simotion TO-Alarme
	0x000000 10      Verbindungsüberwachungs-Meldungen
	0x000000 20      Bausteinbezogene Meldungen (als Conditional Events)

Diagnosevariable	Beschreibung	
	0x000000 40	Symbolbezogene Meldungen (als Conditional Events)
	0x000000 80	Diagnosemeldungen
&connectiontype()	S7-Verbindungstyp Datentyp VT_UI2, nur lesbar 2:S7D_STD_TYPE; Standardverbindung 3:S7D_H_TYPE; hochverfügbare Verbindung Ist die S7-Verbindung noch nicht aufgebaut, wird für dieses Item die Qualität "BAD" gemeldet und die Werte sind ungültig.	
&connectionstate()	S7-Verbindungszustand Datentyp VT_UI2, nur lesbar 0x11:STD_DOWN; Standardverbindung ist gewollt abgebaut 0x12:STD_ABORT; Standardverbindung wurde ungewollt abgebaut (Fehler) 0x13:STD_NOT_USED; Standardverbindung noch nie aufgebaut 0x14:STD_OK; Standardverbindung aufgebaut 0x20:H_OK_RED; hochverfügbare Verbindung aufgebaut (redundant) 0x21:H_OK_RED_PATH_CHG; hochverfügbare Verbindung aufgebaut (redundant, es wurde umgeschaltet) 0x22:H_OK_NOT_RED; hochverfügbare Verbindung nicht redundant aufgebaut 0x23:H_ABORT; hochverfügbare Verbindung wurde ungewollt abgebaut (Fehler) 0x24:H_NOT_USED; hochverfügbare Verbindung noch nie aufgebaut 0x25:H_DOWN; hochverfügbare Verbindung ist gewollt abgebaut Die Datenvariable unterscheidet sich von der &statepath()-Variable dahingehend, dass sie den Verbindungszustand aus Sicht des Protokollstacks wiedergibt und dabei zusätzliche Informationen für die für den OPC-Server transparent benutzten redundanten H-Verbindungen liefert.	
&hconnectionwaystate()	Zustand der H-Verbindungswege Datentyp VT_ARRAY   VT_UI2 Die Werte beschreiben jeweils den Zustand eines Weges der H-Verbindung. 0x30:HW_PROD; Weg ist Produktivverbindung 0x31:HW_STBY; Weg ist Standby-Verbindung 0x32:HW_ABORT; Weg wurde ungewollt abgebaut (Fehler) 0x33:HW_NOT_USED; Weg wurde noch nie aufgebaut 0x34:HW_DOWN; Weg wurde gewollt abgebaut 0x35:HW_CN_BREAK; Weg konnte nicht aufgebaut werden Ist die S7-Verbindung noch nicht aufgebaut, wird für dieses Item die Qualität "BAD" gemeldet und die Werte sind ungültig.	

### 2.6.6.3 Syntax der systemspezifischen Informationsvariablen

S7: [SYSTEM] &version()

#### Hinweis

Der in STEP 7 projektierte Verbindungsname darf *nicht* "SYSTEM" heißen, da dies ein reservierter Bezeichner ist.

#### &version()

Liefert eine Versionskennung für den S7-OPC-Server, hier die Zeichenfolge, z.B.  
SIMATIC NET Core Server S7 V7.xxxx.yyyy.zzzz Copyright 2012

Datentyp: VT\_BSTR

Zugriffsrecht: nur lesbar

S7: [SYSTEM] &sapiversion()

#### &sapiversion()

Liefert eine Versionskennung der SAPI S7-Anwenderschnittstelle.

Datentyp: VT\_BSTR

Zugriffsrecht: nur lesbar

## 2.6.7 Bausteindienste

Bausteindienste steuern die Übertragung und Rückübertragung von Daten- und Programmelementen zwischen PC und Automatisierungsgerät. Auf dem PC werden Daten- und Programmelemente in Dateien gespeichert. Bausteindienste können mit einem Passwort geschützt werden.

### Bausteine

Daten und Programmelemente werden auf S7-Automatisierungsgeräten in Bausteinen abgelegt. Diese Bausteine werden mit Hilfe von STEP 7 generiert und auf das S7-Gerät übertragen. Bei S7-Geräten gibt es folgende Bausteine:

- Organisationsbausteine
- Funktionsbausteine
- Datenbausteine (DB / DI)

### Aufgaben mit Bausteinen

Sie können folgende Aufgaben über den SIMATIC NET OPC-Server ausführen:

- Bausteine zwischen PC und Automatisierungsgerät übertragen
- Bausteine löschen

- Bausteine einketten
- Speicher des Automatisierungsgeräts komprimieren

#### Hinweis

Die Erstellung von Bausteinen ist nicht mit dem OPC-Server möglich, Sie müssen STEP 7 verwenden.

### 2.6.7.1 Syntax der Steuervariablen für Bausteindienste

#### Syntax

S7: [<Verbindungsname>]<Dienstparameter>

#### Erklärungen

##### S7

S7-Protokoll für den Zugriff auf die Prozessvariable.

##### <Verbindungsname>

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

##### <Dienstparameter>

Diese Variablen sind nur schreibbar. Durch das Schreiben eines Wertes wird der Dienst ausgelöst.

Mögliche Werte sind:

##### &blockread()

Baustein von einem Automatisierungsgerät wird auf den PC übertragen und dort in einer Datei abgelegt.

Der geschriebene Wert enthält die Parametrierung des Dienstes. Er wird als Array of Variant mit folgenden Elementen dargestellt:

##### Flags

Folgende hexadezimale Zahlen sind möglich:

0x0001	Ein nicht eingeketteter Block wird gelesen. Wenn eine Zielfile besteht, wird diese nicht überschrieben, es wird eine Fehlermeldung ausgegeben.
0x0040	Ein eingeketteter Block wird gelesen. Wenn eine Zielfile besteht, wird diese nicht überschrieben, es wird eine Fehlermeldung ausgegeben.
0x1001	Ein nicht eingeketteter Block wird gelesen. Eine schon vorhandene Zielfile wird überschrieben.
0x1040	Ein eingeketteter Block wird gelesen. Eine schon vorhandene Zielfile wird überschrieben.

OLE-Datentyp	Visual Basic Typ
VT-BSTR	String

*Block*

Bausteintyp und Nummer

OB	Organisationsbaustein
FB	Funktionsbaustein
FC	Funktion
DB	Datenbaustein

OLE-Datentyp	Visual Basic Typ
VT-BSTR	String

*Datei*

Vollständiger Pfad der Datei, in der der Baustein abgelegt werden soll

OLE-Datentyp	Visual Basic Typ
VT-BSTR	String

**&blockwrite()**

Baustein von einem PC auf ein Automatisierungsgerät übertragen. Nach der Übertragung befindet sich der Baustein auf dem Automatisierungsgerät im passiven (nicht eingeketteten) Zustand. Mit der Funktion **&blocklinkin** muss der Baustein vom passiven Zustand in den eingeketteten Zustand gesetzt werden.

Der geschriebene Wert enthält die Parametrierung dieses Dienstes. Er wird als Array of Variant mit folgenden Elementen dargestellt:

*Flags*

Folgende hexadezimale Zahlen sind möglich:

0x1000	Ein auf dem Automatisierungssystem bestehender nicht eingeketteter Baustein gleichen Namens soll überschrieben werden.
0x0000	Ein auf dem Automatisierungssystem bestehender nicht eingeketteter Baustein gleichen Namens soll nicht überschrieben werden.

OLE-Datentyp	Visual Basic Typ
VT-BSTR	String

*Datei*

Vollständiger Pfad der Datei, in der der Baustein abgelegt ist

OLE-Datentyp	Visual Basic Typ
VT-BSTR	String

---

**Hinweis**

Die Domain-Dienste *blockread()* und *blockwrite()* erlauben keinen Zugriff auf Quell- und Zieldateien der Bausteine auf einem Netzlaufwerk.

Das Überschreiben (0x1000) eines Blocks ist erst möglich, nachdem der ablaufrelevante Teil eingekettet wurde.

---

**&blocklinkin()**

Baustein vom passiven Zustand in den Programmablauf des Automatisierungssystems einketten. Dabei wird der ablaufrelevante Teil des Bausteins in den Ablaufspeicher des Automatisierungssystems kopiert.

Der Baustein ist danach für das Programm auf dem Automatisierungssystem erreichbar. Durch das Einketten eines Bausteins wird ein bestehender, aktiver Baustein ohne Fehlermeldung überschrieben.

Der geschriebene Wert erhält als Parametrierung dieses Dienstes die Angabe des Blocks, der eingekettet werden soll.

*Block*

Bausteintyp und Nummer.

Folgende Typen sind möglich:

OB	Organisationsbaustein
FB	Funktionsbaustein
FC	Funktion
DB	Datenbaustein

OLE-Datentyp	Visual Basic Typ
VT-BSTR	String

---

**Hinweis**

Zwischen einem *blockwrite()* und einem *blocklinkin()* darf die S7-Verbindung nicht abgebaut werden, da sonst die Bausteine auf der remoten CPU sofort wieder gelöscht werden. Die *blocklinkin()* Funktion funktioniert dann nicht.

---

**&blockdelete()**

Löschen eines Bausteins im Automatisierungsgerät. Es können sowohl die passiven, als auch die in den Programmablauf eingeketteten Bausteine entfernt werden.

Der geschriebene Wert enthält die Parametrierung dieses Dienstes. Er wird als Array of Variant mit folgenden Elementen dargestellt:

*Flags*

Folgende hexadezimalen Zahlen sind möglich:

0x0001	Ein nicht eingeketteter Block wird gelöscht
0x0040	Ein eingeketteter Block wird gelöscht
0x0041	Der eingekettete und der nicht eingekettete Block werden gelöscht

OLE-Datentyp	Visual Basic Typ
VT-BSTR	String

**Block**

Bausteintyp und Nummer.

Folgende Typen sind möglich:

OB	Organisationsbaustein
FB	Funktionsbaustein
FC	Funktion
DB	Datenbaustein

OLE-Datentyp	Visual Basic Typ
VT-BSTR	String

**&blockcompress()**

Komprimierung des Speichers des Automatisierungsgeräts.

Fragmentierte Speicherbereiche werden zusammengefasst und es entsteht Platz für die Übertragung neuer Blöcke.

Diese Variable ist nur schreibbar. Durch das Schreiben eines leeren Strings wird der Dienst ausgelöst.

**Hinweis**

Einige der folgenden Bausteindienste werden durch den geschriebenen Wert parametert. Dabei werden Felder als Datentyp verwendet. Es kann aus Anwendersicht günstig sein, wenn die Parameter als Wert im Datentyp *VT\_BSTR* übergeben werden. Der OPC-Server führt selbstständig die Konvertierung in den benötigten Datentyp durch. Die Darstellung eines Feldes in einem String ist beispielsweise:

*{ErstesElement/ZweitesElement/DrittesElement}*.

Beachten Sie dazu auch die Beispiele.

### 2.6.7.2 Beispiele für die Verwendung der Bausteindienste

Hier finden Sie einige Beispiele, die die Syntax von Variablennamen für Bausteindienste verdeutlichen.

#### Lesen eines Bausteins

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

**&blockread()**

Um beispielsweise den Baustein *OB1* in eine Datei "c:\temp\ob1.blk" zu lesen, muss das Item mit folgendem Wert beschrieben werden:

---

{0x0040/OB1/c:ltemplob1.blk}

---

#### Hinweis

Um den Wert wie in diesem Beispiel anzugeben, muss als Datentyp String angefordert werden. Der Wert kann dann in der konvertierten Darstellung für Arrays angegeben werden.

---

### Schreiben eines Bausteins

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

#### **&blockwrite()**

Um beispielsweise den Baustein in der Datei "c:\temp\ob1.blk" zum Automatisierungssystem zu schicken, muss das Item mit folgendem Wert beschrieben werden:

{0x1000/c:ltemplob1.blk}

---

#### Hinweis

Um den Wert wie in diesem Beispiel anzugeben, muss als Datentyp String angefordert werden. Der Wert kann dann in der konvertierten Darstellung für Arrays angegeben werden.

---

### Beispiel für das Einketten eines Bausteins

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

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

Um beispielsweise den Baustein *DB1* in den Programmablauf des Automatisierungsgeräts einzuketten, muss das Item mit folgendem Wert beschrieben werden:

*DB1*

### Beispiel für das Löschen eines Bausteins

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

#### **&blockdelete()**

Für das Löschen eines eingeketteten Bausteins *DB1* muss das Item mit folgendem Wert beschrieben werden:

{0x0040/DB1}

### Beispiel für Speicherkomprimierung des Automatisierungsgeräts

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

#### **&blockcompress()**

Für die Speicherkomprimierung muss das Item mit einem leeren String beschrieben werden.

## 2.6.8 Passwörter

Schreibende und lesende Zugriffe von Bausteindiensten auf die CPU können bei der Projektierung mit einem Passwort versehen werden. Das Passwort hat eine höhere Priorität als der Schlüsselschalter der CPU.

### Schutzstufen

Für S7-Automatisierungssysteme gibt es drei Schutzstufen. Sie werden mit Hilfe von STEP 7 aktiviert:

- Schutz durch die Stellung des Schlüsselschalters
- Schreibschutz
- Schreib- und Leseschutz

Wenn das richtige Passwort übermittelt wird, werden alle Schutzstufen für diese Verbindung aufgehoben. Wenn ein leerer String als Passwort übermittelt wird, wird die Schutzfunktion für die Verbindung wieder aktiv.

Die Schutzstufe von zweiseitig projektierten Verbindungen kann in der Projektierung vorbelegt werden. Wird der Schutz für zweiseitige Verbindungen nicht durch die Projektierung aktiviert, sind auf diesen Verbindungen alle Dienste selbst in Schlüsselschalterstellung *RUN* möglich.

Für einseitig projektierte Verbindungen besteht eine Abhängigkeit von der Schlüsselschalterstellung. Die folgende Tabelle zeigt die Abhängigkeit zwischen Passwortübertragung und Schutzstufe bei einseitig projektierten Verbindungen.

<b>Einseitig projektierte Verbindungen</b>	<b>Resultierende aktive Schutzstufe</b>				
Projektierte Schutzstufe	Passwort wurde <i>nicht</i> übertragen		Richtiges Passwort wurde übertragen		
	Schlüsselschalterstellung		Schlüsselschalterstellung		
	RUN	RUN-P, STOP	RUN	RUN-P, STOP	
Schlüsselschalter RUN - durch Passwort aufhebbar	Lesen	Alles	Alles	Alles	
Schlüsselschalter RUN - durch Passwort <i>nicht</i> aufhebbar	Lesen	Alles	Lesen	Alles	
Schreibschutz mit Passwort	Lesen	Lesen	Alles	Alles	
Schreib- / Leseschutz	Keine	Keine	Alles	Alles	

Legende:

Alles = alle Dienste ausführbar

Lesen = nur Baustein lesen

Keine = keine Bausteindienste erlaubt

### 2.6.8.1 Syntax der Steuervariablen für Passwörter

#### Syntax

`S7: [<Verbindungsname>] &password()`

#### Erklärungen

##### S7

S7-Protokoll für den Zugriff auf die Prozessvariable.

##### <Verbindungsname>

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

##### &password()

Kennzeichen für den Passwort-Dienst.

Die Übermittlung des Passworts wird durch das Schreiben eines Wertes ausgelöst.

Das Passwort kann in folgenden Darstellungen übermittelt werden:

- *Octetstring*  
String mit den hexadezimalen Codes der einzelnen Zeichen des Passworts, getrennt durch einen Punkt (maximal 8 Zeichen)
- *String*  
Beliebiger, alphanumerischer String, eingeleitet durch das Zeichen "&" (maximal 8 Zeichen)

Das Schreiben des Passworts liefert folgende Rückgabewerte:

- *OPC\_E\_BADRIGHTS*  
Das Passwort ist ungültig
- *S\_OK*  
Das Passwort ist gültig

OLE-Datentyp	Visual Basic Typ
String	String

### 2.6.8.2 Beispiel für die Verwendung der Passwörter

Hier finden Sie ein Beispiel, um die Übermittlung von Passwörtern zu verdeutlichen.

#### Beispiel für die Passwortübergabe

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

##### &password()

Um das Passwort *SetMeFre* an das Automatisierungsgerät zu übergeben, muss die Variable mit folgendem Wert beschrieben werden:

`&SetMeFre`

## 2.6.9 Server-Dienste

### Zugriff auf DB 1

Die PC-Station mit einem aktiven OPC-Server für das S7-Protokoll stellt einen S7-Datenbaustein zum Lesen und Schreiben zur Verfügung und wird damit zum S7-Server. Mit PUT- und GET-Funktionsbausteinen können S7-Stationen in ihrem Programm den Datenbaustein der PC-Station beschreiben oder lesen. Der Verbindungsauflaufbau ist beidseitig möglich.

Der Datenbaustein (DB 1) hat eine Größe von 65535 Byte. Symbolische Namen und eine Strukturierung des Datenbausteins nach Variablen bestehen nicht. Die Nummer des Datenbaustein kann von remote und auch lokal, z. B. von einem SIMATIC NET OPC-Server mit OPC Scout, abgefragt und beim Durchsuchen des Namensraums angezeigt werden.

Es wird nur ein Datenbaustein zur Verfügung gestellt. Keine Merker, Ein-, Ausgänge, Zähler oder Timer. Eine Projektierung oder das Laden des Bausteins ist nicht erforderlich.

Über eine lokale S7-Verbindung `@LOCALSERVER` kann ein OPC-Client (z. B. der OPC Scout) die Werte des Datenbausteins lesen oder beschreiben. Der OPC-Client kann zudem den Datenbaustein beobachten und sich über Datenänderung benachrichtigen lassen.

Wenn mehrere Clients gleichzeitig versuchen die selben Datenbereiche zu beschreiben, wird sichergestellt, dass jeder Auftrag nacheinander vollständig ausgeführt wird (Datenkonsistenz bei Parallelzugriff).

Nach Neuanlauf der PC-Station bleiben die Werte im Datenbaustein erhalten (Datenpermanenz).

---

#### Hinweis

Um die S7-Server-Dienste zu betreiben, muss ein lokaler OPC-Client auf der PC-Station mit S7-Items aktiv sein. Falls ein Verbindungsauflaufbau nach Bedarf in NCM projektiert ist, müssen die S7-Items die S7-Verbindung beinhalten, über welche der remote Partner Daten lesen oder schreiben will. Alternativ kann ein permanenter Verbindungsauflaufbau projektiert werden. Passive Verbindungen erfordern permanenten Verbindungsauflaufbau. Dann kann ein beliebiges S7-Item aktiviert werden, um den S7-OPC-Server zu starten.

---

---

#### Hinweis

Die S7-Server-Dienste sind erst mit OPC-Server-V6.3-Projektierungen der PC-Station freigegeben.

---

### 2.6.9.1 Beispiel für die Verwendung von Server-Diensten

Ein S7-Client könnte z. B. eine S7 300-Station sein, welche Statusdaten an die PC-Station melden möchte, ohne dass diese die Statusdaten ständig abpollt, siehe nachfolgende Abbildung. Er schreibt dann (selten) Statuswerte in den Datenbaustein.

Ein lokaler Client auf der PC-Station kann über Datenänderungen der Statuswerte informiert werden, siehe Abbildung "Beobachten durch häufiges zyklisches Lesen eines Werts...". Ein ständiges Pollen der Statuswerte auf der S7-Station wird dadurch vermieden.

Die folgende Abbildung zeigt, wie ein S7-Gerät Daten in einer PC Station mit S7-OPC-Server beschreibt.

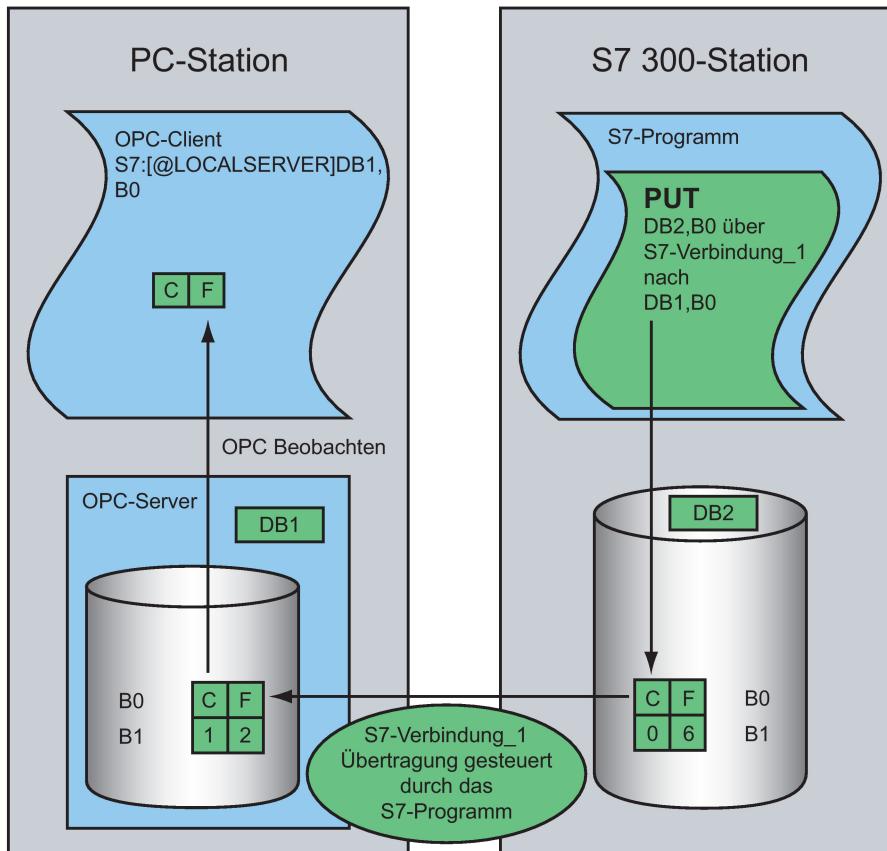


Bild 2-15 Server-Dienst: Seltenes Schreiben (PUT) eines Werts zur Entlastung der S7-Station

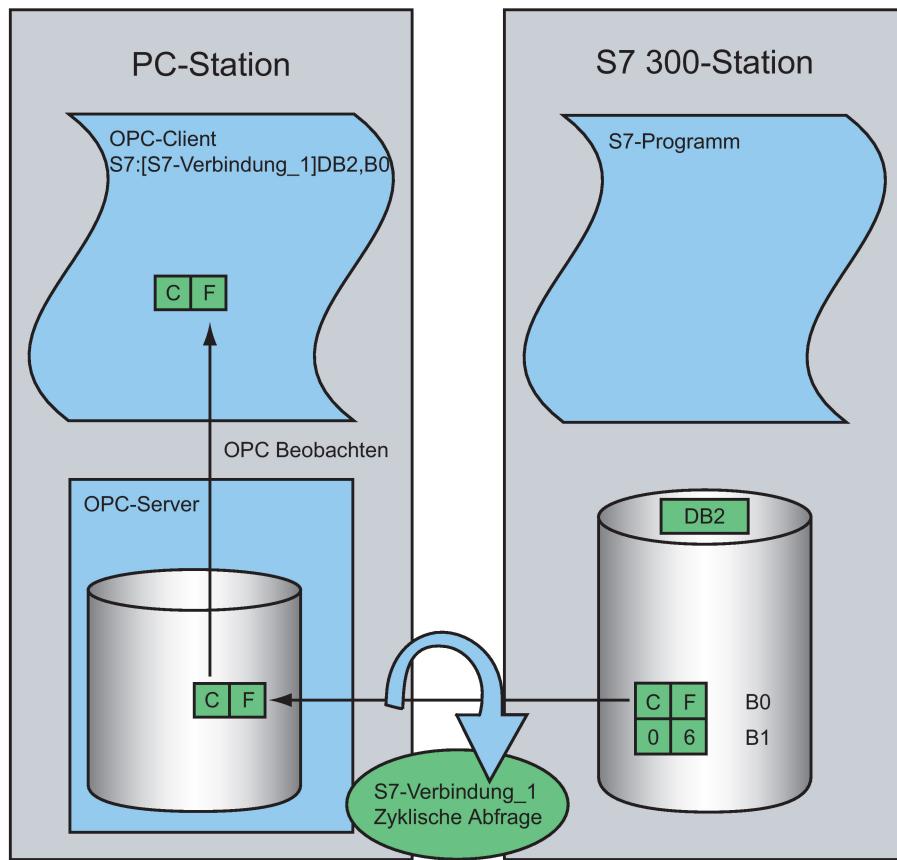


Bild 2-16 Beobachten durch häufiges zyklisches Lesen eines Werts (pollen); Belastung der S7-Station

## 2.6.10 S7 Template-Datenvariablen

Sie haben mit den Prozessvariablen für das S7-Protokoll flexible Einstellmöglichkeiten, um die Prozessdaten Ihrer Anlage in den von ihm gewünschten Datenformaten zu erhalten.

Die Vielfalt der Adressierungsmöglichkeiten lässt sich allerdings nicht in einen vollständig durchsuchbaren Namensraum fassen. Bereits ein Datenbaustein mit der Länge eines einzelnen Bytes besitzt etwa 40 verschiedene Datenformatoptionen – angefangen vom Byte, Char, Felder mit einem Element davon, einzelne Bits, Felder von Bits mit bis zu 8 Feldelementen an unterschiedlichen Bitoffsets beginnend.

Der OPC-Server unterstützt den Anwender deshalb mit den sogenannten Template-Datenvariablen im S7-Namensraum. In einem für einen OPC-Client typischen Texteingabefeld können diese Templates durch Ändern einiger weniger Zeichen in gültige ItemIDs verwandelt werden.

Beispiel:

`S7: [S7-Verbindung_1] DB<db>, DWORD<o>` //Template für ein DWORD eines Datenbausteins

Durch Ersetzen von `<db>` mit der Bausteinadresse und `<o>` dem Offset innerhalb des Datenbausteins erhält der Anwender eine gültige Itemsyntax.

⇒ S7: [S7-Verbindung\_1] DB100, d3

Weiteres Beispiel:

S7: [S7-Verbindung\_1] bsend<len>,<rid>,x<o>. <bit>,<c> //Template für ein Bitarray,  
Blockdienst

⇒ S7: [S7-Verbindung\_1] bsend100,21,x0.0,24

---

#### **Hinweis**

Die Verwendbarkeit von S7-OPC-UA-Template-Datenvariablen kann im Konfigurationsprogramm "Kommunikations-Einstellungen" unter "OPC-Protokollauswahl" > Klicken des Pfeilsymbols bei "S7" aktiviert und deaktiviert werden.

---

### 2.6.10.1 Template-Datenvariablen im Namensraum

Die Template-Datenvariablen befinden sich in eigenen Ordnern im Namensraum.

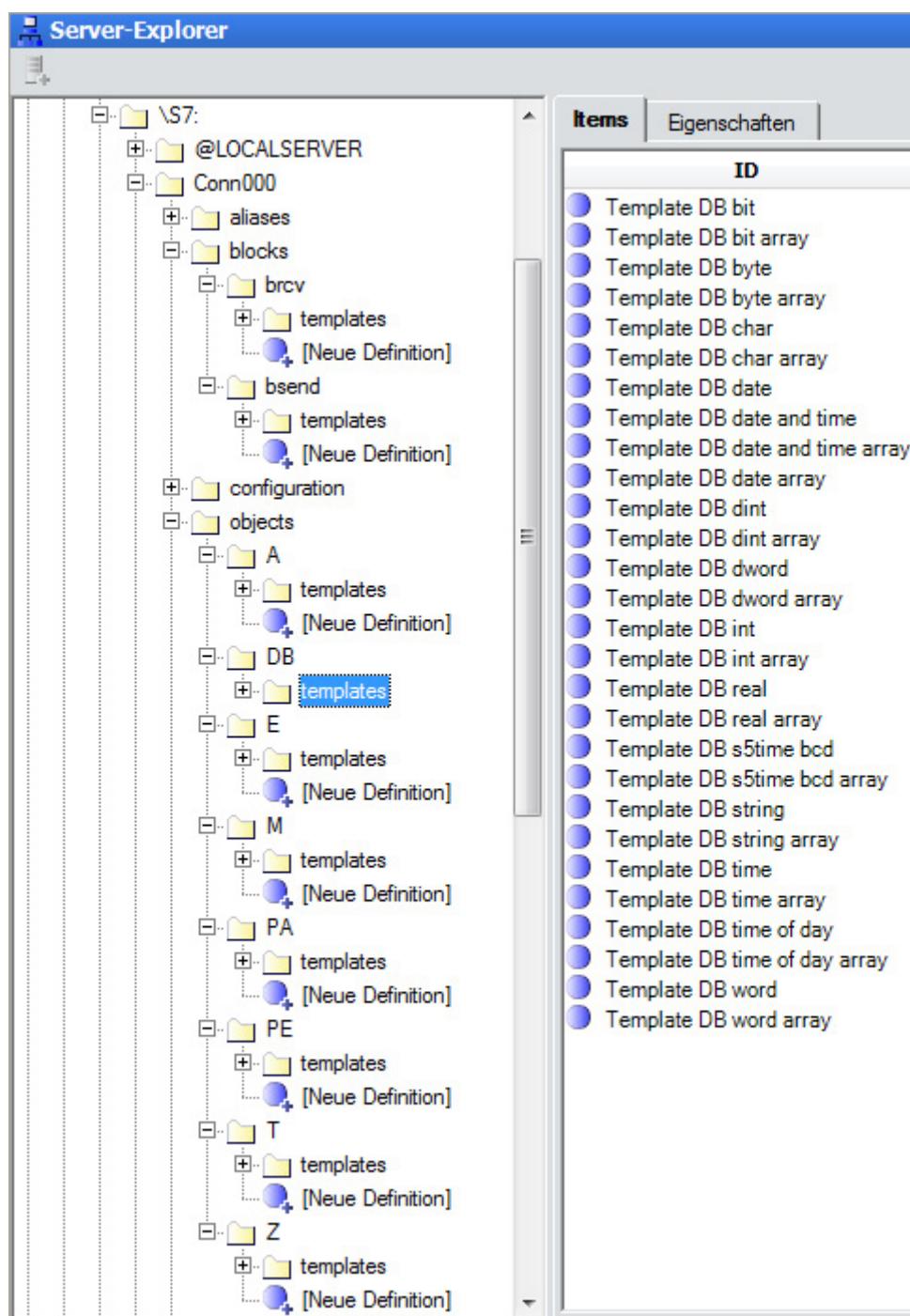


Bild 2-17 Hierarchie der Template-Datenvariablen

Der Blattname im Namensraum ist dabei eine weitreichende Hilfestellung ("Template M Byte").

### 2.6.10.2 Syntax der Template-Datenvariablen

Es gibt folgende Templates:

#### Für Merker, Ein- und Ausgänge

```
S7: [<Verbindungsname>] <Object><Typ><o>, <c>

<Object>:= "E" | // Eingang
           "A" | // Ausgang
           "M" | // Merker
           "PE" | // Peripherie-Eingang
           "PA" // Peripherie-Ausgang

<Typ><o>:= "X<o>.<bit>" | // Bit <Bit>= Template für die Bit-Adresse
           "B<o>" | // Byte (unsigned)
           "W<o>" | // Wort (unsigned)
           "D<o>" | // Doppelwort (unsigned)
           "CHAR<o>" | // Byte (signed)
           "INT<o>" | // Wort (signed)
           "DWORD<o>" | // Doppelwort (signed)
           "REAL<o>" | // Fließkomma 4 Byte
           "DT<o>" | // Datum und Uhrzeit, 8 Byte BCD-Format
           "DATE<o>" | // Datum und Uhrzeit, 8 Byte, Uhrzeit immer 00:00:00
           "TIME<o>" | // Zeitwert (signed), IEC-Format, in ms
           "S5TIMEBCD<o>" | // Zeit-Variable (unsigned, 16 Bit), 0 ... 9990000 ms
           "TOD<o>" | // Tageszeit (unsigned), 0 ... 86399999 ms
           "STRING<o>.<len>" // Zeichenfolge. <len> ist ein Template für die Stringlänge
```

<o> // Template für die Adresse der ersten Variablen, dies ist ein Byte-Offset im Adressbereich.

<c> // Template für die Anzahl der Variablen eines Typs, die ab dem im Parameter Adresse angegebenen Offset angesprochen werden sollen.

#### Für Datenbausteine

```
S7: [<Verbindungsname>] DB<db>, <Typ><o>, <c>
<db> // Template für die Nummer des Datenbausteins
```

#### Blockdienste

```
S7: [<Verbindungsname>] bsend<len>, <rid>, <Typ><o>, <c>
```

S7: [<Verbindungsname>] brcv, <rid>, <Typ><o>, <c>  
 <len> // Template für die Länge des Datenblocks in Bytes, der gesendet werden soll.  
 <rid> // Template für die ID des Adressierungsparameters. Sie ist für ein Bausteinpaar  
 (BSEND/ BRCV) festgelegt und innerhalb einer Verbindung eindeutig definiert.

## S7-Timer

S7: [<Verbindungsname>] TDA<i> // S7-Timer-Variable (Typ TDA):  
 S7: [<Verbindungsname>] TBCD<i>{, <c>} // S7-Timer-Variable (Typ BCD):

## Zähler

S7: [<Verbindungsname>] Z<i>{, <c>} // Zähler, Die Adressangabe <i> ist eine Zählernummer.

## 2.6.11 Unprojektierte S7-Verbindung

### Zugriff auf ein Partnergerät ohne Projektierung

Üblicherweise werden Verbindungen zu Partnergeräten in einer Projektierung definiert. Dafür stehen die Programme STEP 7 bzw. NCM PC zur Verfügung.

Allerdings gibt es Anwendungsfälle, bei denen z. B. Daten ohne Verbindungsprojektierung von einem Partnergerät gelesen bzw. Variablen geschrieben oder beobachtet werden sollen. Es besteht die Möglichkeit, diese Aufgaben auch ohne Projektierung durchzuführen.

### Voraussetzungen

Für einen Gerätezugriff ohne Projektierung müssen alle kommunikationsrelevanten Daten des Partnergeräts bekannt sein. Dazu gehören unter anderem der Verbindungsname, der Zugangspunkt und die Stationsadresse. Die notwendigen Parameter sind im folgenden Abschnitt aufgeführt. Achten Sie bei der Wahl des Verbindungsnamens auf Eindeutigkeit gegenüber bereits projektierten Namen.

### Dienstzugangspunkt (SAP Service Access Point)

Der Dienstzugangspunkt ist jeweils der Punkt, an dem eine Schicht des ISO/OSI-Referenzmodells der direkt übergeordneten Schicht ihre Dienste zur Verfügung stellt.

Der gesamte Informationsaustausch zwischen zwei benachbarten Schichten erfolgt über die Dienstzugangspunkte. Er ist die Schnittstelle zwischen der untergeordneten und der übergeordneten Schicht. Im Falle des Übergangs zwischen Schicht 3 (Network Layer) und Schicht 4 (Transport Layer) heißt der Dienstzugangspunkt NSAP, Network Service Access Point, im Falle des Übergangs von Schicht 4 auf Schicht 5 (Session Layer) heißt er TSAP, Transport Service Access Point.

Einem Dienstzugangspunkt sind üblicherweise bestimmte Ressourcen und Kommunikationspartner zugeordnet. Deshalb werden die innerhalb einer Kommunikation benötigten SAPs mit eindeutigen Namen oder Nummern gekennzeichnet.

## Parameter für eine unprojektierte Verbindung

Eine unprojektierte Verbindung hat folgende Syntax:

S7:[<Verbindungsname>]<VFD><Zugangspunkt><Adress-Spezifikation>]<Datenelement>

Dabei haben die einzelnen Bestandteile folgende Bedeutung:

### <Verbindungsname>

Der Verbindungsname darf noch nicht vorhanden sein. Falls der gewählte Verbindungsname bereits projektiert wurde oder für eine andere frei spezifizierte S7-Verbindung verwendet wurde, werden die Zusatzinformationen für frei spezifizierte Verbindungen ignoriert und das Item wird der bereits vorhandenen Verbindung zugeordnet.

### <VFD>

Ein VFD-Name entspricht dem Applikationsnamen in der PC-Station-Projektierung von NetPro.

Ein beliebiger VFD-Name. Alle Verbindungen können auf demselben VFD angelegt werden. Der VFD-Name darf maximal 32 Zeichen lang sein. Der gewählte Name darf noch nicht vorhanden sein.

### <Zugangspunkt>

Der Zugangspunkt der Kommunikationsbaugruppe muss mit dem Programm "Kommunikations-Einstellungen" vorab konfiguriert werden.

### <Adress-Spezifikation> = <Local TSAP>, <Stationsadresse>, <Remote TSAP>, <Mode>

Die Adress-Spezifikation enthält folgende Informationen, wobei die einzelnen Werte durch Kommas getrennt werden:

- **Local TSAP (Local Transport Service Access Point, lokaler Dienstzugangspunkt)**

Beim S7-Protokoll besteht der Local TSAP aus genau zwei durch Leerzeichen oder Punkt getrennten Zahlen, die folgende Bedeutung haben:

- Das erste Byte kann eine Gerätekennung enthalten, erlaubte Werte sind *02* oder *03*.  
*02* OS (Operating Station Bedienen und Beobachten)  
*03* Sonstiges
- Das zweite Byte ist immer 0.

Empfohlene Einstellung: 02.00

- **Stationsadresse**

Für die Stationsadresse gibt es drei Darstellungsvarianten:

Übertragungsverfahren	Darstellung der Stationsadresse	Beispiel
PROFIBUS	PROFIBUS-Adresse, dezimale Darstellung	65
TCP/IP	TCP/IP-Adresse	192.168.0.7
ISO	MAC-Adresse	08-06-05-e4-3a-00

**Remote TSAP (Remote Transport Service Access Point, entfernter Dienstzugangspunkt)**

Die Darstellung ist die gleiche wie beim Local TSAP, allerdings hat das zweite Bytes eine andere Bedeutung:

- erstes Byte: enthält eine Gerätekennung, erlaubte Werte sind *02* oder *03*:  
*02* OS (Operating Station Bedienen und Beobachten)  
*03* Sonstiges  
 Empfohlene Einstellung: 02
- zweites Byte: enthält die Adressierung der SIMATIC S7-CPU,  
 unterteilt in:  
*Bit 7 ... 5* Rack (Subsystem) der S7-CPU  
*Bit 4 ... 0* Steckplatz der S7-CPU

#### Hinweis

Bei der Parametrierung des "Local TSAP" und des "Remote TSAP" wird empfohlen, für das erste Byte jeweils die gleiche Einstellung zu wählen.

#### Mode

S7-Verbindungen werden entweder vom OPC-Server oder vom Verbindungspartner aktiv aufgebaut. Für solche Verbindungen kann außerdem ein optimierter Schreib- bzw. Lesezugriff eingesetzt werden. Es gibt folgende Werte für Mode:

- *1*: Aktiver Verbindungsaufbau des OPC-Servers mit Optimierung
- *3*: Aktiver Verbindungsaufbau des OPC-Servers ohne Optimierung

#### <Datenelement>

Hier wird beispielsweise ein Datenbaustein mit Nummer sowie der Typ (Byte, Wort usw.) und die Adresse (z. B. Byte-Offset) angegeben. Welche Datenelemente für S7 möglich sind, ist im Kapitel "S7-Kommunikation (Seite 117)" beschrieben.

#### Hinweis

Weitere Verbindungsparameter:

Die PDU-Größe kann nicht vorgegeben werden. Es wird die maximale Größe von 960 Byte beim Verbindungsaufbau vorgeschlagen und damit die maximale mögliche Größe des Partnergeräts ausgehandelt.

Für die Verbindungsaufbau-Timeouts und Auftrags-Timeouts werden feste Werte (jeweils 15.000ms) verwendet.

Der Vorschlagswert für die maximale Anzahl paralleler Netzaufträge beträgt 2.

## Beispiele

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

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

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

### Zugangspunkt definieren

1. Öffnen Sie das Konfigurationsprogramm "Kommunikations-Einstellungen", um einen Zugangspunkt zu definieren und einer Schnittstellenparametrierung zuzuordnen.

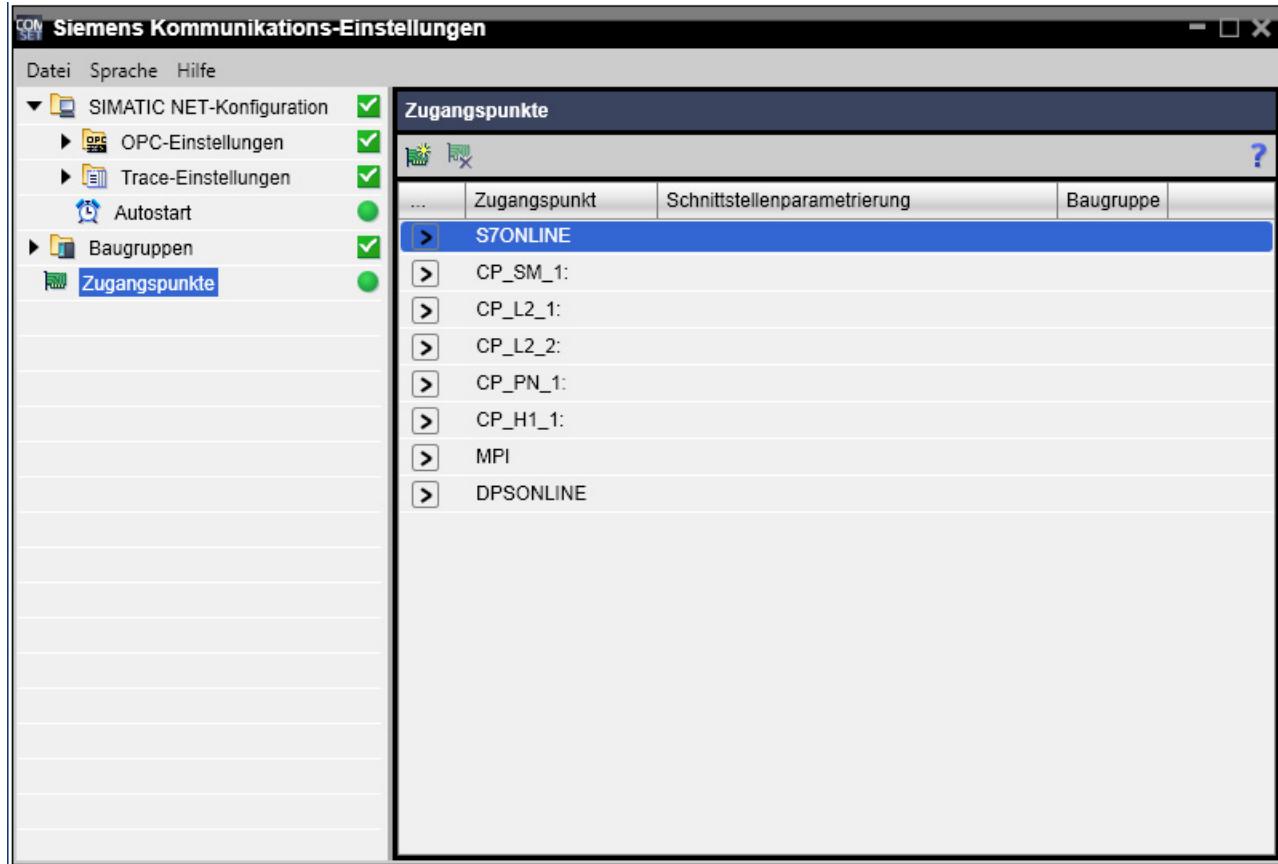


Bild 2-18 Öffnen des Konfigurationsprogramms "Kommunikations-Einstellungen"

2. Klicken Sie bei "S7Online" auf das Pfeilsymbol.
3. Wählen Sie über die Klappliste "Zugeordnete Schnittstellenparametrierung" die Schnittstellenparametrierung für den Zugangspunkt aus.

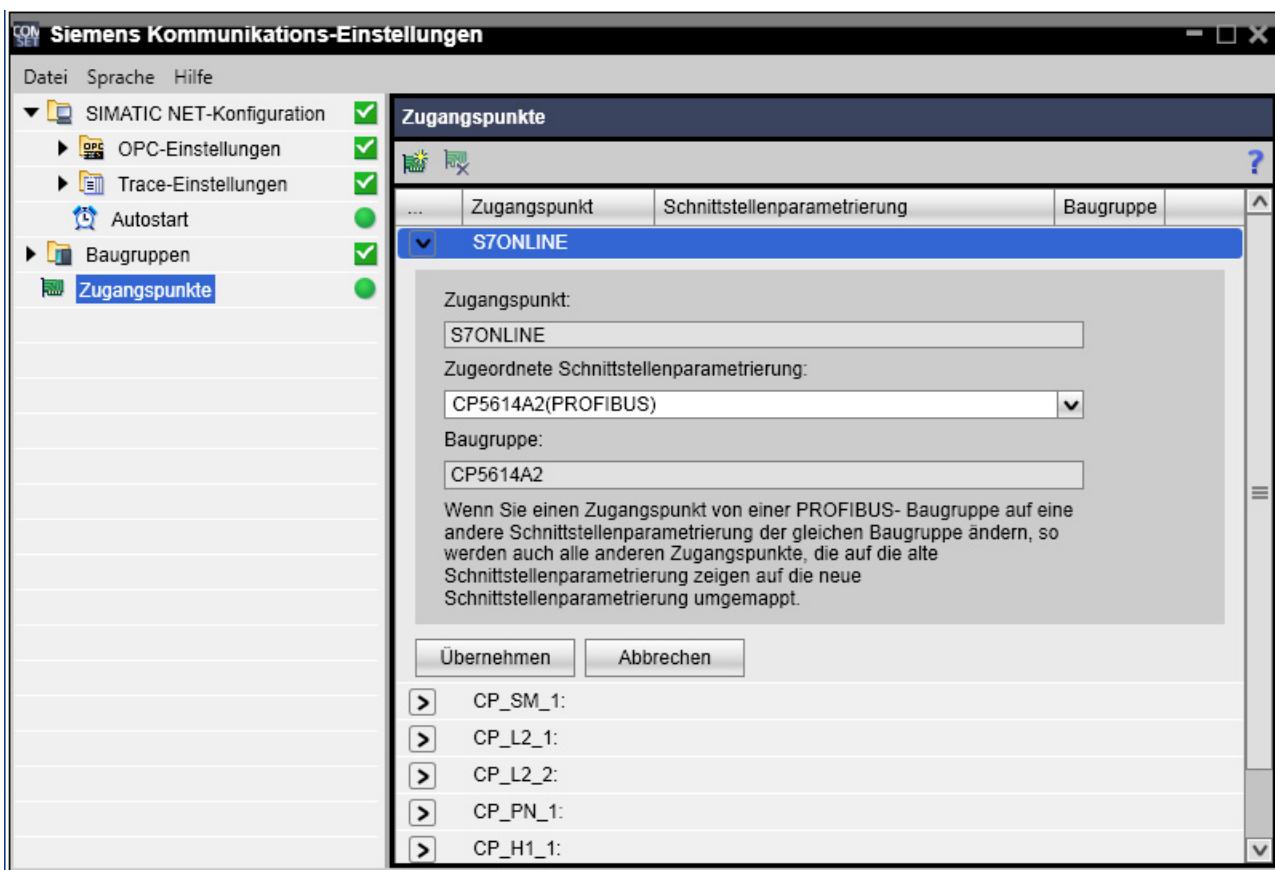


Bild 2-19 Schnittstellenparametrierung zuordnen

### Item hinzufügen

Rufen Sie ein Clientprogramm auf und legen Sie ein Item gemäß obiger Syntax an. Beim Programm OPC Scout V10 ziehen Sie dazu ein beliebiges Item aus dem Ordner "S7" in die DA-Ansicht und klicken anschließend am rechten Rand der Tabellenspalte "ID" auf die Schaltfläche "...". Geben Sie nun die Syntax ein und bestätigen Sie mit "Ok".

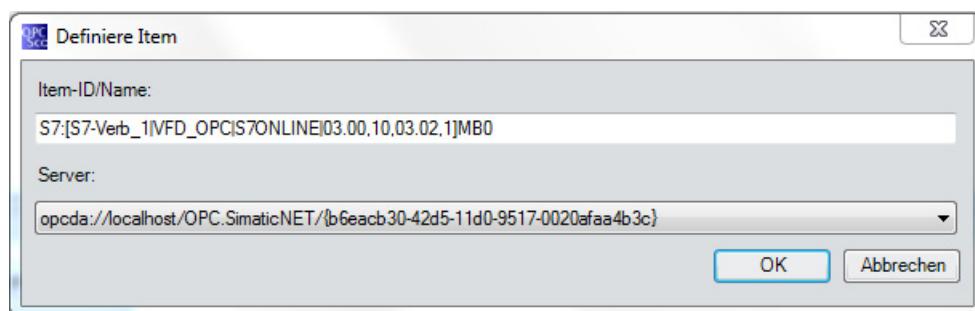


Bild 2-20 OPC Scout V10 - Item einfügen

Nach dem Hinzufügen des Items und solange das Item aktiv ist, kann die Verbindung wie eine projektierte Verbindung verwendet werden. Sie können also im Namensraum browsen und auch weitere Items hinzufügen, ohne die Syntax der unprojektierten Verbindung zu

verwenden. Es genügt die Angabe des Verbindungsnamens, zum Beispiel S7:[S7-Verb\_1]MB1.

## 2.6.12 COML S7-Verbindung

### Zugriff auf Partnergeräte mit lokalen COML S7-Verbindungen

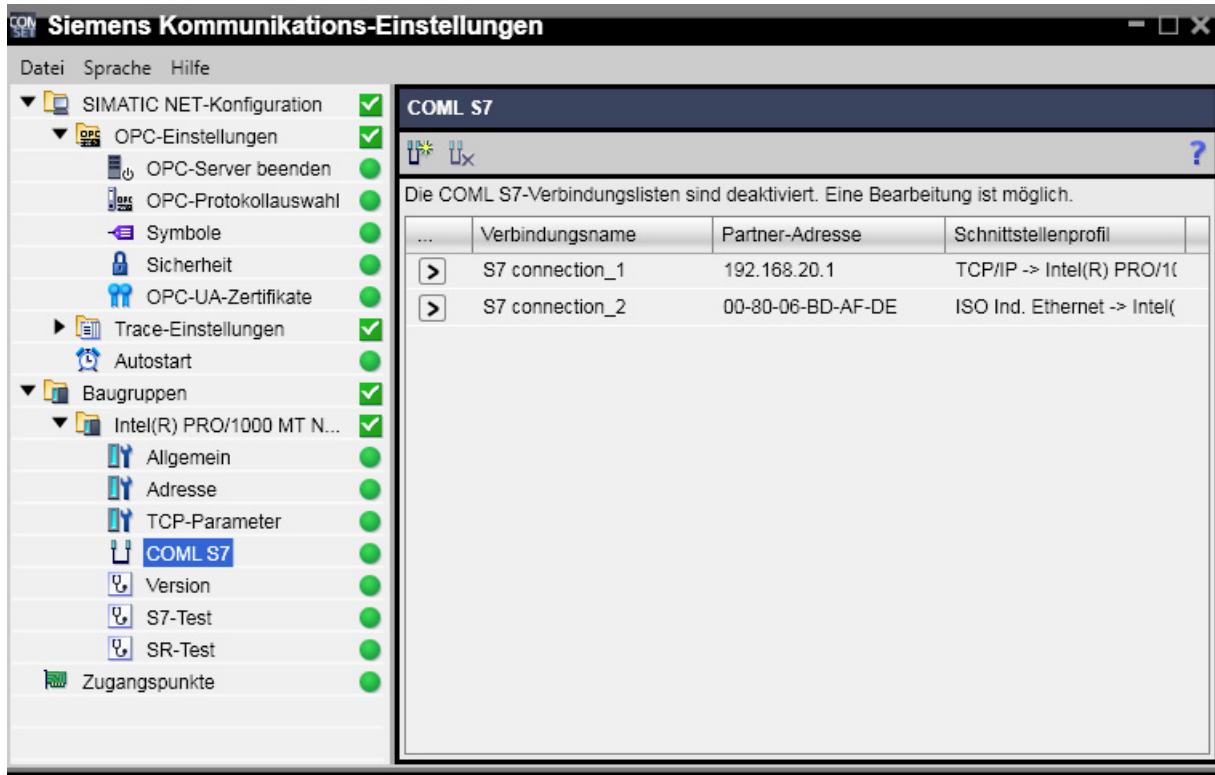


Bild 2-21 COML S7 - Configuration Management lokal - Projektiersoftware für S7-Verbindungen

Mit Hilfe der COML S7-Projektiersoftware können S7-Verbindungen zu einer S7-CPU oder zu einem PC erstellt werden. Unter jeder Baugruppe im Konfigurationsprogramm "Kommunikations-Einstellungen" befindet sich die Lasche "COML S7". Wenn Sie auf diese Lasche klicken, erscheint auf der rechten Seite des Dialog-Fensters die COML S7-Verbindungsliste, über die Sie gezielt für die ausgewählte Baugruppe S7-Verbindungen zu den Partnergeräten anlegen können. Die bei der Projektierung entstandenen Daten werden für alle Baugruppen in einer Datenbasis zusammengefasst und lokal auf dem PC gespeichert. Sie kann zur Datensicherung exportiert und importiert werden. Danach muss die COML-S7-Projektierung im Kontextmenü der Ebene "Baugruppen" aktiviert werden, um die angelegten S7-Verbindungen in den COML S7-Verbindungslisten für eine S7-Kommunikation verwenden zu können.

## Voraussetzungen zur Nutzung der COML S7-Verbindungen

Für einen Gerätezugriff mit COML-S7-Projektierung müssen alle kommunikationsrelevanten Daten des Partnergeräts bekannt sein. Dazu gehören unter anderem der Partner TSAP und die Partner-Adresse. Weitere Informationen über die Parameter finden Sie in der Online-Hilfe beim Anlegen der COML S7-Verbindungen.

---

### Hinweis

Der gleichzeitige Betrieb von projektierten S7-Verbindungen von STEP 7/SIMATIC NCM PC und COML S7 ist nicht möglich. Sie sind gegenseitig verriegelt.

Nach der Aktivierung von COML S7-Verbindungen wird somit das Laden von S7-Verbindungen von STEP 7/SIMATIC NCM PC abgelehnt.

---

### Hinweis

Der gleichzeitige Betrieb von projektierten Verbindungen mit unprojektierten Verbindungen ist nicht möglich. Sie sind gegenseitig verriegelt.

Nach der Aktivierung von projektierten Verbindungen wird der Betrieb von unprojektierten Verbindungen abgelehnt.

---

## 2.7 S7-Kommunikation mit OPC UA

### 2.7.1 Eigenschaften der S7-Kommunikation mit OPC UA

Der SIMATIC NET OPC-Server ermöglicht die Nutzung der S7-Kommunikation über OPC UA.

Der S7-OPC-UA-Server von SIMATIC NET hat folgende Eigenschaften:

- Variabldienste
  - Zugriff und Beobachtung von S7-Variablen
- Blockorientierte Dienste
  - Programmgesteuerte Übertragung größerer Datenblöcke
- Server-Funktionalität
  - Der PC kann als Server für Datenblöcke und Datenbausteine eingesetzt werden.
- Bausteindienste
  - Übertragung eines ladbaren Datenbereichs von und zu S7

- S7-Passwortfunktionen
  - Setzen eines Passworts zum Zugriff auf geschützte Bausteine
- Events, Conditions und Alarme
  - Verarbeitung von S7-Meldungen und S7-Diagnoseereignissen

## **2.7.2 SIMATIC NET OPC-UA-Server für das S7-Protokoll**

### **Einleitung**

Der folgende Abschnitt beschreibt eine Konfigurationsvariante für das S7-Protokoll, die auch OPC UA unterstützt. Hierfür werden die S7-COM-OPC-Data-Access-Server als Outproc-OPC-Server eingerichtet.

## Konfiguration

Die Aktivierung des S7-UA-Servers erfolgt durch die Auswahl von "S7" und "OPC UA" im Konfigurationsprogramm "Kommunikations-Einstellungen" im Katalog "OPC-Protokollauswahl":

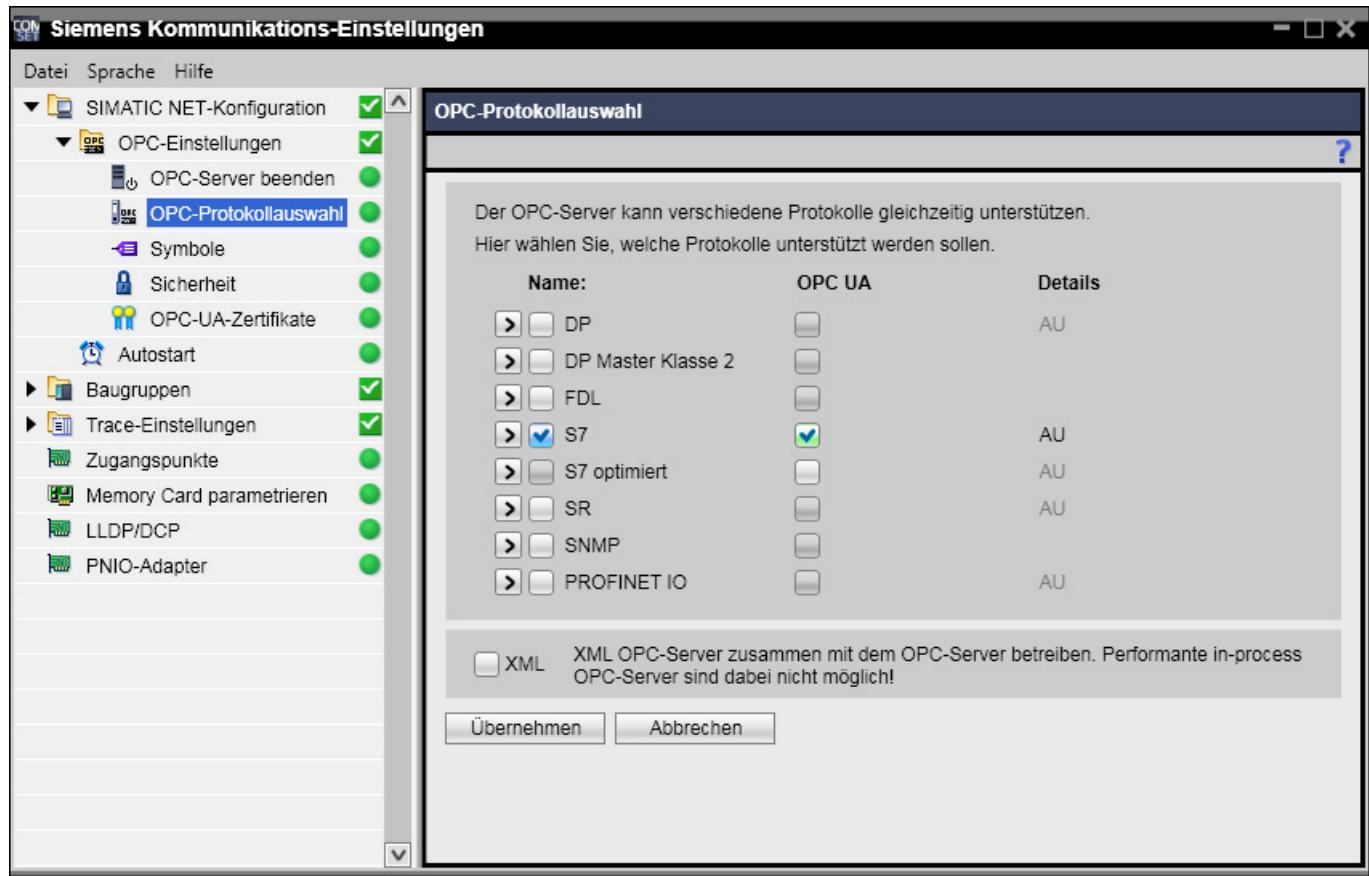


Bild 2-22 Fenster des Konfigurationsprogramms "Kommunikations-Einstellungen" zur Auswahl von OPC UA für das S7-Protokoll

Zusätzlich dürfen Sie "Symbolik" auswählen.

### Hinweis

Für die Erstellung von Symbolen mit STEP 7 Professional (TIA Portal) oder dem Symbol-Editor ist die Verwendung folgender Zeichen erlaubt: A-Z, a-z, 0-9, \_, -, ^, !, #, \$, %, &, ', /, (, ), <, >, =, ?, ~, +, \*, '!', ., |, @, [ , ], { , }, ". Zusätzlich sollten Sie bei der Erstellung von Symbolen mit STEP 7 darauf achten, dass es bei der Array-Auflösung zu Problemen kommen kann, wenn Ihre Symboldatei gleichzeitig Symbole der Form <Symbolname> und <Symbolname>[<Index>] enthält.

## **Vorteile / Nachteile**

Bei Verwendung des S7-OPC-UA-Servers ist nur der Outproc-Betrieb von S7 möglich. Der S7-OPC-UA-Server-Prozess muss zum Erhalt der Empfangsbereitsschaft in Betrieb sein. Ein Beenden des S7-OPC-UA-Servers - auch nach Abmeldung aller OPC-UA-Clients - wird nicht ausgeführt und darf nicht ausgeführt werden.

Dem stehen folgende Vorteile gegenüber:

- Es ist keine COM/DCOM-Konfiguration mehr nötig.
- Performante, sichere Kommunikation
- Für Ereignisse und Datenzugriffe ist nur noch ein Server erforderlich.
- Nachteil: Ein gleichzeitiger Betrieb des performanten Inproc OPC-DA-S7-Servers ist nicht möglich.

## **Hinweise**

### **Hinweis**

Wenn OPC UA für das S7-Protokoll aktiv ist, werden permanent projektierte S7-Verbindungen aufgebaut, sobald der erste OPC-UA-Client am S7-OPC-UA-Server angemeldet ist. Wenn sich kein OPC-UA-Client verbunden hat, z.B. nach Hochfahren des Rechners, werden noch keine permanenten S7-Verbindungen aufgebaut.

### **Hinweis**

S7-Verbindungen können zum Aufbau "bei Bedarf" in SIMATIC STEP 7/NCM PC projektiert werden.

Die Kommunikation der OPC-Funktionen wird dann erst bei Bedarf aufgebaut; somit kann der Verbindungsaufbau länger dauern und OPC in der Anlaufphase Fehler melden.

## 2.7.3 Wie wird der S7-OPC-UA-Server adressiert?

### Server-URL

Für das native binäre TCP-Protokoll gibt es für den OPC-Client zwei Möglichkeiten der Server-Adressierung:

- Direkte Adressierung:
  - opc.tcp://<hostname>:55101
  - oder
  - opc.tcp://<IP-Adresse>:55101
  - oder
  - opc.tcp://localhost:55101

Der S7-OPC-UA-Server hat den Port 55101.

Der redundante S7-OPC-UA-Server wird von remote über die gemeinsame redundante IP-Adresse angesprochen.

- Die URL des S7-OPC-UA-Server kann auch über den OPC-UA-Discovery-Dienst gefunden werden.

Die Eingabe zum Auffinden des Discovery-Servers lautet:

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

Der Discovery-Server hat den Port 4840 (für TCP-Verbindungen) und den Port 52601 (für HTTP-Verbindungen).

### IPv6-Adresse

Für die IP-Adresse kann auch eine IPv6-Adresse verwendet werden. Die Adresse muss in Klammern angegeben werden, z.B. [fe80:e499:b710:5975:73d8:14]

### Endpunkte und Sicherheitsmodi

Der SIMATIC NET S7-OPC-UA-Server unterstützt Endpunkte mit dem nativen binären TCP-Protokoll und erfordert Authentisierung über Zertifikate und eine verschlüsselte Übertragung.

Der Discovery-Dienst auf dem angesprochenen Host meldet die Endpunkte der Server, sowie deren Sicherheitsanforderungen und Protokollunterstützung.

Die Server-URL "opc.tcp://<hostname>:55101" des S7-OPC-UA-Server bietet folgende Endpunkte:

- Endpunkt im Sicherheitsmodus "SignAndEncrypt":

Zur Kommunikation mit dem Server werden Signierung und Verschlüsselung gefordert.

Die Kommunikation ist durch Zertifikataustausch und Passworteingabe geschützt.

Zusätzlich zum Sicherheitsmodus wird die Sicherheitsrichtlinie Basic128Rsa15 angezeigt.

- Endpunkt im Sicherheitsmodus "keiner":

In diesem Modus werden keine Sicherheitsfunktionen vom Server gefordert (Sicherheitsrichtlinie "None").

Weitere Details zu den Sicherheitsfunktionen finden Sie im Kapitel "OPC-UA-Schnittstelle programmieren (Seite 508)".

Die Sicherheitsrichtlinien "Basic128Rsa18" und "None" finden Sie in der UA-Spezifikation der OPC Foundation unter folgender Internet-Adresse:

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

Weitere Informationen finden Sie auf folgender Internetseite:

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

## Die OPC-UA-Discovery des OPC Scout V10

Der OPC Scout V10 ermöglicht das Öffnen des OPC-UA-Discovery-Dialogs zur Übernahme von UA-Endpunkten in den Navigationsbereich des OPC Scout V10.

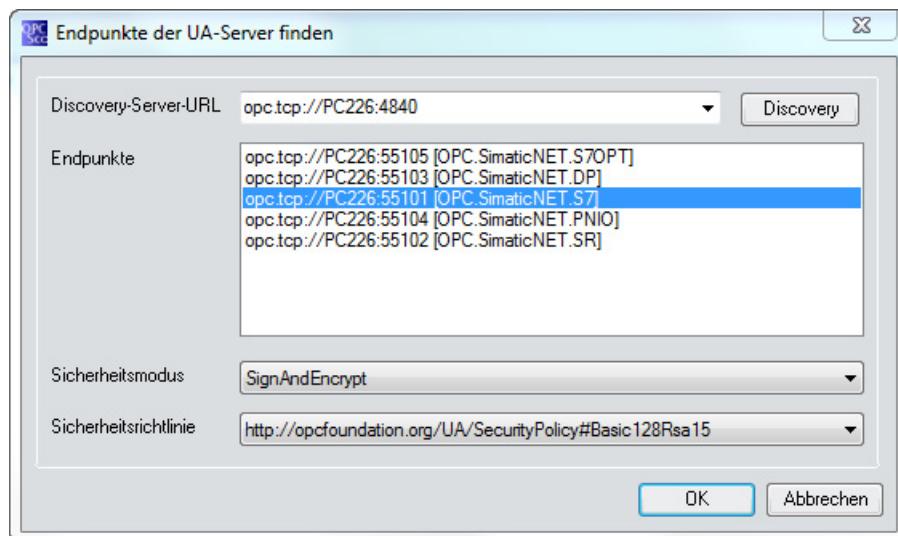


Bild 2-23 Das Dialogfeld "Endpunkte der UA-Server finden" des OPC Scout V10

Der S7-OPC-UA-Server kann über den OPC-UA-Discovery-Dienst gefunden werden. Zur Eingabe siehe oben unter "Server-URL".

Der OPC Scout V10 beinhaltet eine Liste der OPC-UA-Endpunkte. Der Discovery-Dienst auf dem angesprochenen Host meldet dann die registrierten OPC-UA-Server sowie deren Ports und Sicherheitsmodi.

Weitere Details finden Sie in der Online-Hilfe des OPC Scout V10.

## 2.7.4 Welche Namensräume bietet der S7-OPC-UA-Server an?

**Der S7-OPC-UA-Server bietet folgende Namensräume an:**

Tabelle 2- 2 Namensräume von OPC UA

Namensraum-Index	"Bezeichner" (Namensraum-URI) / Kommentar
0	"http://opcfoundation.org/UA/" von der OPC Foundation spezifiziert
1	"urn:Siemens:Automation.SimaticNET.S7:(GUID)" Eindeutiger Bezeichner des lokalen performanten S7-OPC-UA-Server.
2	"S7TYPES:" Definitionen für S7-spezifische Objekttypen.
3	"S7:" Bezeichner des lokalen performanten S7-OPC-UA-Server mit neuer vereinfachter Syntax (durchsuchbar und verwendbar mit UA)
4	"S7COM:" Bezeichner des Servers mit alter Syntax, S7-OPC-DA-kompatibel (mit UA verwendbar aber nicht durchsuchbar)
5	"S7SOURCES:" Bezeichner für Quellen von Alarmen und Anwenderdiagnoseereignissen.
6	"S7AREAS:" Bezeichner für Bereiche einer Alarmhierarchie.
7	"SYM:" Optionaler Server mit ATI-S7-Symbolik; abhängig von der Projektierung und der Konfiguration der PC-Station (durchsuchbar und verwendbar mit UA). Alternativ kann hier ein Präfix stehen, der in der Symbolik-Parametrierung ("Kommunikations-Einstellungen") festgelegt wird.

Die Namensraum-Indizes 0 und 1 sind reserviert und in ihrer Bedeutung von der OPC Foundation spezifiziert.

Die Zuordnung der restlichen Namensraum-Indizes zu den Bezeichnern (Namensraum-URI) muss zu Beginn einer OPC-UA-Session vom Client unter Angabe des Bezeichners über die Datenvariable "NamespaceArray" ermittelt werden. Die Bezeichner "S7TYPES:", "S7:", "S7COM:", "S7AREAS:" und "S7SOURCES:" sind beim S7-OPC-UA-Server immer vorhanden.

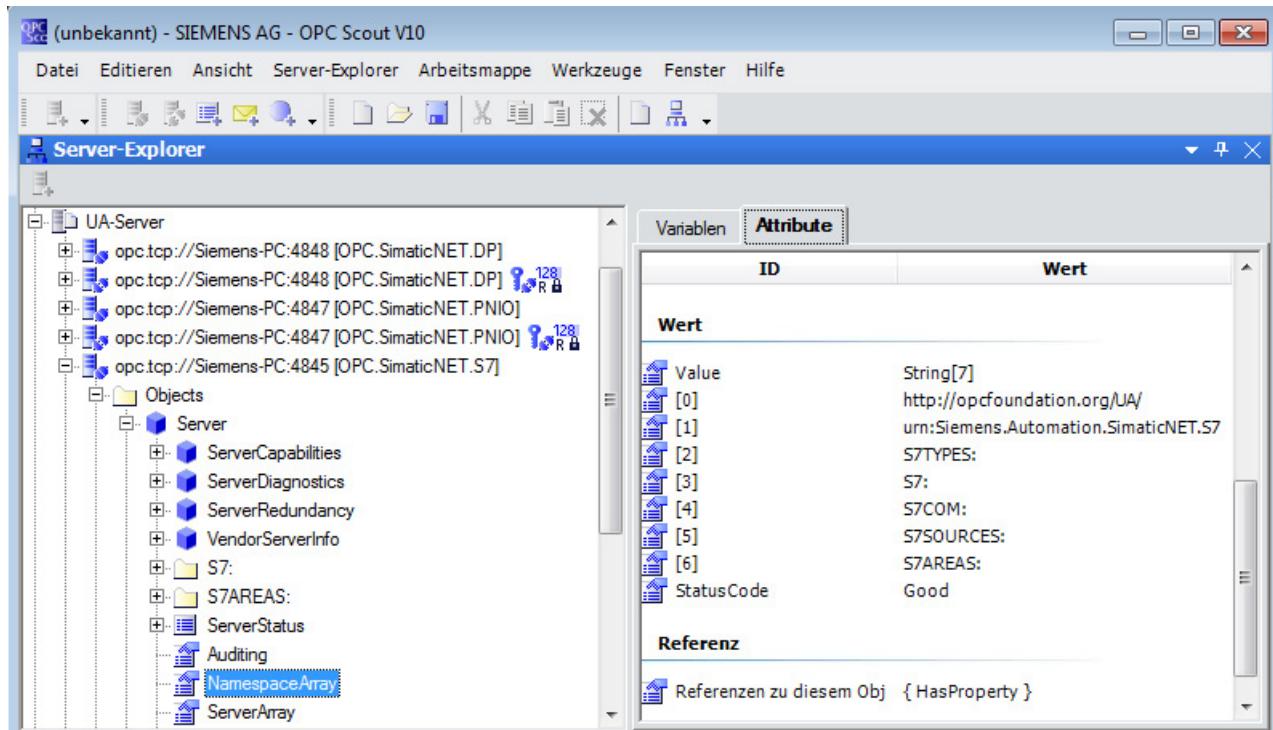


Bild 2-24 Anzeige der S7-OPC-UA-Namensräume mittels der Browse-Funktion des OPC Scout V10

## 2.7.5 Die Nodeld

### Identifikation einer S7-Prozessvariable

Die Nodeld identifiziert mit Hilfe des folgenden Tupels eine S7-Prozessvariable eindeutig:

- Namensraum-Index
- Bezeichner (Zeichenfolge, numerischer Wert)

## Beispiele

- Nodeld:
  - Namensraum-URI:  
*S7:*  
(= Namensraum-Index 3) für Siemens.Automation.SimaticNET.S7
  - Bezeichner:  
*S7-Verbindung\_1.m.10,b*
  
- Nodeld:
  - Namensraum-URI:  
*S7COM:*  
(= Namensraum-Index 4) für OPC.SimaticNET; die Syntax ist S7-OPC-DA-kompatibel
  - Bezeichner:  
*S7:[S7-Verbindung\_1]mb10*

## Wie verhält sich der neue auf OPC UA angepasste Namensraum?

Die Welt der OPC-Data-Access-Items eines COM-Servers ist zum Lesen und Schreiben von Prozessvariablen in sich abgeschlossen. Daneben existiert unabhängig davon die Alarmwelt.

Dagegen ist die OPC-UA-Sicht auf Automatisierungsobjekte auch auf verschiedene Eigenschaften der Objekte bezogen. OPC UA greift nicht mehr alleine auf Items zu, sondern auf Objekte und deren Unterobjekte.

- Datenvariablen, Methoden und zum Teil Ereignisse sind beispielsweise Unterobjekte eines S7-Verbindungsobjekts. Attribute und Properties definieren die Objekte näher.
- Ein OPC-Data-Access-Item für den Bausteinzugriff entspricht dabei am ehesten einer OPC-UA-Datenvariablen.
- Ein OPC-Data-Access-Item für Domain-Dienste entspricht am ehesten einer OPC-UA-Methode.

Den qualifizierten Bezeichnern der Nodelds kommt unter OPC UA eine größere Bedeutung als unter OPC Data Access zu. Jeder einzelne Zugriff auf ein Objekt, Unterobjekt, Property und Attribut erfolgt über dessen Nodeld.

Unter anderem für die Unterstützung durch lokale Sprachen sieht OPC UA den Anzeigenamen vor. So kann ein und dasselbe Objekt beispielsweise in unterschiedlichen Sprachumgebungen, die der OPC-UA-Client vorgibt, unterschiedlich durchsucht werden, wobei aber jedes Mal die selbe Nodeld präsentiert wird. Der Anzeigename wird analog zur jeweiligen Nodeld gewählt. Die Texte des gesamten Namensraums sind in Englisch.

## Syntax der S7-OPC-UA-Datenobjekte

Unter OPC UA wird eine optimierte Syntax eingeführt. Die Nodelds aller OPC-UA-Objekte haben folgenden Aufbau:

*<verbindungsobjekt>".<unterobjekt>".<property>*

Ein Unterobjekt kann weitere Unterobjekte beinhalten.

Eine nicht interpretierbare Nodeld wird mit einem Fehler zurückgewiesen. Die Groß- oder Kleinschreibung der Buchstaben "A-Z" wird bei allen Items ignoriert.

## Symbolische Objektdarstellung

Die OPC-UA-Spezifikation empfiehlt zur hierarchischen Beschreibung des Adressraums eine einheitliche Symboldarstellung. Folgende Symbole werden in diesem Dokument verwendet:

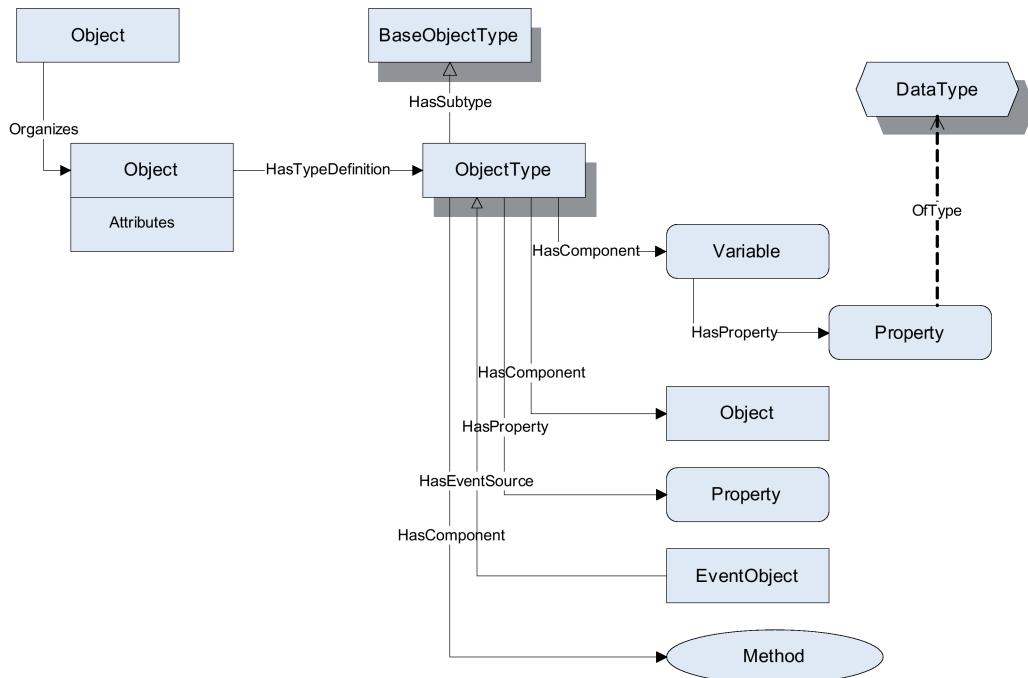


Bild 2-25      Symbole des OPC-UA-Adressraums

## 2.7.6 Verbindungsobjekte

### S7-Verbindungs-Objekte

Es gibt folgende S7-Verbindungs-Objekte:

- Produktive S7-Verbindungen

S7-Verbindungen werden zum Datenaustausch zwischen Automatisierungsgeräten genutzt und im Allgemeinen über STEP 7 projektiert.

- Die DEMO-Verbindung

Sie dient ausschließlich zum Test.

- Die @LOCALSERVER-Verbindung

Sie stellt die lokalen S7-Datenbausteine für die S7-Serverfunktionalität zur Verfügung.

## Demo-Verbindung

<Verbindungsobjekt>:= "DEMO"

Unter der Demo-Verbindung mit dem Namen DEMO gibt es Objekte, die einen ähnlichen Namensraum wie S7-Verbindungen beinhalten. Die Demo-Verbindung ist zum Kennenlernen des SIMATIC NET OPC-Systems gedacht und kann über die Konfiguration hinzugeschaltet werden.

---

### Hinweis

Die Demo-Verbindung mit dem Namen DEMO darf nicht gleichzeitig mit einer S7-Verbindung gleichen Namens eingesetzt werden. Eine mit diesem Namen projektierte S7-Verbindung wird ignoriert, wenn die Demo-Verbindung hinzukonfiguriert wurde.

---

## LocalServer-Verbindung

<Verbindungsobjekt>:= "@LOCALSERVER"

Unter der virtuellen LocalServer-Verbindung mit dem Namen "@LOCALSERVER" gibt es Datenvariablen, die einen Zugriff auf die lokal vom Server verwalteten Datenbausteine ermöglichen. Nach der Installation ist nur der Datenbaustein DB1 verfügbar.

Die Syntax der Datenvariablen ist identisch zu den S7-Variablen für Datenbausteine, z.B.:  
@LOCALSERVER.DB1.100,B

Die S7-Variablen für Datenbausteine sind weiter unten beschrieben.

Weitere auf einer normalen S7-Verbindung vorhandene Datenvariablen und Methoden wie z.B. Statepath, Block- oder Domain-Dienste gibt es nicht.

---

### Hinweis

Der Verbindungsname "@LOCALSERVER" ist reserviert und darf niemals als Name einer S7-Verbindung eingesetzt werden.

---

## 2.7.6.1 Verbindungsnamen

### Der Verbindungsname einer S7-Verbindung

Der Verbindungsname ist der in STEP 7 oder COML S7 projektierte Name zur Identifikation der Verbindung. Dieser Name heißt bei STEP 7 "Lokale ID". Die Lokale ID ist innerhalb des OPC-Servers eindeutig.

### Verbindungstypen

Der OPC-Server unterstützt folgende Verbindungstypen:

- S7-Verbindung
- S7-Verbindung hochverfügbar

### **Welche Zeichen sind für S7-Verbindungsnamen erlaubt?**

Für den <Verbindungsname> sind Ziffern "0-9", alphabetische Zeichen in Groß- und Kleinschreibung "A-z" und Sonderzeichen " \_-+()" erlaubt. Der Verbindungsname darf 24 Zeichen lang sein. Groß- und Kleinschreibung wird nicht unterschieden.

Weitere sichtbare Zeichen sind nicht erlaubt.

Die Verbindungsnamen "SYSTEM" bzw. der Verbindungsname "@LOCALSERVER" sind reserviert und dürfen nicht verwendet werden.

### **Beispiele für Verbindungsnamen:**

Typische Beispiele sind:

- S7-Verbindung\_1
- S7-OPC-Verbindung

## **2.7.7 Aufbau und Funktionen des produktiven S7-Verbindungsobjekts**

### **Was sind S7-Verbindungsobjekte?**

Alle produktiven protokollspezifischen Objekte sind immer einer Verbindung zugeordnet. Bei S7 sind dies die Verbindungen zu Kommunikationspartnern (S7-Verbindung). Ausnahmen hiervon bilden die Systemverbindung und die Demoverbindung.

### 2.7.7.1 Typ-Definition des S7-Verbindungsobjekts

#### Typ-Definition des S7-Verbindungsobjekts

Für die Objekte und Funktionalitäten, die über eine produktive S7-Verbindung verwendbar sind, ist ein spezifischer OPC-UA-Objekttyp definiert:

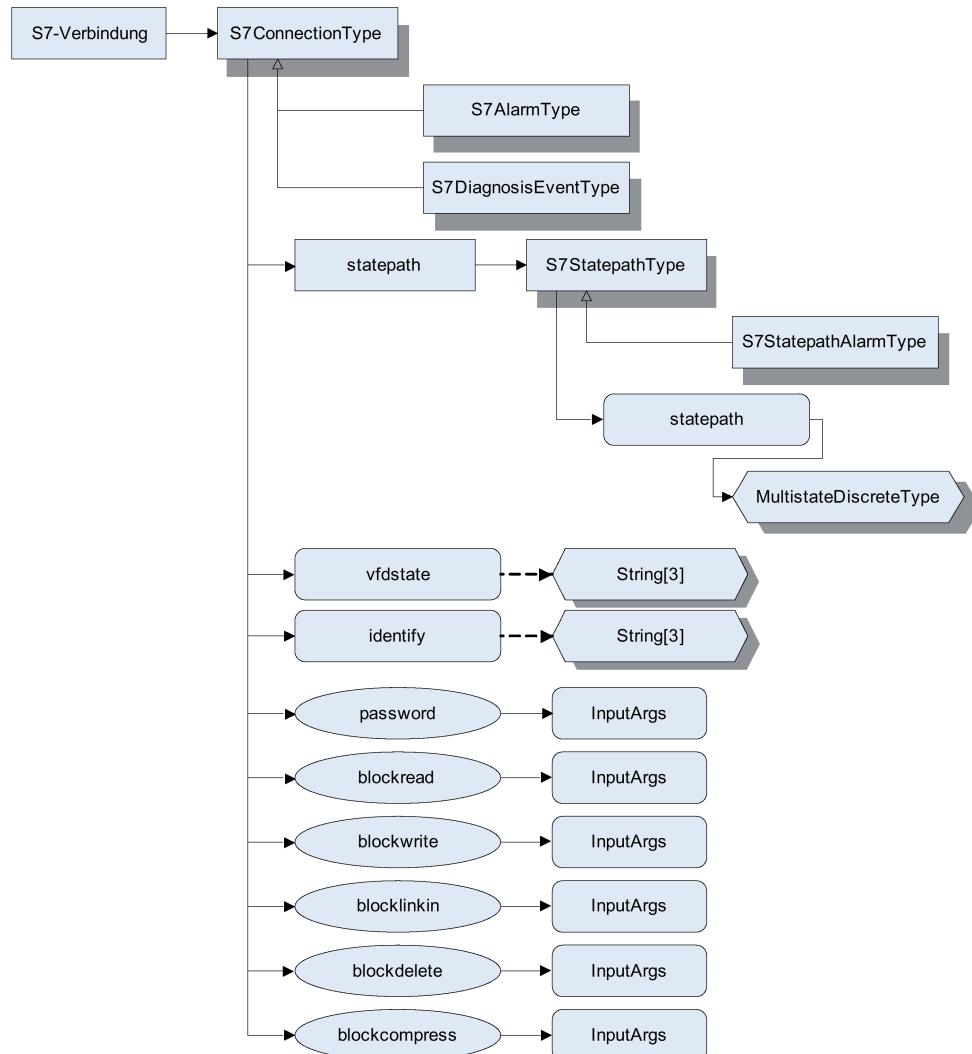


Bild 2-26 Der Typ des S7-Verbindungsobjekts im Namensraum von OPC UA

Im OPC-UA-Namensraum für Objekte werden Instanzen dieses Typs angezeigt. Der Typ selbst kann unter "Typen" strukturiert ausgelesen werden.

### 2.7.7.2 S7-Verbindungs-Informationsobjekte

#### S7-spezifische Datenvariablen für Informationen

Es gibt S7-spezifische Datenvariablen, mit denen Sie Informationen über die S7-Kommunikation und die aufgebauten Verbindungen abfragen können.

Folgende Informationen können ermittelt werden:

- Attribut eines virtuellen Geräts (VFD)
- Status einer S7-Verbindung
- Status eines virtuellen Geräts
- Status der Anmeldung für Meldungen

(siehe Parameter "events" in Kapitel "Diagnose- und Konfigurations-Informationen (Seite 192)".)

#### Syntax der S7-spezifischen Informationsvariablen

Nodeld:

*Namensraum-Index: 3 // für Siemens.Automation.SimaticNET.S7*

*<Verbindungsobjekt>.<Informationsparameter>*

#### Erklärungen

*<Informationsparameter>:= "identify"|"vfdstate"|"statepath"*

identify	Herstellerattribute eines Kommunikationspartners. Datentyp Array Of String (3 Arrayelemente), ReadOnly. "identify" kann beispielsweise folgende Werte zurückgeben:	
	Hersteller	Siemens AG
	Modell	6ES7 416-3XR05-0AB0
	Revision	V5.0
	Zustand eines virtuellen Geräts Datentyp Array Of String (3 Arrayelemente), ReadOnly.	
vfdstate	Logischer Status	S7_STATE_CHANGES_ALLOWED Alle Dienste sind zulässig.
	Physikalischer Status	S7_OPERATIONAL Das reale Gerät ist einsetzbar. S7_NEEDS_COMMISSIONING Das reale Gerät ist erst nach dem Abschluss lokaler Eingriffe einsetzbar.
	Detailinformationen VFD-Status, geräteabhängig	<bytestring(3)> (Feld von 3 Byte) Der Status wird als Oktett-String zurückgegeben. Weitere Informationen zur Bedeutung des Rückgabewerte müssen der Dokumentation des Partnergeräts entnommen werden.

statepath	Zustand einer Kommunikationsverbindung zum Partnergerät Der Wert der Variable wird als Zahl ausgelesen und kann durch zusätzliches Auslesen des zugehörigen Enumstring {UNKNOWN, DOWN, UP, RECOVERY, ESTABLISH} einem Text zugeordnet werden. Variable vom UA-Typ MultistateDiscreteType, ReadOnly		
	0	UNKNOWN	Für zukünftige Erweiterungen reserviert
	1	DOWN	Verbindung ist nicht aufgebaut
	2	UP	Verbindung ist aufgebaut
	3	RECOVERY	Verbindung ist nicht aufgebaut. Es wird versucht, die Verbindung aufzubauen.
	4	ESTABLISH	Für zukünftige Erweiterungen reserviert

### 2.7.7.3 Beispiele für S7-spezifische Informationsvariablen und Rückgabewerte

Hier finden Sie Beispiele, welche die Syntax der Namen von S7-spezifischen Informationsvariablen verdeutlichen.

#### Informationen über die Herstellerattribute eines virtuellen Geräts

- Nodeld:
  - Namensraum-URI:  
*S7: (Namensraum-Index: 3) // für Siemens.Automation.SimaticNET.S7*
  - Bezeichner:  
*S7-Verbindung\_1.identify*

#### Mögliche Rückgabewerte:

- Hersteller: *SIEMENS AG*
- Modell des virtuellen Geräts: *6ES7413-1AE0-0AB0*
- Revision: *V1.0*

#### Zustand eines Geräts

- Nodeld:
  - Namensraum-URI:  
*S7: (Namensraum-Index: 3) // für Siemens.Automation.SimaticNET.S7*
  - Bezeichner:  
*S7-Verbindung\_1.vfdstate*

#### Mögliche Rückgabewerte:

- Logischer Status: *S7\_STATE\_CHANGES\_ALLOWED*  
*Alle Dienste sind zulässig.*
- Physikalischer Status: *S7\_OPERATIONAL*  
*Das reale Gerät ist einsatzbereit.*
- Detailinformationen: *02.00.00*  
*Detailinformationen zum lokalen VFD-Status*

## Zustand einer Kommunikationsverbindung als String

- Nodeld:
  - Namensraum-URI:  
*S7: (Namensraum-Index: 3) // für Siemens.Automation.SimaticNET.S7*
  - Bezeichner:  
*S7-Verbindung\_1.statepath*

### Mögliche Rückgabewerte vom Typ "MultistateDiscreteType":

- Verbindungsstatus: *RECOVERY*  
*Die Verbindung wird momentan aufgebaut.*

## 2.7.7.4 Methoden für die S7-Bausteindienste

### Was machen Bausteindienste?

Bausteindienste steuern die Übertragung und Rückübertragung von Daten- und Programmelementen zwischen PC und Automatisierungsgerät. Auf dem PC werden Daten und Programmelemente in Dateien gespeichert. Bausteindienste können mit einem Passwort geschützt werden.

### Bausteine

Daten und Programmelemente werden auf S7-Automatisierungsgeräten in Bausteinen abgelegt. Diese Bausteine werden mit Hilfe von STEP 7 generiert und auf das S7-Gerät übertragen. Bei S7-Geräten gibt es folgende Bausteine:

- Organisationsbausteine (OB)
- Funktionsbausteine (FB)
- Funktionen (FC)
- Datenbausteine (DB / DI)

### Aufgaben mit Bausteinen

Sie können folgende Aufgaben über den SIMATIC NET OPC-Server ausführen:

- Bausteine zwischen PC und Automatisierungsgerät übertragen
- Bausteine löschen
- Bausteine einketten
- Speicher des Automatisierungsgeräts komprimieren

---

### Hinweis

Die Erstellung von Bausteinen ist nicht mit dem OPC-Server möglich. Hierzu müssen Sie STEP 7 verwenden.

---

OPC UA unterstützt produktspezifische Methoden. Die S7-Bausteindienste entsprechen dem Ausführen von Methoden.

## Syntax der Methoden für die S7- Bausteindienste

Nodeld:

*Namensraum-URI: S7: (Namensraum-Index: 2)*  
für Siemens.Automation.SimaticNET.S7

**<Verbindungsname>. <DomainMethod>**

**<DomainMethod>:=**

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

Parameter	Bedeutung	
password()	Passwort für Domain-Dienste an Partnergerät übertragen InputArgument1: "password"; Datentyp: ByteString; WriteOnly Um das Passwort an den Verbindungspartner zu übertragen, muss als Passwortargument eine entsprechende Zeichenfolge übergeben werden. Für die Zeichenfolge gibt es zwei Möglichkeiten	
	"" (leere Zeichenfolge)      Passwort wird zurückgesetzt <bytestring(8)>      Passwort in Bytestring-Darstellung (Feld von 8 Byte)	
blockread()	Ein Baustein von einem Automatisierungsgerät wird auf den PC übertragen und dort in einer Datei abgelegt: InputArgument1: "flags", Datentyp UInt32 InputArgument2: "block", Datentyp String InputArgument3: "filename"; Datentyp: String	
	Flags <unsigned32> Folgende hexadezimale Zahlen sind möglich: 0x0001 Ein nicht eingeketteter Block wird gelesen. Wenn eine Zielfile besteht, wird diese nicht überschrieben. Eine Fehlermeldung wird ausgegeben. 0x0040 Ein eingeketteter Block wird gelesen. Wenn eine Zielfile besteht, wird diese nicht überschrieben. Eine Fehlermeldung wird ausgegeben. 0x1001 Ein nicht eingeketteter Block wird gelesen. Eine schon vorhandene Zielfile wird überschrieben. 0x1040 Ein eingeketteter Block wird gelesen. Eine schon vorhandene Zielfile wird überschrieben.	
	Bausteintyp und Nummer: "OB"<unsigned16> "FB"<unsigned16>  "FC"<unsigned16> "DB"<unsigned16>	
	Vollständiger Pfad der Datei, in der der Baustein abgelegt ist: <string(511)>	

Parameter	Bedeutung	
blockwrite()	<p>Ein Baustein von einem PC wird auf ein Automatisierungsgerät übertragen. Nach der Übertragung befindet sich der Baustein auf dem Automatisierungsgerät im passiven (nicht eingeketteten) Zustand. Mit der Funktion <code>&amp;blocklinkin</code> muss der Baustein vom passiven Zustand in den eingeketteten Zustand gesetzt werden.</p> <p>InputArgument1: "flags", Datentyp UInt32 InputArgument2: "filename", Datentyp String</p>	
	Flags <unsigned32>	Flags Folgende hexadezimale Zahlen sind möglich: 0x1000 Ein auf dem Automatisierungssystem bestehender nicht eingeketteter Baustein gleichen Namens soll überschrieben werden. 0x0000 Ein auf dem Automatisierungssystem bestehender nicht eingeketteter Baustein gleichen Namens soll nicht überschrieben werden.
	<p>Vollständiger Pfad der Datei, in welcher der Baustein abgelegt ist. Datentyp &lt;string(511)&gt;</p>	
blocklinkin()	<p>Baustein vom passiven Zustand in den Programmablauf des Automatisierungssystems einketten. Dabei wird der ablaufrelevante Teil des Bausteins in den Ablaufspeicher des Automatisierungssystems kopiert.</p> <p>Der Baustein ist danach für das Programm auf dem Automatisierungssystem erreichbar. Durch das Einketten eines Bausteins wird ein bestehender, aktiver Baustein ohne Fehlermeldung überschrieben.</p> <p>Der geschriebene Wert erhält als Parametrierung dieses Dienstes die Angabe des Blocks, der eingekettet werden soll.</p> <p>InputArgument1: "block" Datentyp String</p>	
	<p>Bausteintyp und Nummer: "OB"&lt;unsigned16&gt;"FB"&lt;unsigned16&gt; "FC"&lt;unsigned16&gt;"DB"&lt;unsigned16&gt;</p>	
blockdelete()	<p>Baustein in einem Automatisierungsgerät löschen</p> <p>InputArgument1: "flags" Datentyp UInt32 InputArgument2: "block" Datentyp String</p>	
	Flags Datentyp VT_BSTR <unsigned32>	Flags Folgende hexadezimale Zahlen sind möglich: 0x0001 Ein nicht eingeketteter Block wird gelöscht. 0x0040  Ein eingeketteter Block wird gelöscht. 0x0041 Der eingekettete und der nicht eingekettete Block werden gelöscht.
	<p>Bausteintyp und Nummer Datentyp VT_BSTR "OB"&lt;unsigned16&gt;"FB"&lt;unsigned16&gt; "FC"&lt;unsigned16&gt;"DB"&lt;unsigned16&gt;</p>	
blockcompress()	<p>Komprimierung des Speichers des Automatisierungsgeräts "" (leere Zeichenfolge)</p>	

---

**Hinweis**

Die Baustein-Dienste "blockread" und "blockwrite" erlauben keinen Zugriff auf Quell- und Zielfilen der Bausteine auf einem Netlaufwerk.

Das Überschreiben (0x1000) eines Blocks ist erst möglich, nachdem der ablaufrelevante Teil eingekettet wurde.

---

**Passwörter**

Schreibende und lesende Zugriffe von Bausteindiensten auf die CPU können bei der Projektierung mit einem Passwort versehen werden. Das Passwort hat eine höhere Priorität als der Schlüsselschalter der CPU.

Zu Passwörtern und Schutzstufen siehe Kapitel "Passwörter (Seite 147)".

**2.7.8 Variabldienste****2.7.8.1 Variabldienste****Was machen Variabldienste?**

Variabldienste ermöglichen den Zugriff und die Beobachtung von S7-Variablen im Automatisierungsgerät. Die Adressierung der S7-Variablen erfolgt als Namenskürzel der adressierten Objekte. Die Art des Zugriffs orientiert sich an der Notation der S7-Werkzeuge.

Objekte im Automatisierungsgerät der S7-OPC-UA-Server unterstützten folgende Objekte:

- Datenbausteine
- Instanzdatenbausteine
- Eingänge
- Ausgänge
- Peripherieeingänge
- Peripherieausgänge
- Merker
- Timer
- Zähler

Nicht jedes S7-Automatisierungsgerät unterstützt alle Objekttypen.

## 2.7.8.2 Syntax der Variabldienste

### Syntax der Prozessvariablen

Vereinfachte Syntax der Prozessvariablen der S7-OPC-UA-Node-ID:

*Namensraum-URI: S7: (Namensraum-Index: 3)*

### Klassische Syntax

Es gibt drei Möglichkeiten:

- <Verbindungsname>.<S7Objekt>.<Adresse>{,<S7Typ>{,<Anzahl>}}
- <Verbindungsname>.<S7TimerObjekt>.<Adresse>{,<S7TimerTyp>{,<Anzahl>}}
- <Verbindungsname>.<S7CounterObjekt>.<Adresse>{,<S7CounterTyp>{,<Anzahl>}}

### Erklärungen

Namensraum-URI: S7: (Namensraum-Index: 3)

#### <Verbindungsname>

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

Das folgende Trennzeichen ist der Punkt (".").

#### <S7Objekt>

Angabe des Bereichstyps in der S7. Mögliche Werte sind:

Parameter	Bedeutung
db<Nr>	Datenbaustein oder Instanzdatenbaustein. Kennzeichen für eine S7-Variable aus einem Datenbaustein. Instanzdatenbausteine unterscheiden sich in der S7-Kommunikation nicht von normalen Datenbausteinen. Deshalb kann auf die Vergabe einer zusätzlichen Kennung zur Vereinfachung verzichtet werden. <Nr> Nummer des Datenbausteins oder Instanzdatenbausteins ohne führende Nullen.
m	Merker
i	Eingang Schreib- und lesbar Es gibt vereinfachend nur diese englische Bezeichnung.
q	Ausgang Schreib- und lesbar Es gibt vereinfachend nur diese englische Bezeichnung.

Parameter	Bedeutung
pi	Peripherie-Eingang Nur lesbar. Es gibt vereinfachend nur diese englische Bezeichnung.
pq	Peripherie-Ausgang Nur schreibbar. Es gibt vereinfachend nur diese englische Bezeichnung.

Das folgende Trennzeichen ist der Punkt (".").

#### <S7TimerObjekt>

Parameter	Bedeutung
t	Timer. Wort (unsigned). Die folgende Adressangabe ist eine Timernummer.

#### <S7CounterObjekt>

Parameter	Bedeutung
c	Zähler. Wort (unsigned). Die folgende Adressangabe ist eine Zählernummer.

#### <Adresse>

Byte-Adresse der ersten Variablen, die angesprochen werden soll. Es werden keine führenden Nullen (z.B. 001) unterstützt.

Der Wertebereich für die Byte-Adresse ist 0...65534. Geräte- und typabhängig kann der tatsächlich verwendbare Wert der Adresse geringer sein.

#### <S7Typ>

S7-Datentyp

Ein S7-Datentyp wird im OPC-UA-Server in den entsprechenden OPC-UA-Datentyp umgewandelt. Die folgende Tabelle listet den Typ-"Bezeichner" und den entsprechenden OPC-UA-Datentyp auf, in dem der Variablenwert dargestellt werden kann.

S7-Datentyp <S7Typ>	OPC-UA-Datentyp	Beschreibung
b	Byte	Byte (unsigned) Wird als Defaultwert verwendet, falls kein <S7Typ> angegeben ist.
w	UInt16	Wort (unsigned)
c	SByte	Byte (signed)
i	Int16	Wort (signed)
di	Int32	Doppelwort (signed)

S7-Datentyp <S7Typ>	OPC-UA-Datentyp	Beschreibung
dw	UInt32	Doppelwort (unsigned)
r	Float	Fließkomma (4 Byte)
dt	DateTime	Datum und Uhrzeit, Wertebereich ab 01.01.1990, (auf CPU DATE_AND_TIME)
date	DateTime	Datum und Uhrzeit (8 Byte), wobei die Uhrzeit immer 00:00:00 ist, Wertebereich ab 01.01.1990. Abbildung des CPU Datentyps DATE (unsigned, 16 Bit).
t	Int32	Vorzeichenbehafteter Zeitwert in Millisekunden
tod	Int32	Tageszeit, 0...86399999 ms ab Mitternacht
s5bcd	UInt16	Abbildung des CPU Datentyps S5TIME auf UInt 16 (unsigned, 16 Bit) mit eingeschränktem Wertebereich, 0...9990000 ms.*)
x<Bitadresse>	Boolean	Bit (bool) Zusätzlich zum Byte-Offset im Bereich ist noch die <Bitadresse> im jeweiligen Byte anzugeben. Wertebereich 0...7
s<Stringlänge>	String	Es ist noch die für den String reservierte <Stringlänge> anzugeben. Wertebereich 1...254 Beim Schreiben können auch kürzere Strings geschrieben werden, wobei die übertragene Datenlänge immer die reservierte Stringlänge in Byte zuzüglich 2 Byte ist. Die nicht benötigten Bytes werden mit dem Wert 0 gefüllt. Das Lesen und Schreiben von Strings und String-Arrays wird intern auf das Lesen und Schreiben von Byte-Arrays abgebildet. Der String muss auf der S7 mit gültigen Werten initialisiert sein.

\*) Siehe nachfolgende Tabelle "Zeitraster und Wertebereich für den S7-Datentyp s5btcd"

Die Datentypen "c", "i", "di" und "r" können in einem Datenbaustein (db) oder einem Instanzdatenbaustein verwendet werden, für Merker (m), für Eingänge (i), für Ausgänge (q), für Peripherieeingänge (pi) und für Peripherieausgänge (pq).

### Zeitraster und Wertebereich für den S7-Datentyp s5bcd:

Der Wertebereich der Zeit-Variable vom Datentyp s5bcd ist BCD-kodiert. Der Wertebereich ergibt sich entsprechend der folgenden Tabelle:

Bit-Nr.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bedeutungssymbol	0	0	x	x	z	z	z	z	z	z	z	z	z	z	z	z
<b>Erläuterung:</b>																
Bedeutungssymbol "0"	nicht relevant															
Bedeutungssymbol "x"	Angabe des Zeitrasters															
	Bit 13 und 12								Zeitraster in Sekunden							
	00								0,01							
	01								0,1							
	10								1							
	11								10							
Bedeutungssymbol "z"	BCD-kodierter Zeitwert (0...999)															

### Beispiel:

Bit-Nr.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Wert	0	0	0	1	0	0	0	0	0	1	1	1	0	1	0	1

Bit 0-11 legen die Zahl 075 fest. Bit 12 und 13 legen das Zeitraster 0,1 fest.  
 $75 * 0,1 = 0,75$  Sekunden

Der OPC-Datentyp der Zeit-Variable vom Datentyp s5bcd ist ein Wort (unsigned, UInt16). Beim Schreiben ist der Wertebereich entsprechend eingeschränkt.

### <S7TimerTyp>

S7-Datentyp <S7Typ>	OPC-UA-Datentyp	Beschreibung
tbcd	UInt16	Timer, BCD-codiert Wird als Defaultwert verwendet, falls kein <S7TimerTyp> angegeben ist.
tda	UInt16[2]	Timer, Dezimale Zeitbasis und Zeitwert

Zeitraster und Wertebereich von S7-Timer-Variablen "t", "tbcd" (<S7TimerObjekt> = t, <S7TimerTyp> = tbcd).

Der Wertebereich von OPC-Prozessvariablen für S7 vom Typ Timer (t) ist BCD-kodiert. Aus dem Wertebereich ergibt sich (für das Schreiben) das Zeitraster entsprechend der folgenden Tabelle:

Bit-Nr.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bedeutungssymbol	0	0	x	x	z	z	z	z	z	z	z	z	z	z	z	z
<b>Erläuterung:</b>																
Bedeutungssymbol "0"	nicht relevant															

Bedeutungssymbol "x"	Angabe des Zeitrasters														
	Bit 13 und 12				Zeitraster in Sekunden										
	00				0,01										
	01				0,1										
	10				1										
	11				10										
Bedeutungssymbol "z"	BCD-kodierter Zeitwert (0...999)														

**Beispiel:**

Bit-Nr.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Wert	0	0	0	1	0	0	0	0	0	1	1	1	0	1	0	1

Bit 0-11 legen die Zahl 075 fest. Bit 12 und 13 legen das Zeitraster 0,1 fest.

$$75 * 0,1 = 0,75 \text{ Sekunden}$$

**Zeitraster und Wertebereich von S7-Timer-Variablen "t", "tda" ( $\langle\text{S7TimerObjekt}\rangle = t$ ,  $\langle\text{S7TimeTyp}\rangle = tda$ ):**

Mit der einfachen Timer-Variablen vom Typ "t" lassen sich Timer zwar einfach bedienen. Es sind aber nicht alle möglichen Kombinationen von Zeitraster und Wertebereich einstellbar, da die Wertebereiche überlappen dürfen. Für diesen Fall kann die S7-Timer-Variablen vom Typ "tda" (Decimal Array) verwendet werden.

Datentyp: Feld von zwei Worten {Zeitraster in Millisekunden als UInt16 | Zeitwert UInt16}.

Tabelle 2- 3 Wertebereich

Zeitraster [ms]	10, 100, 1 000, 10 000
Zeitwert	0...999 0 ist erlaubt aber ohne Funktion.
Zeitbereiche [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

Bei dem Objekt TDA ist keine Eingabe der  $\langle\text{Anzahl}\rangle$  möglich.

**Beispiel:**

Beschreiben des Timers "T.3", "tda" mit Wert {100|50} initialisiert den Timer 3 mit dem Wert  $50 * 100 \text{ ms} = 5000 \text{ ms}$  und dieser taktet 50 mal in 100-ms-Stufen herunter.

Mit den Typen "t" und "tbcd" ist diese Einstellung nicht möglich.

 **$\langle\text{S7CounterTyp}\rangle$** 

c	UInt16	Zähler Wertebereich für S7: 0...999, dezimal kodiert Wird als Defaultwert verwendet, falls kein $\langle\text{S7CounterTyp}\rangle$ angegeben ist.
---	--------	--

**<Anzahl>**

Anzahl der Variablen eines Typs, die ab dem im Parameter "Adresse" angegebenen Offset angesprochen werden sollen (Wertebereich 1...65 535).

Die Angabe einer Anzahl von Arrayelementen führt immer zur Bildung eines Feldes vom entsprechenden Typ, auch wenn nur ein einziges Feldelement adressiert wird.

Das Trennzeichen ist ein Komma (",").

Beim Datentyp "x" ist die Eingabe der Anzahl für Schreibzugriff nur in Vielfachen von 8 möglich. Die Bit-Adresse muss dann Null sein.

Beim Datentyp "x" ist die Eingabe der Anzahl für Lesezugriff nicht eingeschränkt. Der Wertebereich der Bit-Adresse erlaubt dann 0...7.

Bei dem Objekt TDA ist keine Eingabe der <Anzahl> möglich.

**Beispiele:**

Namensraum-URI: *S7:(Namensraum-Index: 3)*

*S7-OPC-1.db1.10,x0,64*, Zugriffsrechte RW

*S7-OPC-1.db1.10,x3,17*, Zugriffsrechte R

### 2.7.8.3 Beispiele für Prozessvariablen für S7-OPC-UA-Variablen-Dienste

Hier finden Sie Beispiele, die die Syntax von Variablennamen für Variablen-Dienste verdeutlichen.

#### Datenbaustein DB Einzel-Byte

*Namensraum-URI: S7:(Namensraum-Index: 3) // für Siemens.Automation.SimaticNET.S7*

*S7-Verbindung-1.db5.12,B*

bezeichnet Datenbyte 12 im Datenbaustein 5 über S7-Verbindung-1.

#### Datenbaustein DB, Feld von Wörtern

*Namensraum-URI: S7:(Namensraum-Index: 3)*

*S7-Verbindung-1.db5.10,w,9*

bezeichnet 9 Datenwörter ab Byteadresse 10 im Datenbaustein 5 über S7-Verbindung-1.

#### Datenbaustein DB, Feld von Strings

*Namensraum-URI: S7:(Namensraum-Index: 3)*

*S7-Verbindung-1.db100.50,s32,3*

bezeichnet 3 Strings der Länge 32 ab Byteadresse 50 im Instanzdatenbaustein 100 über S7-Verbindung-1.

## Eingang 0

*Namensraum-URI: S7: (Namensraum-Index: 3)*

*S7-Verbindung-1.i.0*

bezeichnet das Eingangsbyte 0 über S7-Verbindung-1.

## Ausgang 0 Bit 0

*Namensraum-URI: S7: (Namensraum-Index: 3)*

*S7-Verbindung-1.q.0,x0*

bezeichnet in Ausgangsadresse 0 das Bit 0 über S7-Verbindung-1.

## Feld von lesbaren Merkerbits

*Namensraum-URI: S7: (Namensraum-Index: 3)*

*S7-Verbindung-1.m.3,x4,12 Zugriffsrechte R*

bezeichnet 12 Bits ab Merkeradresse 3 und dort ab Bitadresse 4 über S7-Verbindung-1. Nur lesbar.

## Timer 22 BCD-kodiert

*Namensraum-URI: S7: (Namensraum-Index: 3)*

*S7-Verbindung-1.t.22*

bezeichnet Timer 22, TBCD-Default über S7-Verbindung-1.

## 2.7.9 Blockorientierte Dienste

### Was machen blockorientierte Dienste?

Blockorientierte Dienste ermöglichen eine programmgesteuerte Übertragung größerer Datenblöcke. Die Übertragung wird durch Variablen realisiert:

- Variablen, die Datenblöcke empfangen
- Variablen, die Datenblöcke senden

Die Datenmenge beim Datentransfer beträgt bis zu 65 534 Byte, unabhängig von der Größe der PDU. Die Segmentierung der Daten wird von den Funktionen selbst übernommen.

---

### Hinweis

Blockorientierte Dienste können nur bei zweiseitigen Verbindungen verwendet werden. Die erstellte Verbindungsprojektierung muss in das S7-Automatisierungsgerät geladen werden.

### 2.7.9.1 Syntax der blockorientierten Dienste

#### Vereinfachte Syntax

Vereinfachte Syntax der Prozessvariablen S7-OPC-UA-Nodeld:

Namensraum-URI: *S7:(Namensraum-Index: 2)* für "Siemens.Automation.SimaticNET.S7"

#### Klassische Syntax

Es gibt folgende Möglichkeiten:

- <Verbindungsname>.brcv<rid>{.<Adresse>{,{<S7Typ>}{},<Anzahl>}}}
- <Verbindungsname>.brcv<rid>{,<Adresse>{,{<S7Typ>}{},<Anzahl>}}}
- <Verbindungsname>.bsend<rid>.<Pufferlänge>{.<Adresse>{,{<S7Typ>}{},<Anzahl>}}}
- <Verbindungsname>.bsend<rid>.<Pufferlänge>{,<Adresse>{,{<S7Typ>}{},<Anzahl>}}}

#### Erklärungen

Parameter der Namensvariablen	Bedeutung
Verbindungsname	Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.
brcv	Datentyp: Array of Byte brcv enthält den zuletzt vom Partner empfangenen Datenblock. Inhalt und Länge der Empfangsdaten werden durch den sendenden Partner vorgegeben. Die Variable ist nur lesbar. Setzen Sie diese Variable zur Beobachtung ein. Damit wird der Empfang eines Datenblocks durch den OPC-Server an eine OPC-Client-Anwendung gemeldet.
bsend	Datentyp: Array of Byte BSEND enthält den Sendedatenblock zur Übertragung an ein Partnergerät. Der Datenblock wird erst dann an das Partnergerät übertragen, wenn die Variable geschrieben wird. Bei einem Lesezugriff wird der Inhalt des zuletzt erfolgreich übertragenen Datenblocks geliefert.
<rid>	ID des Adressierungsparameters. Sie ist für ein Bausteinpaar (BSEND/BRCV) festgelegt und innerhalb einer Verbindung eindeutig definiert. Sie können mehrere BSEND-Bausteine über eine Verbindung senden bzw. mehrere BRCV-Bausteine empfangen, aber immer mit einer unterschiedlichen ID. Gleiche IDs können für weitere Verbindungen verwendet werden.
<Pufferlänge>	Länge (in Byte) des Datenblocks, der gesendet werden soll. Es kann eine beliebige Anzahl Schreibpuffer unterschiedlicher Länge definiert werden. Damit ist es möglich, Teilbereiche unterschiedlicher Länge an den Partner zu schicken.

Parameter der Namensvariablen	Bedeutung
<Adresse>, <S7Typ>	<p>Die Datenpuffer können strukturiert werden. Sie können daraus einen oder mehrere Teilbereiche selektieren. Geben Sie hierzu den Datentyp &lt;S7Typ&gt; und die &lt;Adresse&gt; an, die beide auch für die S7-Datenbaustein-Variable gültig sind. Sie sind beide oben im Kapitel "Variabeldienste (Seite 122)" beschrieben.</p> <p><b>Lesen:</b> Da Struktur und Länge des vom Partner empfangenen Datenblocks nicht fest sein müssen, ist es nicht verboten, Variablen außerhalb des Bereichs zu definieren und anzufordern. Wenn der entsprechende Bereich bei einem Empfang von Daten nicht mehr ausgefüllt werden kann, dann wechselt die Quality der Variablen entsprechend.</p> <p><b>Schreiben:</b> Beim Schreibzugriff auf einen Teilbereich eines an den Partner zu sendenden Datenblocks wird immer der ganze Datenpuffer abgeschickt, dessen Länge über &lt;Pufferlänge&gt; spezifiziert wird. Wenn innerhalb eines OPC-UA-Mengenaufrufs mehrere Teilbereiche des Datenpuffers geschrieben werden, dann wird der Datenpuffer erst abgeschickt, nachdem alle seine Teilbereiche aktualisiert wurden. Nicht spezifizierte Teilbereiche werden aus dem Cache gefüllt.</p>
	<p><b>Hinweis</b> Im Datenpuffer werden die Datentypen im Motorola-Format interpretiert und zum Schreiben in das Intel-Format konvertiert.</p>
Anzahl	Anzahl der Feldelemente Wertebereich: 1...65 534

### Hinweise zum Lesen und Schreiben

- Das Lesen (brcv) und Schreiben (bsend) einzelner Bits (Format x) ist möglich.
- Das Lesen und Schreiben von Feldern einzelner Bits ist ohne Einschränkung der Anfangs-Bitadressen und mit beliebiger Feldlänge innerhalb des Datenblocks möglich.

Beispiel: S7-OPC-1.brcv1.10.x4,78

- Variablen für Sendedaten oder Empfangsdaten mit unterschiedlicher Länge, unterschiedlicher Adressierungsparameter-ID (RID) oder unterschiedlichem Verbindungsnamen verfügen über unabhängige Speicherbereiche.
- Wenn ein Item für einen unabhängigen Speicherbereich angelegt wird, dann wird der Sendedatenpuffer allokiert und mit Null initialisiert. Ein Schreibauftrag auf ein BSEND-Item wird in einen internen Schreibpuffer geschrieben und übertragen.

- Die Übertragung der Datenblöcke erfolgt azyklisch. Parallelie Netzaufträge für die gleichen Daten sind möglich. Es wird immer der vollständige Sendedatenblock übertragen. Dies gilt auch bei Zugriff auf ein Sub-Element oder wenn mehrere Clients gleichzeitig dieses Item beschreiben. Ein paralleles Schreiben des gleichen Sendedatenblocks oder von Teilbereichen des Sendedatenblocks durch mehrere Clients kann zu Inkonsistenzen führen.  
Es wird deshalb empfohlen, ...
  - immer den vollständigen Datenblock zu lesen oder zu schreiben oder
  - in NCM S7 die maximale Anzahl paralleler Netzaufträge auf 1 zu setzen. Dies reduziert jedoch die Übertragungsleistung.
- Wenn das Empfangen von Datenblöcken über den OPC-UA-Beobachtungsdienst gelesen wird (brcv als MonitoredItem), dann kann über die Konfiguration des Beobachtungsdienstes das Verhalten beim Empfang von Datenblöcken eingestellt werden. Vorausgesetzt der DataChangeTrigger ist im Beobachtungsdienst auf OpcUa\_DataChangeTrigger\_StatusValueTimestamp gesetzt, werden auch neu empfangene Datenblöcke mit gleichen Daten an den OPC-UA-Client gemeldet. Dadurch wird es einem Client ermöglicht, auch unveränderte gesendete Datenpuffer vom Partner zu erhalten. Die DataChange Notification erfolgt nicht schneller als die ausgehandelte Aktualisierungszeit (Update rate). Stellen Sie also hierfür immer Aktualisierungszeiten schneller als die Senderate der bsend/brcv-Daten ein.

### 2.7.9.2 Beispiele für Prozessvariablen für blockorientierte Dienste

Hier finden Sie Beispiele, die die Syntax von Variablennamen für blockorientierte Dienste verdeutlichen.

#### Datenblock empfangen im gesamten Puffer

*Namensraum-URI: S7: (Namensraum-Index: 3)*

*S7-OPC-1.brcv1*

Ein Datenblock wird im Empfangspuffer mit der RID 1 empfangen über S7-OPC-1. Der vollständige Puffer wird auf ein Feld von Bytes abgebildet.

#### Teilzugriff auf empfangenen Datenblock

*Namensraum-URI: S7: (Namensraum-Index: 3)*

*S7-OPC-1.brcv1.2,w,4*

Aus dem empfangenen Datenblock wird der Inhalt ab Adresse 2 auf ein Feld aus 4 Wörtern abgebildet. Insgesamt werden also 8 Byte aus dem Datenblock betrachtet.

#### Doppelwort übertragen

*Namensraum-URI: S7: (Namensraum-Index: 3)*

*S7-OPC-1.bsend1.16,2,d*

In einem Datenblock der Länge 16 mit der RID 1 wird ab Adresse 2 ein Doppelwort über S7-OPC-1 adressiert. Wenn ein Schreibbefehl auf die Variable angewendet wird, dann wird der geschriebene Wert an der angegebenen Position im Block eingetragen und der Datenblock wird gesendet.

### **Fließkommazahlen übertragen**

*Namensraum-URI: S7: (Namensraum-Index: 3)*

*S7S7-OPC-1.bsend1.32,20,r,2*

In einem Datenblock der Länge 32 mit der RID 1 wird ab Offset 20 ein Feld mit Fließkommazahlen über S7-OPC-1 adressiert. Wenn ein Schreibbefehl auf die Variable angewendet wird, dann wird der geschriebene Wert an der angegebenen Position im Block eingetragen und der Datenblock wird gesendet.

## 2.7.10 Baustein-Informations-Objekte einer S7-Verbindung

### 2.7.10.1 Längeninformationen

#### Die Bausteinobjekte unter einem S7-Verbindungsobjekt

Die Art und Größe des Baustinaufbaus in einem S7-Automatisierungsgerät wird zur Laufzeit ermittelt. Es gibt folgende Bausteinobjekte unter einem S7-Verbindungsobjekt, die diese Informationen einem OPC-UA-Client zur Verfügung stellen.

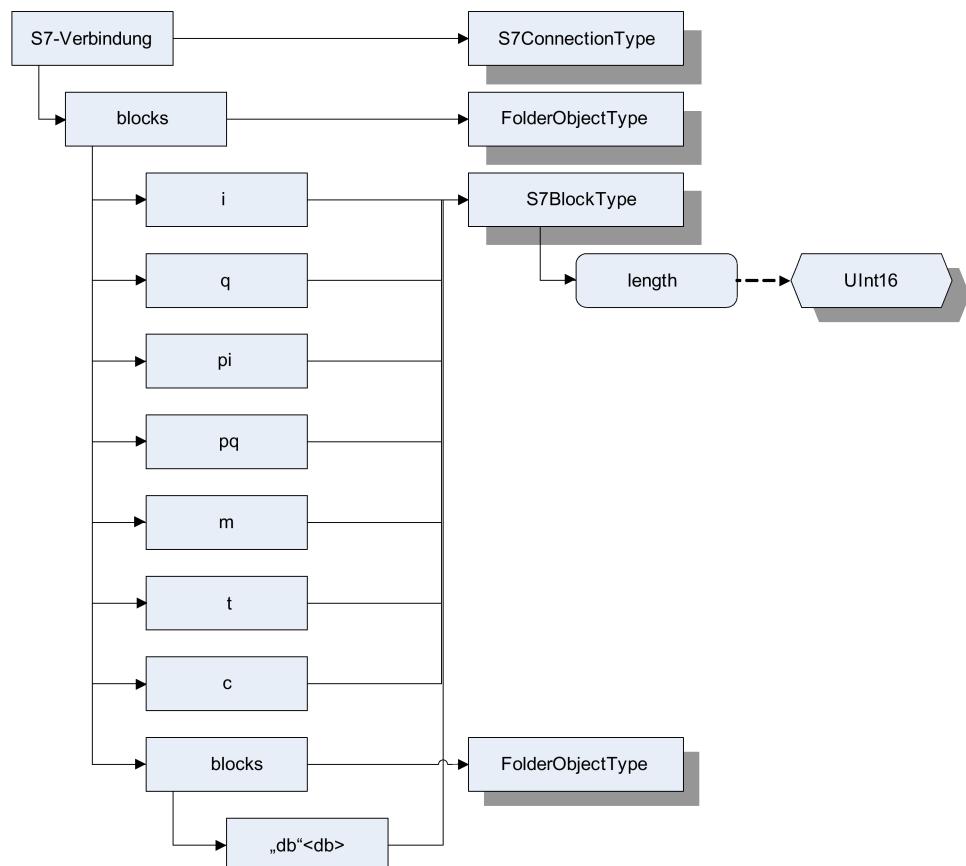


Bild 2-27 Bausteinobjekte unter einem S7-Verbindungsobjekt

Jedes Bausteinobjekt enthält dabei eine OPC-UA-Property mit einer Längen- bzw. Größenangabe über den Baustein (length).

#### Beispiel:

Namensraum-URI: s7: (→Namensraum-Index: 3)

*S7-Verbindung\_1.db10.length* //Property length des DB10-Bausteins (Länge in Byte)

## **2.7.10.2 Musterobjekte**

### **Das Musterobjekt eines Bausteinobjekts**

Für jedes beim Durchsuchen angezeigte Bausteinobjekt wird ein Musterobjekt angezeigt, dessen Nodeld als Vorlage für weitere benutzerdefinierte Datenobjekte verwendet werden kann. Das Musterobjekt besitzt den Standard-Datentyp B (bzw. c oder tbcd) für das jeweilige Bausteinobjekt und beginnt immer ab Adresse 0. Sollte dieses Item beim Verbindungspartner nicht zugreifbar sein, dann wird dies über entsprechende Zugriffsergebnisse und Quality-Codes angezeigt.

#### **Beispiel:**

Namensraum-URI: *S7: (Namensraum-Index: 3)*

*S7-Verbindung\_1.db10.0,b*

*S7-Verbindung\_1.m.0,b*

## **2.7.10.3 Diagnose- und Konfigurations-Informationen**

### **Die Properties eines S7-Verbindungsobjekts**

Im Allgemeinen werden die Eigenschaften einer S7-Verbindung mit dem Projektierungswerkzeug STEP 7 projektiert. Zur Laufzeit kann es sinnvoll sein, einige Projektierungsparameter auszuwerten.

Einige Projektierungsparameter werden für OPC UA als Properties zum S7-Verbindungsobjekt bereitgestellt:

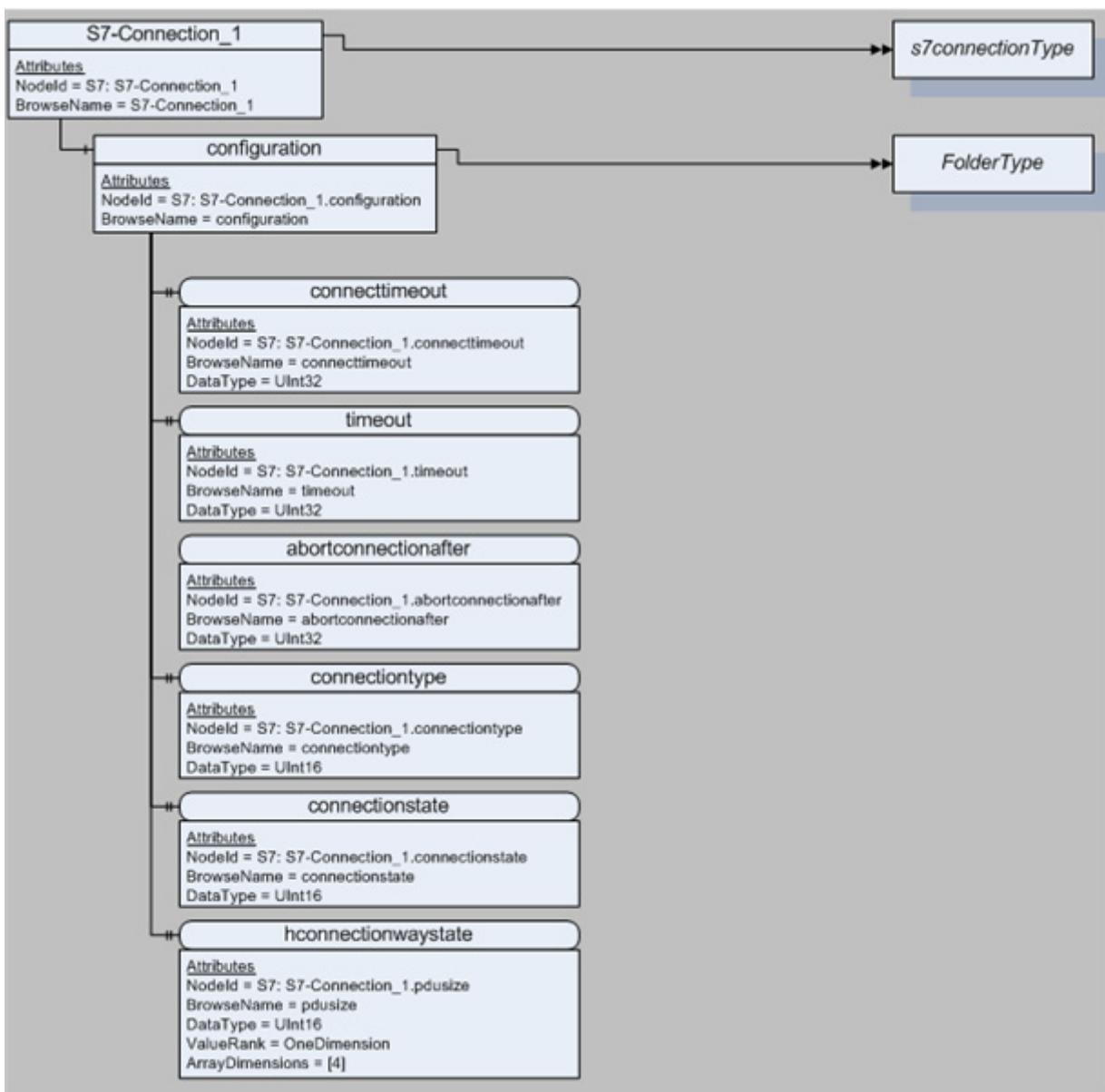


Bild 2-28 Properties eines S7-Verbindungsobjekts (Teil 1)

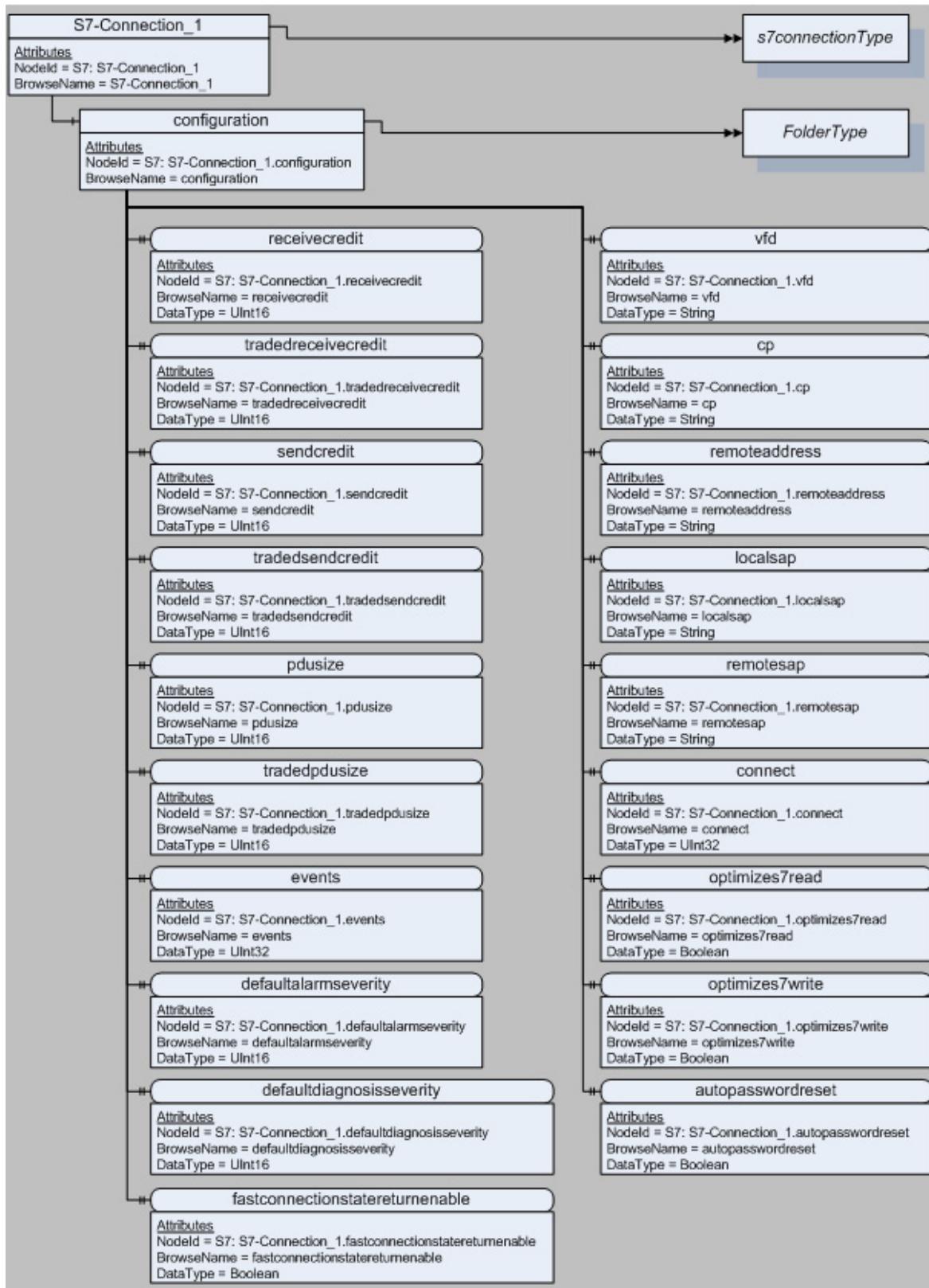


Bild 2-29 Properties eines S7-Verbindungsobjekts (Teil 2)

## Syntax zu Diagnose- und Konfiguration-Informationen

Namensraum-URI: s7: (--)Namensraum-Index: 3)

<Verbindungsname>. <S7VerbindungsProperty>

```
<S7VerbindungsProperty>:="vfd"|"cp"|"remoteaddress"|"localsap"|"remotesap"|
    "connect"|"autopasswordreset"|"fastconnectionstatereturnenable"|
    "sendcredit"|"receivecredit"|"pdusize"|
    "tradedsendcredit"|"tradedreceivecredit"|"tradedpdusize"|
    "connecttimeout"|"timeout"|"abortconnectionafter"|
    "optimizes7read"|"optimizes7write"|"defaultalarmseverity"|
    "defaultdiagnosiseverity"|"events"|"connectiontype"|
    "connectionstate"|"hconnectionwaystate"
```

S7-Verbindungsdiagnose-Property	Bedeutung
vfd	Name des OPC-Servers, dem die Verbindung zugeordnet ist. Üblicherweise hat dieser bei NCM-projektierten Verbindungen den Text "OPC Server". Datentyp String, nur lesbar.
cp	Name der Schnittstellenparametrierung, dem die Verbindung zugeordnet ist. Datentyp String, nur lesbar.
remoteaddress	Adresse des Verbindungspartners. Datentyp String, nur lesbar. Die Adresse des Verbindungspartners ist ein Datenpuffer mit einer vom Verbindungs-typ abhängigen Datenlänge. Für die übersichtlichere Auswertung durch den Anwender wird der Datenpuffer formatiert in einem String dargestellt. Profibus-Adresse Format: "ddd" (1-3 dezimale Ziffern) IP-Adresse (ISOOnTCP) Format: "ddd.ddd.ddd.ddd" (je 1-3 dezimale Ziffern) MAC-Adresse (ISO) Format: "xx-xx-xx-xx-xx-xx" (je 2 hexadezimale Ziffern)
localsap	Lokaler SAP der Verbindung. Datentyp String, nur lesbar. Der lokale SAP des Verbindungspartners ist ein Datenpuffer mit einer vom Verbin-dungstyp abhängigen Datenlänge. Für die übersichtlichere Auswertung durch den Anwender wird der Datenpuffer formatiert in einem String dargestellt. Format: "xx.xx " (je 2 hexadezimale Ziffern)
remotesap	Remoter SAP der Verbindung. Datentyp String, nur lesbar. Der remote SAP des Verbindungspartners ist ein Datenpuffer mit einer vom Verbin-dungstyp abhängigen Datenlänge. Für die übersichtlichere Auswertung durch den Anwender wird der Datenpuffer formatiert in einem String dargestellt. Format: "xx.xx " (je 2 hexadezimale Ziffern)
connect	Art des Verbindungsaufbaus. Datentyp UInt32, nur lesbar.

S7-Verbindungsdiagnose-Property	Bedeutung	
	0	Passiv, Verbindung wird permanent aufrecht erhalten.
	1	Aktiv, Verbindungsauflaufbau wird erst bei Bedarf hergestellt, Verbindungsabbau ohne Benutzung nach Wartezeit.
	2	Aktiv, Verbindung wird permanent aufrecht erhalten.
autopasswordreset	Automatisches Rücksetzen des S7-Passworts zum Bausteinzugriff Datentyp Boolean, nur lesbar. True: Rücksetzen aktiviert False: Rücksetzen deaktiviert In einer S7 bleibt eine Freischaltung der Domäindienste durch Passwort bis zu einem expliziten Rücksetzen aktiviert. Ein automatisches Rücksetzen des Passworts bei Verbindungsauflaufbau sorgt besonders im Zusammenwirken mit dem automatischen Verbindungsabbau nach unbenutztter Zeit dafür, dass das Passwort nicht unnötig lange freigegeben wird.	
fastconnectionstatereturnable	Schnelle Rückgabe eines Schreib/Lesezugriffs bei unterbrochener Verbindung. Datentyp Boolean, nur lesbar. True: aktiviert False: deaktiviert	
sendcredit	Maximale Anzahl paralleler Netzaufträge, Senderichtung Vorschlagswert für Verbindungsauflaufbau. Datentyp UInt16, nur lesbar. >=1, Vorschlagswert für Verbindungsauflaufbau Wird gemeinsam mit &receivecredit() über die Projektierung eingestellt.	
receivecredit	Maximale Anzahl paralleler Netzaufträge, Empfangsrichtung Vorschlagswert für Verbindungsauflaufbau. Datentyp UInt16, nur lesbar. >=1, Vorschlagswert für Verbindungsauflaufbau	
pdusize	Größe der Protokoll-PDU Vorschlagswert für Verbindungsauflaufbau Datentyp UInt16, nur lesbar. >=1, Vorschlagswert für Verbindungsauflaufbau	
tradedsendcredit	Anzahl paralleler Protokollauflaufträge, Senderichtung ausgehandelt nach Verbindungsauflaufbau. Datentyp UInt16, nur lesbar. Ist die Verbindung unterbrochen, so ist die Qualität dieser Property "BAD".	
tradedreceivecredit	Anzahl paralleler Protokollauflaufträge, Empfangsrichtung ausgehandelt nach Verbindungsauflaufbau. Datentyp UInt16, nur lesbar. Ist die Verbindung unterbrochen, so ist die Qualität dieser Property "BAD".	
tradedpdusize	Größe der Protokoll-PDU, ausgehandelt nach Verbindungsauflaufbau. Datentyp UInt16, nur lesbar. Ist die Verbindung unterbrochen, so ist die Qualität dieser Property "BAD".	

S7-Verbindungsdiagnose-Property	Bedeutung
connecttimeout	Verbindungsauftakt-Timeout Datentyp UInt32, nur lesbar. 0:kein Timeout >0:Timeout in ms
timeout	Auftrags-Timeout für den Produktivverkehr in ms. Datentyp UInt32, nur lesbar. 0:kein Timeout >0:Timeout in ms
abortconnectionafter	Automatischer Verbindungsabbau. Verzögerungszeit für den automatischen Verbindungsabbau: Der OPC-Server baut die Verbindung nach dieser Zeit selbstständig wieder ab, sofern in dieser Zeit kein erneuter Variablenzugriff erfolgt. Auf diese Weise können bei Zugriffen auf Variablen in sehr großen Zeitabständen die Anzahl der benötigten Verbindungen reduziert werden. Datentyp UInt32, nur lesbar. 0:kein Abbau >0:Leerlaufzeit bis zum Abbau in ms
optimizes7read	Optimierung von s7 Lesezugriffen auf Bausteine. Datentyp Boolean, Lesen und Schreiben. True: Optimierung False: Keine Optimierung Optimierung bedeutet: Mehrere Zugriffsaufträge auf einzelne Variablen werden intern in einen einzigen Feldzugriff auf den Kommunikationspartner umgewandelt.
optimizes7write	Optimierung von S7-Schreibzugriffen auf Bausteine. Datentyp Boolean, nur lesbar. True: Optimierung False: Keine Optimierung Optimierung bedeutet: Mehrere Zugriffsaufträge auf einzelne Variablen werden intern in einen einzigen Feldzugriff auf den Kommunikationspartner umgewandelt.
defaultalarmseverity	Vorgabe-Priorität für unprojektierte Alarmereignisse. Datentyp UInt16, nur lesbar. 1:niederprior ... 1000:hochprior In der Projektierung gibt es eine Einstellmöglichkeit für Vorgabe-Priorität von Meldungen für S7-Meldungen und S7-Diagnosemeldungen.
defaultdiagnosissSeverity	Vorgabe-Priorität für unprojektierte Diagnoseereignisse. Datentyp UInt16, nur lesbar. 1:niederprior ... 1000:hochprior In der Projektierung gibt es eine Einstellmöglichkeit für Vorgabe-Priorität von Meldungen für S7-Meldungen und S7-Diagnosemeldungen.

S7-Verbindungsdiagnose-Property	Bedeutung																	
events	<p>Anmeldung von Alarmen und Events beim Verbindungspartner.            Datentyp UInt32, nur lesbar.            Die einzelnen Werte können kombiniert werden</p> <table border="1"> <tr><td>0x00000001</td><td>SCAN-Item (nicht mehr unterstützt)</td></tr> <tr><td>0x00000002</td><td>Einfache Meldungen (nicht mehr unterstützt)</td></tr> <tr><td>0x00000004</td><td>Einfache symbolbezogen Meldungen (nicht mehr unterstützt)</td></tr> <tr><td>0x00000008</td><td>Simotion TO-Alarne</td></tr> <tr><td>0x00000010</td><td>Verbindungsüberwachungs-Meldungen</td></tr> <tr><td>0x00000020</td><td>Bausteinbezogene Meldungen (als Conditional Events)</td></tr> <tr><td>0x00000040</td><td>Symbolbezogene Meldungen (als Conditional Events)</td></tr> <tr><td>0x00000080</td><td>Diagnosemeldungen</td></tr> </table>		0x00000001	SCAN-Item (nicht mehr unterstützt)	0x00000002	Einfache Meldungen (nicht mehr unterstützt)	0x00000004	Einfache symbolbezogen Meldungen (nicht mehr unterstützt)	0x00000008	Simotion TO-Alarne	0x00000010	Verbindungsüberwachungs-Meldungen	0x00000020	Bausteinbezogene Meldungen (als Conditional Events)	0x00000040	Symbolbezogene Meldungen (als Conditional Events)	0x00000080	Diagnosemeldungen
0x00000001	SCAN-Item (nicht mehr unterstützt)																	
0x00000002	Einfache Meldungen (nicht mehr unterstützt)																	
0x00000004	Einfache symbolbezogen Meldungen (nicht mehr unterstützt)																	
0x00000008	Simotion TO-Alarne																	
0x00000010	Verbindungsüberwachungs-Meldungen																	
0x00000020	Bausteinbezogene Meldungen (als Conditional Events)																	
0x00000040	Symbolbezogene Meldungen (als Conditional Events)																	
0x00000080	Diagnosemeldungen																	
connectiontype	<p>S7-Verbindungstyp            Datentyp UInt16, nur lesbar.            2:S7D_STD_TYPE; Standardverbindung            3:S7D_H_TYPE; hochverfügbare Verbindung            Ist die S7-Verbindung noch nicht aufgebaut, wird für dieses Item die Qualität "BAD" gemeldet und die Werte sind ungültig.</p>																	
connectionstate	<p>S7-Verbindungszustand            Datentyp UInt16, nur lesbar.            0x11:STD_DOWN; Standardverbindung ist gewollt abgebaut            0x12:STD_ABORT; Standardverbindung wurde ungewollt abgebaut (Fehler)            0x13:STD_NOT_USED; Standardverbindung noch nie aufgebaut            0x14:STD_OK; Standardverbindung aufgebaut            0x20:H_OK_RED; hochverfügbare Verbindung aufgebaut (redundant)            0x21:H_OK_RED_PATH_CHG; hochverfügbare Verbindung aufgebaut (redundant, es wurde umgeschaltet)            0x22:H_OK_NOT_RED; hochverfügbare Verbindung nicht redundant aufgebaut            0x23:H_ABORT; hochverfügbare Verbindung wurde ungewollt abgebaut (Fehler)            0x24:H_NOT_USED; hochverfügbare Verbindung noch nie aufgebaut            0x25:H_DOWN; hochverfügbare Verbindung ist gewollt abgebaut            Ist die S7-Verbindung noch nicht aufgebaut, wird für dieses Item die Qualität "BAD" gemeldet und die Werte sind ungültig.</p>																	
hconnectionwaystate	<p>Zustand der H-Verbindungswege            Datentyp Arrays Of UInt16, 4 Arrayelemente, nur lesbar.            0x30:HW_PROD; Weg ist Produktivverbindung            0x31:HW_STBY; Weg ist Standby-Verbindung            0x32:HW_ABORT; Weg wurde ungewollt abgebaut (Fehler)            0x33:HW_NOT_USED; Weg wurde noch nie aufgebaut            0x34:HW_DOWN; Weg wurde gewollt abgebaut            0x35:HW_CN_BREAK; Weg konnte nicht aufgebaut werden            Ist die S7-Verbindung noch nicht aufgebaut, wird für dieses Item die Qualität "BAD" gemeldet und die Werte sind ungültig.</p>																	

## 2.7.11

### S7-OPC-UA-Template-Datenvariablen

Sie haben mit den Prozessvariablen für das S7-Protokoll unter OPC UA flexible Einstellmöglichkeiten, um die Prozessdaten Ihrer Anlage in den gewünschten Datenformaten zu erhalten.

Die Vielfalt der Adressierungsmöglichkeiten lässt sich allerdings nicht in einen vollständig durchsuchbaren Namensraum fassen. Bereits ein Datenbaustein mit der Länge eines einzelnen Bytes besitzt etwa 40 verschiedene Datenformatoptionen – angefangen vom Byte, SByte, Felder mit einem Element davon, einzelne Bits, Felder von Bits mit bis zu 8 Feldelementen an unterschiedlichen Bitoffsets beginnend.

Der OPC-UA-Server unterstützt den Anwender deshalb mit den sogenannten Template-Datenvariablen im S7-Namensraum. In einem für einen OPC-Client typischen Texteingabefeld können diese Templates durch Ändern einiger weniger Zeichen in gültige ItemIDs verwandelt werden.

Beispiel:

```
S7-Verbindung1.db<db>.<o>, dw
```

Durch Ersetzen von <db> mit der Datenbausteinnummer und <o> dem Offset innerhalb des Datenbausteins erhalten Sie eine gültige Nodeld.

```
-> S7-Verbindung1.db10.4, dw
```

Weiteres Beispiel:

```
S7-Verbindung1.bsend<rid>.<l>,<o>,b,<c>
```

```
-> S7-Verbindung1.bsend43.1000,0,b,100
```

Der Vorteil dieses Konzepts ist, dass es von nahezu allen OPC-UA-Clients eingesetzt werden kann, ohne dass Anpassungen der Clients erforderlich sind.

#### Hinweis

Die Verwendbarkeit von S7-OPC-UA-Template-Datenvariablen kann im Konfigurationsprogramm "Kommunikations-Einstellungen" unter "OPC-Protokollauswahl" > Klicken des Pfeilsymbols bei "S7" aktiviert und deaktiviert werden.

### Template-Datenvariablen innerhalb der Browse-Hierarchie

Die Template-Datenvariablen sind neben den ihnen entsprechenden Ordnern in der Namensraum-Darstellung eingesortiert, so dass sie bei Bedarf leicht genutzt werden können.

### Spezielle Nutzung einiger Attribute der Template-Datenvariablen

Die Verwendung der OPC-UA-Attribute ist durch die UA-Spezifikation vorgegeben und bedarf keiner weiteren Erläuterung.

Beispiel für das Template einer Datenbaustein-Bytevariablen:

Nodeld: S7-Verbindung1.db<db>.<o>,d  
 Browse-Name: Template Byte  
 Beschreibung: <db>Adresse der Datenbausteinnummer  
 <o> offset innerhalb der Datei

## 2.7.12 Events, Conditions und Alarme

### 2.7.12.1 Welche Alarme gibt es?

Dieser Abschnitt beschreibt die Abbildung von S7-Meldungen und S7-Diagnoseereignisse auf OPC-UA-Events, Conditions und Alarme.

Folgende Event- und Alartypen gibt es für OPC UA und S7:

- Statepath-Alarm  
Meldungen zum S7-Verbindungszustand
- Symbolbezogene Meldungen (SCANs)  
Ermöglichen asynchron zum SPS-Anwenderprogramm die Überwachung von Bits in den Bereichen E, A, M und DB der CPU.
- Bausteinbezogene Meldungen (Alarm-SFB, Alarm-SFC)  
Quittierbare Alarm-SFBs sind: ALARM (SFB 33), ALARM\_8 (SFB 34) und ALARM\_8P (SFB 35)  
Nicht quittierbar sind die SFBs: NOTIFY (SFB 36) und NOTIFY\_8P (SFB 31)  
Quittierbare Alarm-SFCs sind: ALARM\_SQ (SFC 17) und ALARM\_DQ (SFC 107)  
Nicht quittierbar sind die SFCs: ALARM\_S (SFC 18) und ALARM\_D (SFC 108)
- Diagnosemeldungen  
Systemdiagnose (ID 0x1000-0x79FF, 0xC000-0xEFFF, 0xF900-0xF9FF)  
Anwenderdiagnose (ID 0x8000-0xB9FF) mit WR\_USMSG (SFC 52)

### 2.7.12.2 Was sind Events?

Eine Anlage ist charakterisiert durch die Stati seiner Hardware- und Software-Komponenten. UA-Alarming bietet die Möglichkeit, Statusänderungen einer Auswahl aller Stati dem dafür angemeldeten Anwender als Ereignisse zu melden. Die Informationen des Events sind in seinen Properties abgelegt. Welche Properties ein Event aufweist, wird durch den Eventtyp definiert.

Der Eventtyp weist eigene oder von einem anderem Eventtyp (der seinerseits vererbte Properties aufweisen kann) vererbten Properties auf. Die Möglichkeit der einfachen Vererbung führt zu einer Eventtyp hierarchie. Die UA-Alarming-Spezifikation weist eine Vielzahl von vordefinierten Eventtypen auf, die in der vordefinierten Typ hierarchie strukturiert sind. Des Weiteren macht die UA-Alarming-Spezifikation Angaben zum Typ der Properties und zu der Semantik dieser Properties. Alle vordefinierten Events befinden sich in Namespace ns="http://opcfoundation.org/UA/".

### 2.7.12.3 Welche Events des S7-UA-Alarming-Servers gibt es?

Der S7-UA-Alarming-Server lehnt sich an vordefinierten Eventtypen an und leitet die eigenen Eventtypen von den vordefinierten Eventtypen ab. Es sind für S7-Events eigene Eventtypen definiert. Alle S7-Eventtypen befinden sich im Namespace ns="S7TYPES:".

S7-UA-Alarming dient zur Darstellung von S7-Meldungen. Die untenstehende Tabelle gibt an, welche Eventtypen ein S7-UA-Alarming-Server meldet. Mit der Ausnahme des ersten Eventtypen, treten Events mit dem angegebenen Eventtyp erst nach dem Empfang einer abgebildeten S7-Meldung auf.

Tabelle 2- 4 Eventtypen von S7-UA-Alarming und ihre Verwendung

Nodeld des S7-Eventtypen	Anzeigename	Bildet bei OPC-Server 7.0 in der S7-Anlage folgendes ab:	Bildet bei OPC-Server 8.0 in der S7-Anlage folgendes ab:
ns=S7Types, i=14	"S7StatepathAlarmType"	1)	1)
ns=S7Types, i=40	"S7ExclusiveLimitAlarmType"	-	Alle projektierten Baustein- und symbolbezogene Meldungen mit Meldeklasse: "Alarm - oben", "Alarm unten", "Warnung - oben" oder "Warnung - unten".
ns=S7Types, i=41	"S7ExclusiveDeviationAlarmType"	-	Alle projektierten Baustein- und symbolbezogene Meldungen mit Meldeklasse: "Toleranz - oben" oder "Toleranz unten".
Ns=S7Types, i=43	"OffNormalAlarmType"	Baustein- und symbolbezogene Meldungen	<ol style="list-style-type: none"> <li>1. Alle unprojektierten Baustein- und symbolbezogene Meldungen und</li> <li>2. projektierte Meldungen mit anderen Meldeklassen als "Toleranz - oben", "Toleranz - unten", "Alarm - oben", "Alarm - unten", "Warnung - oben" und "Warnung - unten").</li> </ol>
Ns=S7Types, i=60	"S7DiagnosisEventType"	Diagnosemeldungen	Projektierte oder unprojektierte Diagnosemeldungen

Zu 1): Der Eventtyp ns="S7TYPES:", i=14 mit dem Anzeigennamen "S7StatepathAlarmType" bildet den inversen Zustand einer S7-Verbindung ab ("Inaktiv", wenn S7-Verbindung aufgebaut ("UP") ist; und "Aktiv", wenn S7-Verbindung nicht aufgebaut ist ("Down")). Der Zustand wird nur PC-seitig ermittelt und kann hierdurch auch dann gemeldet werden, wenn eine physikalische Verbindung zum S7-Gerät nicht besteht.

### 2.7.12.4 Eventtypenhierarchie von S7-UA-Alarming-Server

Die Eventtypenhierarchie des S7-UA-Alarming-Servers besteht aus der Standard-Eventtypenhierarchie, wobei aus manchen Eventtypen S7-Eventtypen abgeleitet werden.

Die Eventtyp hierarchie kann mit dem OPC Scout V10 durchsucht werden. Beim Durchsuchen werden jedoch nicht die Nodelds der Elemente angezeigt, sondern ihre Anzeigenamen.

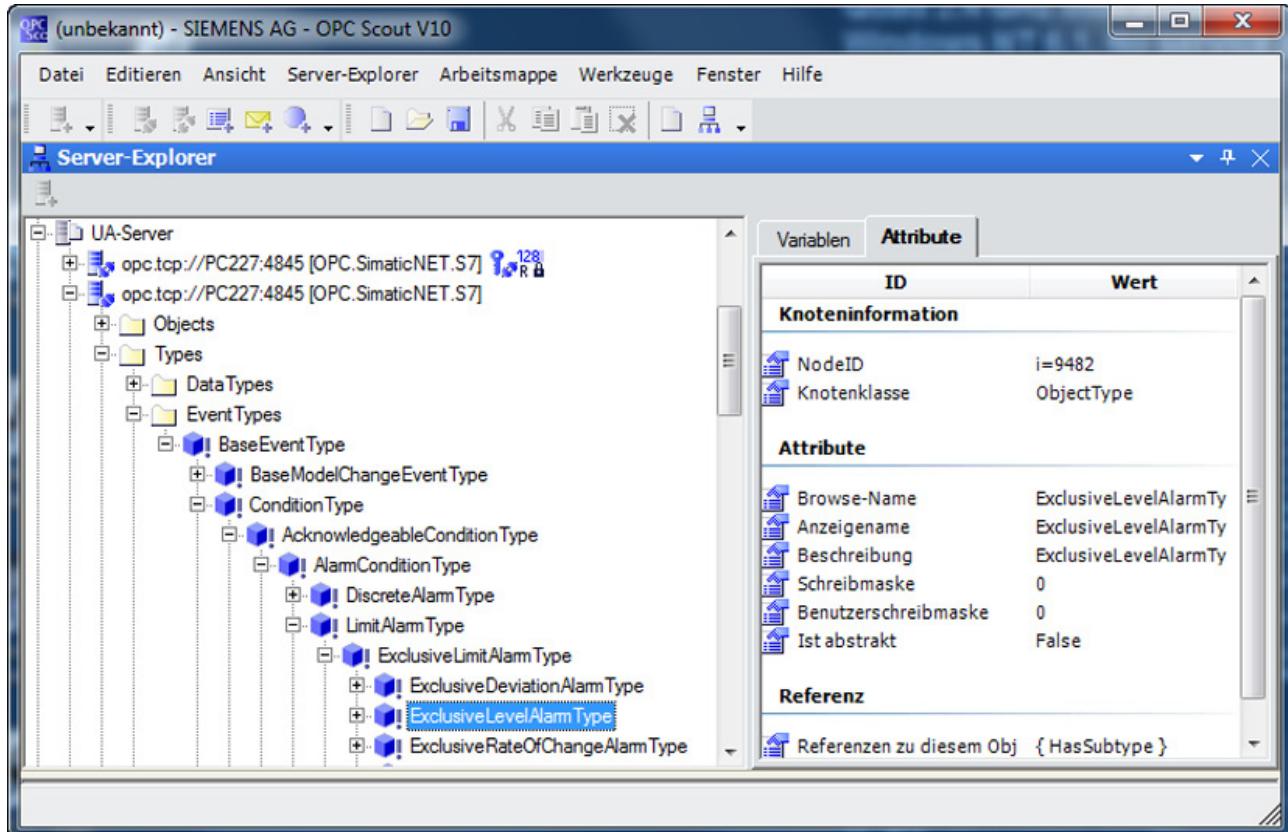


Bild 2-30 Abbildung der Eventtyp hierarchie

Die Vererbung der Eventtypen ist wie folgt:

Eventtyp mit Anzeigenamen	erbt von ...
"ExclusiveLevelAlarmType"	"ExclusiveLimitAlarmType"
"ExclusiveLimitAlarmType"	"LimitAlarmType"
"LimitAlarmType"	"AlarmConditionType"
"AlarmConditionType"	"AcknowledgeableConditionType"
"AcknowledgeableConditionType"	"ConditionType"

Der Eventtyp mit dem Anzeigenamen "BaseEventType" ist der Typ von dem alle Eventtypen abgeleitet sind. Dieser Typ definiert alle Properties, die in allen Events verwendet werden sowie auch deren Verhalten.

Ein Eventtyp ist eine numerische Nodeld (z. B. ns="http://opcfoundation.org/UA/", i=2041 und Anzeigename ="BaseEventType").

Beim S7 UA-Alarming werden Eventtypen, die vom ConditionType abgeleitet sind, instanziert. Ein S7-Alarm ist deshalb mit all seinen Properties in einem Alarmobjekt abgebildet: die Condition-Instanz. Die Properties der Condition-Instanz können hilfsweise auch über Data Access-Zugriffe gelesen oder beobachtet werden.

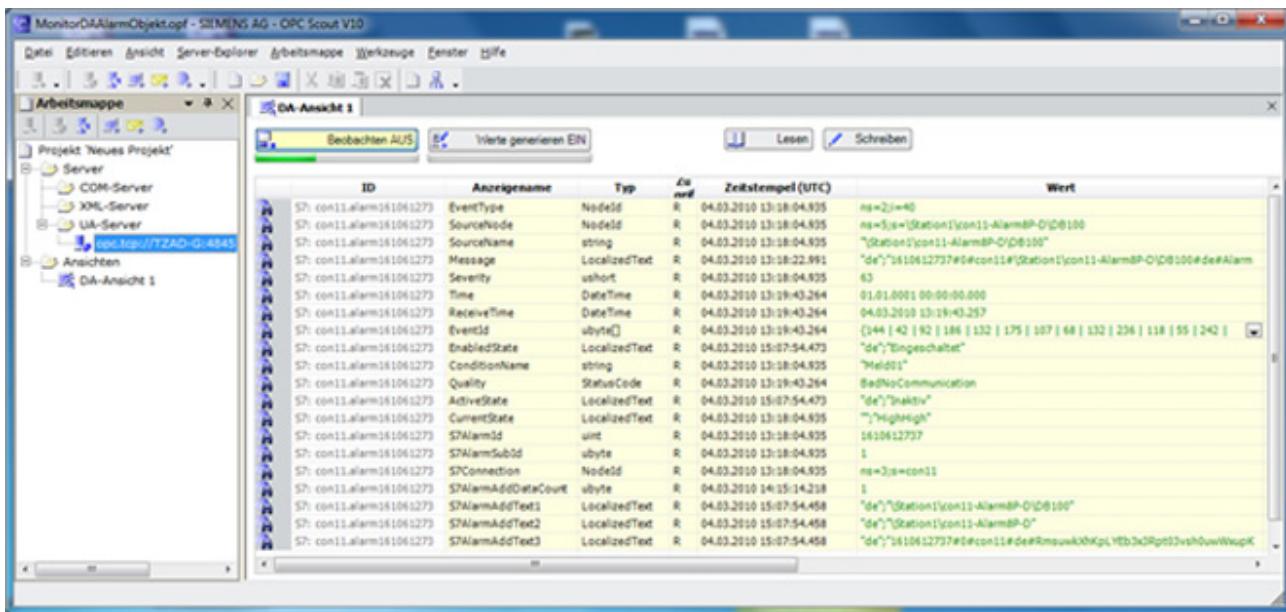


Bild 2-31 Beobachten der Alarmobjekte über Data Access

In den Folgenden Abschnitten werden die für den Anwender bei der Programmierung einer UA-Applikation relevanten Properties angegeben. Sie sind gruppiert nach den Eventtypen, in dem sie definiert werden. Bei der Referenzierung einer Property verwendet man den Datentyp "Qualified Name". "Qualified Name" beinhaltet einen Namespace und einen Browse-Namen (gelegentlich Browse-Pfad).

Da jedoch im Kontext des Dokumentes nur Properties referenziert werden, deren "Qualified Name" einen Namespace haben, der identisch mit dem Namespace des definierenden Typen ist, wird auf die Angabe des Namespace verzichtet. Statt "Qualified Name" wird der Browse-Name des Properties angegeben. Zusätzlich wird in Klammern das angeforderte Attribut getrennt durch " | " von dem Datentyp angegeben. In den meisten Fällen ist das angeforderte Attribut numerisch 13, was den "Value" angibt. Selten tritt der Wert numerisch 1 auf, was die Nodeld angibt.

## 2.7.13 Standard-Eventtypen

### 2.7.13.1 Standard-Eventtyp mit dem Anzeigenamen "BaseEventType"

NodId: ns="http://opcfoundation.org/UA/", i=2041

Abgeleitet von: ist von keinem anderen EventType abgeleitet.

Browse-Namen der relevanten Properties

- "EventType" (13|NodId)
- "SourceNode" (13|NodId)
- "SourceName" (13|String)
- "Message" (13|LocalizedText)
- "Severity" (13|UInt16)
- "Time" (13|DateTime)
- "ReceiveTime" (13|DateTime)
- "EventId" (13|ByteString)

Direkt von diesem Typen abgeleitete S7-UA-Eventtypen: ns="S7TYPES:", i =60, Anzeigenamen="S7DiagnosisEventType". Er hat keine Condition-Instanz. Ein Event von diesem Eventtyp wird eindeutig durch den Source-Namen identifiziert.

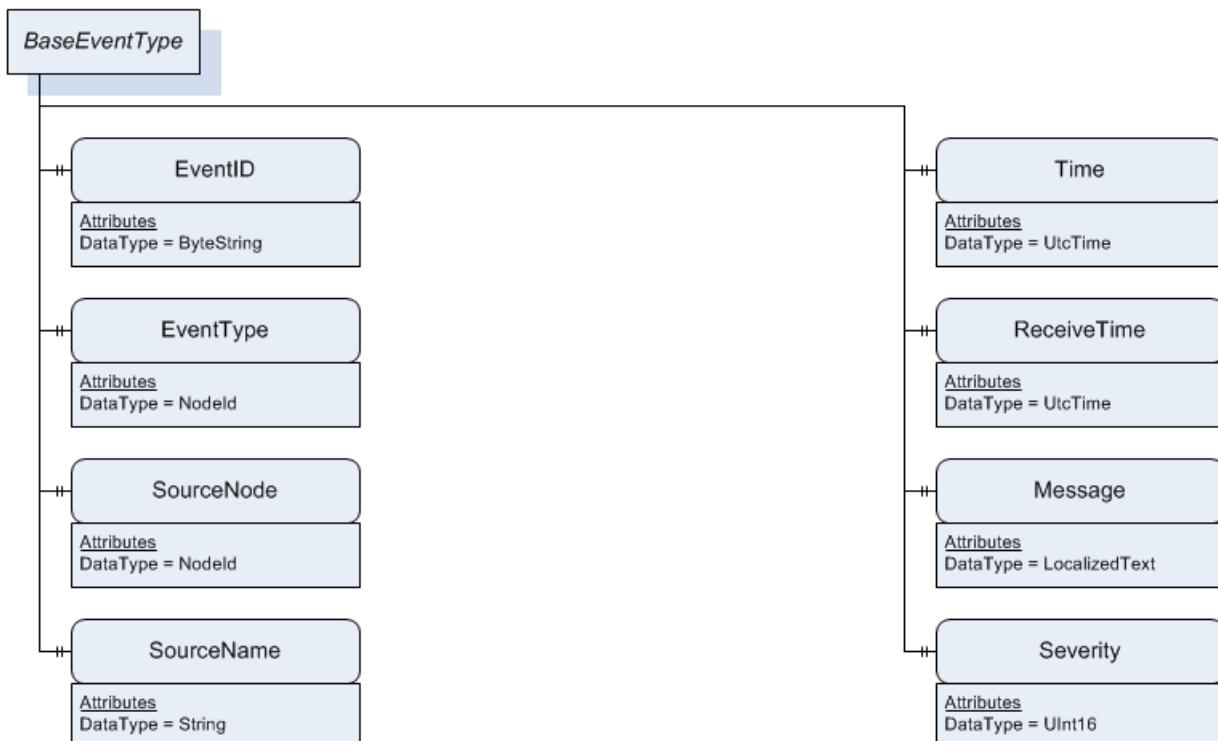


Bild 2-32 Beispielhafte Darstellung des BaseEventType

## EventType

ist immer einer der bereits aufgelisteten S7-Eventtypen.

**SourceNode**

Herkunftsnode eines Ereignisses, bei S7 UA-Alarming ein durch eine NodId spezifiziertes Objekt im ns="S7SOURCES:" oder das S7-Verbindungsobjekt im ns="S7:". Siehe auch im Kapitel "Bereichsbaum und Herkunftsraum (Seite 224)".

**SourceName**

nicht lokalisierter Name der Herkunft. Siehe folgende Tabelle "Bildung des Source-Namen".

**Message**

Siehe folgende Tabelle "Bildung der Meldung".

**Severity**

Priorität siehe folgende Tabelle "Bildung der Severity".

**Time**

Zeitstempel, der so nahe wie möglich am Prozess ist. Dies wird durch die Projektierung bestimmt, wobei es folgende Möglichkeiten gibt:

- CPU-Zeitstempel,
- CPU-Zeit + Offset,
- PC Zeit (UTC).

Bei PC-Zeit (UTC) ist "Time" identisch mit "ReceiveTime".

**ReceiveTime**

Zeitstempel des PC.

**EventId**

Eine Kennung (opaque Id), welche ein Event eindeutig bestimmt und referenziert. Der Client benötigt die EventId z.B. zur Quittierung von Alarmen.

**Bildung des Source-Namen**

Nodeld des S7-Eventtypen	Anzeigename	Bildungsregel bei OPC-Server 7.0:	Bildungsregel bei OPC-Server 8.0:
ns=S7TYPES:, i=14	"S7StatepathAlarmType"	S7-Verbindungsname + ".statepath". Beispiel con13.statepath	S7-Verbindungsname + ".statepath" Beispiel: con13.statepath. Es existiert ein entsprechender SourceNode.
ns= S7TYPES:, i=40	"S7ExclusiveLimitAlarmType"	S7-Verbindungsname Beispiel: con13	<ol style="list-style-type: none"> <li>1. Unprojektierte S7-Meldungen: S7-Verbindugsname Beispiel: con13</li> <li>2. Projektierte Meldungen: Zusatztext 1 Es existiert ein entsprechender SourceNode.</li> <li>3. Projektierte Meldungen ohne Zusatztext1: Anlagepfad + "\" + Baustein (bzw. Symbolnamen bei SCAN) . Beispiel: Station1\con13-Scan-Notify\S7-Programm(3)\DB604. Es existiert ein entsprechender SourceNode.</li> </ol>
ns= S7TYPES:, i=41	"S7ExclusiveDeviationAlarmType"	S7-Verbindungsname Beispiel: con13	<ol style="list-style-type: none"> <li>1. Unprojektierte S7-Meldungen: S7-Verbindugsname Beispiel: con13</li> <li>2. Projektierte Meldungen: Zusatztext 1. Es existiert ein entsprechender Source Node.</li> <li>3. Projektierte Meldungen ohne Zusatztext1: Anlagepfad + "\" + Baustein (bzw. Symbolnamen bei SCAN) . Beispiel: Station1\con13-Scan-Notify\S7-Programm(3)\DB604. Es existiert ein entsprechender SourceNode.</li> </ol>

Nodeld des S7-Eventtypen	Anzeigename	Bildungsregel bei OPC-Server 7.0:	Bildungsregel bei OPC-Server 8.0:
ns= S7TYPES:, i=43	"OffNormalAlarmType"	S7-Verbindungsname Beispiel: con13	<ol style="list-style-type: none"> <li>1. Unprojektierte S7-Meldungen: S7-Verbindungsname Beispiel: con13</li> <li>2. Projektierte Meldungen: Zusatztex 1. Es existiert ein entsprechender SourceNode.</li> <li>3. Projektierte Meldungen ohne Zusatztex 1: Anlagepfad + "\" + Baustein (bzw. Symbolnamen bei SCAN) . Beispiel: Station1\con13-Scan-Notify\S7-Programm(3)\DB604. Es existiert ein entsprechender SourceNode.</li> </ol>
ns= S7TYPES:, i=60	"S7DiagnosisEventType"	S7-Verbindungsname + ".diagnosis" + Meldungsnummer als Hex-Wert Beispiel: Con13.diagnosis0xA001	<ol style="list-style-type: none"> <li>1. Unprojektierte Meldungen: S7-Verbindungsname + ".diagnosis" + Meldungsnummer als Hex-Wert Beispiel: con13.diagnosis0xA001</li> <li>2. Projektierte Meldungen: Anlagepfad + "\" + Meldebezeichner Beispiel: Station1\con13-Scan-Notify\S7-Programm(3)\DB604</li> <li>3. Projektierte Meldungen: Anlagepfad + "\" + Meldebezeichner. Beispiel: Station1\con13-Scan-Notify\S7-Programm(3)\WR_USMSG (1). Es existiert ein entsprechender SourceNode.</li> </ol>

## Bildung der Meldung

Nodeld des S7-Eventtypen	Anzeigename	Bildungsregel bei OPC-Server 7.0:	Bildungsregel bei OPC-Server 8.0:
ns= S7TYPES:, i=14	"S7StatepathAlarmType"	"statepath"	"statepath"
ns= S7TYPES:, i=40	"S7ExclusiveLimitAlarmType"	Verbindungsname + ".alarm" + Meldenummer Beispiel: con13.alarm1	<ol style="list-style-type: none"> <li>1. Unprojektierte S7-Meldungen: S7-Verbindungsname + ".alarm" + Meldenummer Beispiel: con13.alarm1</li> <li>2. Projektierte Meldungen: Meldetext</li> </ol>

Nodeld des S7-Eventtypen	Anzeigename	Bildungsregel bei OPC-Server 7.0:	Bildungsregel bei OPC-Server 8.0:
ns= S7TYPES; i=41	"S7ExclusiveDeviationAlarmType"	Verbindungsname + ".alarm" + Meldenummer Beispiel: con13.alarm1	<ol style="list-style-type: none"> <li>Unprojektierte S7-Meldungen: S7-Verbindungsname + ".alarm"+ Meldenummer Beispiel: con13.alarm1</li> <li>Projektionierte Meldungen: Meldetext</li> </ol>
ns= S7TYPES; i=43	"OffNormalAlarmType"	Verbindungsname + ".alarm" + Meldenummer Beispiel: con13.alarm1	<ol style="list-style-type: none"> <li>Unprojektierte S7-Meldungen: S7-Verbindungsname + ".alarm"+ Meldenummer Beispiel: con13.alarm1</li> <li>Projektionierte Meldungen: Meldetext</li> </ol>
ns= S7TYPES; i=60	"S7DiagnosisEventType"	S7-Verbindungsname + ".diagnosis" + Hex-Wert. Beispiel: con13.diagnosis0xA001	<ol style="list-style-type: none"> <li>Unprojektierte Meldungen: S7-Verbindungsname + ".diagnosis" + Hex-Wert. Beispiel: con13.diagnosis0xA001</li> <li>Projektionierte Meldungen: Meldetext kommend und Meldetext gehend</li> </ol>

### Bildung der Severity

Nodeld des S7-Eventtypen	Anzeigename	Bildungsregel bei OPC-Server 7.0:	Bildungsregel bei OPC-Server 8.0:
ns= S7TYPES; i=14	"S7StatepathAlarmType"	Für jede S7-Verbindung Vorgabe-Priorität für Meldungen.	Für jede S7-Verbindung Vorgabe-Priorität für Meldungen.
ns= S7TYPES; i=40	"S7ExclusiveLimitAlarmType"	-	<ol style="list-style-type: none"> <li>Für jede S7-Verbindung Meldungsweise erfasste Priorität falls vorhanden oder sie wird aus der S7-Meldungspriorität (0 bis 16) der Bausteine und Symbole abgeleitet.</li> </ol>
ns= S7TYPES; i=41	"S7ExclusiveDeviationAlarmType"	-	<ol style="list-style-type: none"> <li>Für jede S7-Verbindung Meldungsweise erfasste Priorität falls vorhanden oder sie wird aus der S7-Meldungspriorität (0 bis 16) der Bausteine und Symbole abgeleitet.</li> </ol>

Nodeld des S7-Eventtypen	Anzeigename	Bildungsregel bei OPC-Server 7.0:	Bildungsregel bei OPC-Server 8.0:
ns= S7TYPES:, i=43	"OffNormalAlarmType"	Für jede S7-Verbindung Vorgabe-Priorität für Meldungen. Meldungnummerweise erfasste Priorität, falls vorhanden.	<ol style="list-style-type: none"> <li>1. Unprojektierte S7-Meldungen: Für jede S7-Verbindung Vorgabe-Priorität für Meldungen oder Meldungnummerweise erfasste Priorität.</li> <li>2. Projektierte Meldungen: Für jede S7-Verbindung erfasste Priorität falls vorhanden oder sie wird aus der S7-Meldungspriorität (0 bis 16) der Bausteine und Symbole abgeleitet.</li> </ol>
ns= S7TYPES:, i=60	"S7DiagnosisEvent Type"	Für jede S7-Verbindung Vorgabe-Priorität für Meldungen oder Meldungnummerweise erfasste Priorität.	<ol style="list-style-type: none"> <li>1. Für jede S7-Verbindung Vorgabe-Priorität für Meldungen oder</li> <li>2. Meldungnummerweise erfasste Priorität, falls vorhanden.</li> </ol>

Tabelle 2- 5 Umformungstabelle S7-Meldungspriorität Severity

S7-Meldungspriorität	Severity
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
13	813
14	875
15	938
16	1000

### 2.7.13.2 Standard-Eventtyp mit dem Anzeigenamen "ConditionType"

Nodeld: ns="http://opcfoundation.org/UA/", i=2782  
 Abgeleitet von: ns="http://opcfoundation.org/UA", i=2041 mit Anzeigenamen "BaseEventType"

Browse-Namen der relevanten Properties

- "" (1|Nodeld)
- "ConditionName" (13|String)
- "EnableState|ID" (13|Boolean)(Browse-Path)
- "EnableState" (13|LocalizedText)
- "Quality" (13|StatusCode)

Direkt von diesem Typen abgeleitete S7-Eventtypen: keine

Direkt oder indirekt davon abgeleitete Eventtypen haben eine Condition-Instanz.

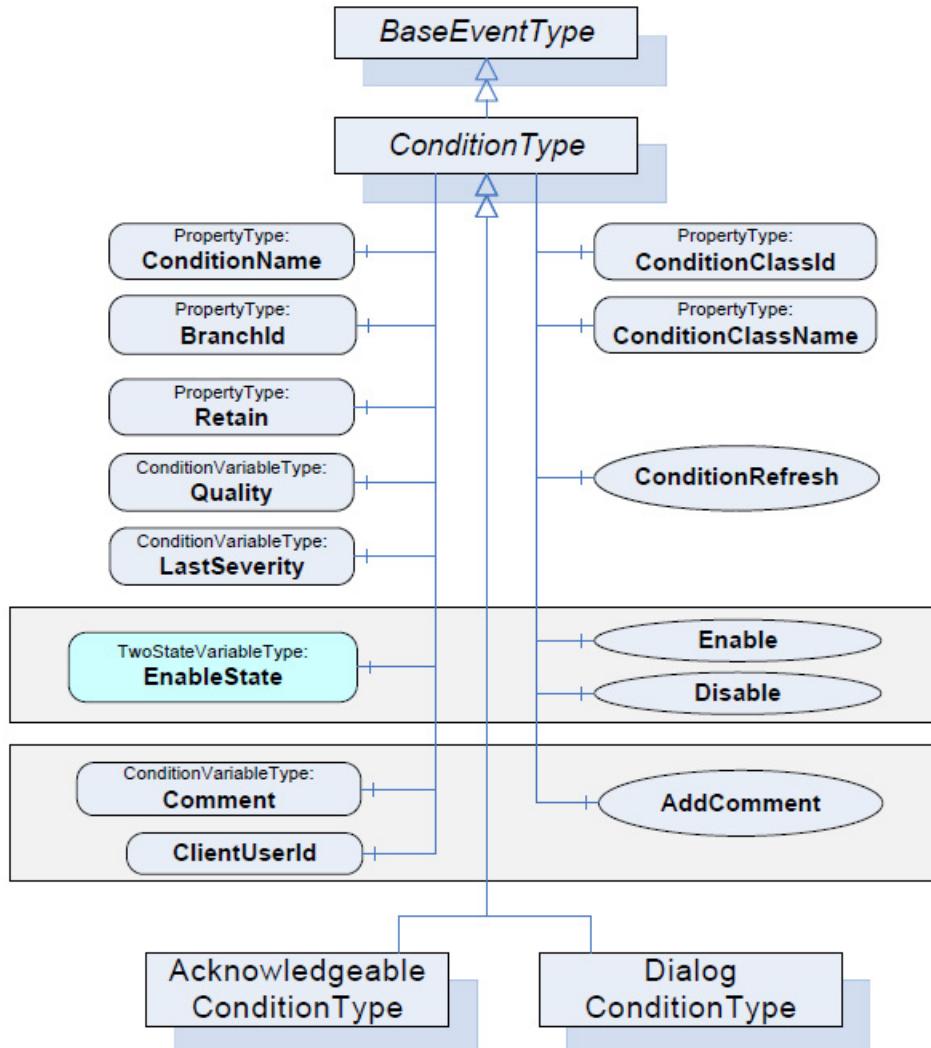


Bild 2-33 Beispielhafte Darstellung des ConditionType

## ConditionName

Ein Event einer Condition wird durch die Kombination "SourceName" und "ConditionName" eindeutig identifiziert. Siehe unten, Abschnitt "Bildung des Condition Namen".

## EnableState

Ist Teil einer Zustandsmaschine. Mögliche Werte gemäß UA-Alarming-Spezifikation für EnableState (LocalizedText) sind "de";"Eingeschaltet" und "de";"Ausgeschaltet". S7 UA Alarming unterstützt das Ausschalten nicht, daher erscheint immer "de";"Eingeschaltet".

## Quality

Die möglichen Quality-Werte siehe UA-Spezifikation.

## Bildung des Condition-Namen

Nodeld des S7-Eventtypen	Anzeigename	Bildungsregel bei OPC-Server 7.0:	Bildungsregel bei OPC-Server 8.0:
ns=S7TYPES:, i=14	"S7StatepathAlarmType"	"statepath"	"statepath"
ns= S7TYPES:, i=40	"S7ExclusiveLimitAlarmType"	"alarm" + Meldungsnummer. Beispiel: alarm1	<ol style="list-style-type: none"> <li>1. Unprojektierte S7-Meldungen: "alarm" + Meldungsnummer Beispiel: alarm1</li> <li>2. Projektierte Meldungen:  Bausteinbezogene Meldungen: Name der mit dem Attribut S7_a_type gekennzeichneten Eingabeveriablen des generierenden Funktionsbausteins. Bei den Signalen 2 ... 8 gefolgt von "," und der Signalnummer. Beispiel: Meld01, oder Meld01,  Symbolbezogene Meldungen: Name des Symbols Beispiel: Scan01</li> </ol>

Nodeld des S7-Eventtypen	Anzeigename	Bildungsregel bei OPC-Server 7.0:	Bildungsregel bei OPC-Server 8.0:
ns= S7TYPES; i=41	"S7ExclusiveDeviationAlarmType"	"alarm" + Meldungsnummer. Beispiel: alarm1	<p>1. Unprojektierte S7-Meldungen: "alarm" + Meldungsnummer Beispiel: alarm1</p> <p>2. Projektierte Meldungen: Bausteinbezogene Meldungen: Name der mit dem Attribut S7_a_type gekennzeichneten Eingabevariablen des generierenden Funktionsbausteins. Bei den Signalen 2 ... 8 gefolgt von "," und der Signalnummer. Beispiel: Meld01, oder Meld01,</p> <p>Symbolbezogene Meldungen: Name des Symbols Beispiel: Scan01</p>
ns= S7TYPES; i=43	"OffNormalAlarmType"	"alarm" + Meldungsnummer. Beispiel: alarm1	<p>1. Unprojektierte S7-Meldungen: "alarm" + Meldungsnummer Beispiel: alarm1</p> <p>2. Projektierte Meldungen: Bausteinbezogene Meldungen: Name der mit dem Attribut S7_a_type gekennzeichneten Eingabevariablen des generierenden Funktionsbausteins. Bei den Signalen 2 ... 8 gefolgt von "," und der Signalnummer. Beispiel: Meld01, oder Meld01,</p> <p>Symbolbezogene Meldungen: Name des Symbols Beispiel: Scan01</p>

### 2.7.13.3 Standard Eventtyp mit dem Anzeigenamen "AcknowledgeableConditionType"

Nodeld: ns:"http://opcfoundation.org/UA/", i=2881

Abgeleitet von: ns="http://opcfoundation.org/UA", i=2782 mit Anzeigenamen "ConditionType"

Browse-Namen der relevanten Properties

"AckedState|Id" (13|Boolean)

"AckedState" (13|LocalizedText)

Direkt von diesem Typen abgeleitete S7-Eventtypen: keine

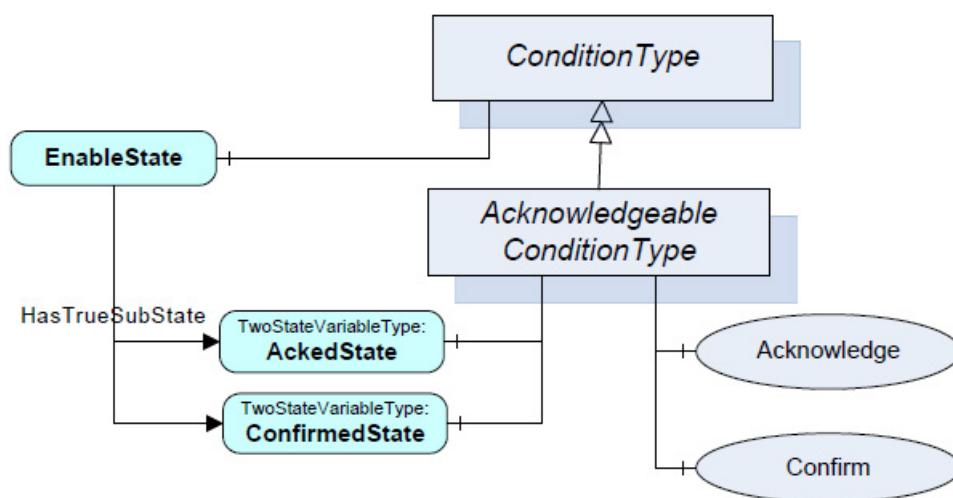


Bild 2-34 Beispielhafte Darstellung des AcknowledgeableConditionType

#### AckedState

Ist ein Teil einer Zustandsmaschine. Mögliche Werte sind "de";"Quittiert" und "de";"Unquittiert". Jede Werteänderung von AckedState wird mit einem Event gemeldet. Events, die mit dem AckedState "de", "Unquittiert" gemeldet werden, müssen mit der Acknowledge-Funktion quittiert werden. Siehe STEP 7-Beschreibung der Bausteinalarme, die eine Quittierung erfordern.

### 2.7.13.4 Standard-Eventtyp mit dem Anzeigenamen "AlarmConditionType"

Nodeld: ns:"http://opcfoundation.org/UA/", i=2915

Abgeleitet von: ns="http://opcfoundation.org/UA", i=2881 mit Anzeigenamen

"AcknowledgeableConditionType"

Browse-Namen der relevanten Properties

"ActiveState|Id" (13|Boolean)

"ActiveState" (13|LocalizedText)

Direkt von diesem Typen abgeleitete S7-Eventtypen: keine

Mögliche Werte gemäß UA-Alarming-Spezifikation für ActiveState (LocalizedText) sind "de";"Aktiv" und "de";"Inaktiv".

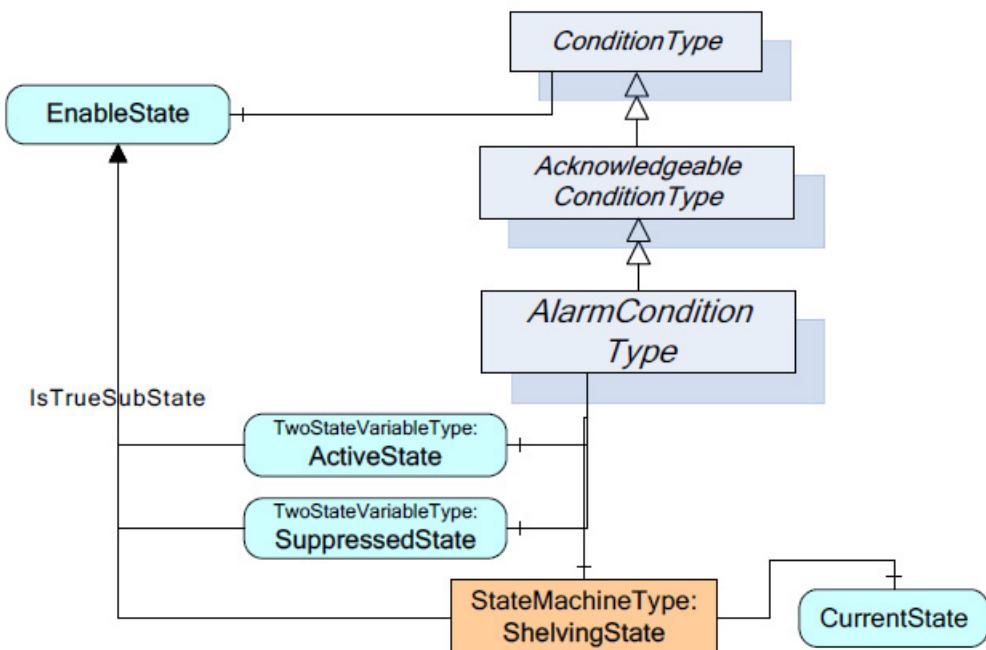


Bild 2-35 Beispielhafte Darstellung des AlarmConditionType

## ActiveState

Ist Teil einer Zustandsmaschine. Bei S7 UA Alarming wird der ActiveState durch Baustein- und symbolbezogene Meldungen gesteuert. Wenn das meldungsauslösende Signal den Wert "1" hat, wird AckedState "Aktiv" gemeldet, wenn der Wert "0" hat, dann wird "Inaktiv" gemeldet. Jede Werteänderung von ActiveState wird durch einen Event gemeldet.

## ActiveState bei Eventtyp mit dem Anzeigenamen="S7StatepathAlarmType"

Hier wird der Zustand "Aktiv" erreicht, wenn die S7-Verbindung nicht aufgebaut ist. Der Zustand "Inaktiv" ist erreicht, wenn die S7-Verbindung aufgebaut ist.

### 2.7.13.5 Standard-Eventtyp mit dem Anzeigenamen "ExclusiveLimitAlarmType"

Nodeld: ns="http://opcfoundation.org/UA/", i=9341

Abgeleitet indirekt von: ns="http://opcfoundation.org/UA/", i=2915 mit Anzeigenamen "AlarmConditionType"

Browse-Namen der relevanten Properties:

"LimitState|CurrentState" (13|LocalizedText)

Direkt von diesem Typen abgeleiteten S7-UA-Eventtypen: keine

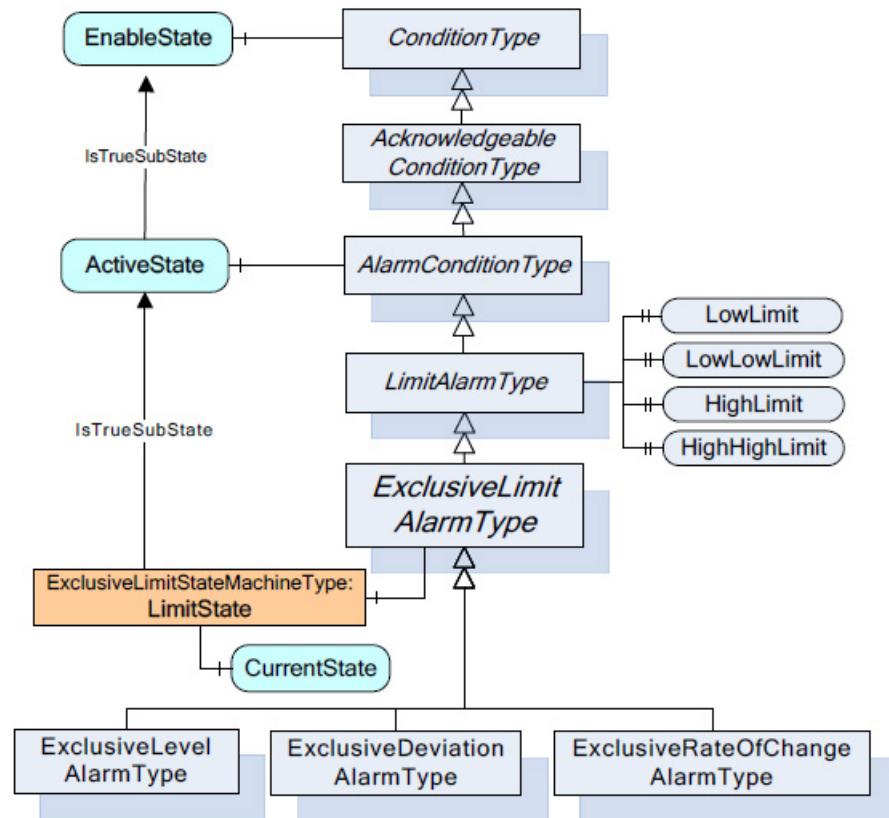


Bild 2-36 Beispielhafte Darstellung des ExclusiveLimitAlarmType

#### LimitState|CurrentState

Mögliche Werte sind "High", "HighHigh", "Low" und "LowLow".

### 2.7.13.6 Standard-Eventtyp mit dem Anzeigenamen "ExclusiveLevelAlarmType"

Nodeld: ns="http://opcfoundation.org/UA/", i=9482

Abgeleitet von: ns="http://opcfoundation.org/UA/", i=9341 Anzeigenamen "ExclusiveLimitAlarm Type"

Browse-Namen der relevanten Properties: keine

Direkt von diesem Typen abgeleiteten S7-UA-Eventtyp:

ns="S7TYPES:", i =40, Anzeigenamen="S7ExclusiveLevelAlarm Type"

Der Eventtyp mit dem Anzeigenamen "S7ExclusiveLevelAlarm Type" wird generiert, wenn ein OPC-Server 8.0 gewählt ist und für die Meldeklasse der Meldungen

"Alarm - oben" (LimitState|CurrentState="HighHigh"),  
 "Alarm – unten" (LimitState|CurrentState="LowLow"),  
 "Warnung - oben" (LimitState|CurrentState="High") und  
 "Warnung - unten" (LimitState|CurrentState="Low") eingestellt ist.

### 2.7.13.7 Standard-Eventtyp mit dem Anzeigenamen "ExclusiveDeviationAlarmType"

NodId: ns:"http://opcfoundation.org/UA/", i=9764

Abgeleitet von: ns="http://opcfoundation.org/UA/", i=9341 Anzeigenamen  
 "ExclusiveLimitAlarm Type"

Browse-Namen der relevanten Properties: keine

Direkt von diesem Typen abgeleiteten S7-UA-Eventtype:

ns=S7Types, i =41, Anzeigenamen="S7ExclusiveDeviationAlarm Type"

Der Eventtype mit dem Anzeigenamen "S7ExclusiveDeviationAlarm Type" wird generiert, wenn ein OPC-Server 8.0 gewählt ist und für die Meldeklasse der Meldungen "Toleranz - oben" oder "Toleranz - unten" eingestellt ist.

### 2.7.13.8 Standard-Eventtyp mit dem Anzeigenamen "OffNormalAlarmType"

NodId: ns:"http://opcfoundation.org/UA/", i=10637

Abgeleitet indirekt von: ns="http://opcfoundation.org/UA/", i=2915 mit Anzeigenamen  
 "AlarmCondition Type"

Browse-Namen der relevanten Properties: keine

Direkt von diesem Typen abgeleiteten S7-UA-Eventtypen:

ns=S7TYPES;, i = 43, Anzeigename="S7OffNormalAlarm Type"

ns=S7TYPES;, i =14, Anzeigenamen="S7StatepathAlarm Type"

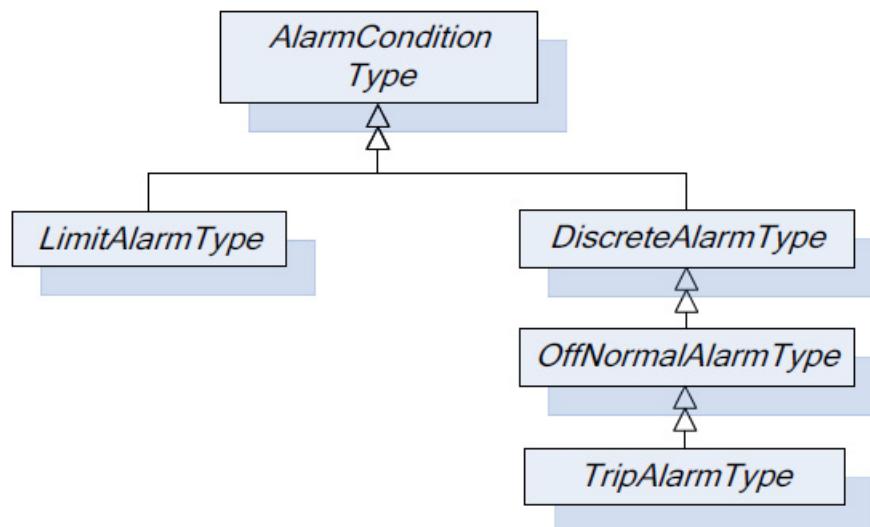


Bild 2-37 Beispielhafte Darstellung des OffNormalAlarm Type

Der Eventtype mit dem Anzeigenamen "S7OffNormalAlarm Type" wird generiert, wenn ein OPC-Server 7.0 gewählt ist oder OPC-Server 8.0 und für die Meldungen andere Meldeklassen als "Toleranz - oben", "Toleranz - unten", "Alarm - oben", "Alarm - unten", "Warnung - oben" und "Warnung - unten" eingestellt sind.

## 2.7.14 S7-Eventtypen

### 2.7.14.1 S7-Eventtyp mit dem Anzeigenamen "S7StatepathAlarmType"

NodeID: ns="S7TYPES:", i=14

Abgeleitet indirekt von: ns="http://opcfoundation.org/UA/", i=2915 mit Anzeigenamen "AlarmCondition Type"

Browse-Namen der relevanten Properties:  
"S7Connection" (13|NodeID)

Dieser Eventtyp bildet den inversen Zustand einer S7-Verbindung ab ("Inaktiv", wenn S7-Verbindung aufgebaut ("UP") ist; und "Aktiv", wenn S7-Verbindung nicht aufgebaut ist ("Down")). Der Zustand wird nur PC-seitig ermittelt und kann hierdurch auch dann gemeldet werden, wenn eine physikalische Verbindung zum S7-Gerät nicht besteht.

#### S7Connection

Gibt die NodeID der Verbindung an, über die eine S7-Meldung empfangen wird.  
Bsp.: ns="S7:", s="Verbindungsname".

### 2.7.14.2 S7-Eventtyp mit dem Anzeigenamen "S7ExclusiveLevelAlarmType"

Nodeld: ns="S7TYPES:", i=40  
 Abgeleitet indirekt von: ns="http://opcfoundation.org/UA/", i=9482 mit Anzeigenamen "ExclusiveLevelAlarm Type"  
 Browse-Namen der relevanten Properties:  
 "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)

Dieser S7-Eventtyp bildet bei OPC-Server 8.0 alle projektierten Baustein- und symbolbezogene Meldungen mit den Meldeklassen "Alarm - oben", "Alarm unten", "Warnung - oben" oder "Warnung - unten" ab. Bei OPC-Server 7.0 wird dieser S7 Eventtyp nicht verwendet.

#### S7AlarmId

Meldungsnummer der S7-Meldung.

#### S7AlarmSubId

Die Nummer des auslösenden Signals einer S7-Meldung. Der Wertebereich ist 1 ... 8.

**S7Connection**

Gibt die Nodeld der Verbindung an, über die die S7 Meldung empfangen wird.  
Bsp.: ns="S7:", s="Verbindungsname".

**S7AlarmAddData|Data**

Sind die Begleitwerte der generierenden S7-Meldungen.  
Meldungen, dargestellt im ByteString-Format.  $1 \leq n \leq 10$

**S7AlarmAddData8|DataType**

Sind die S7-Datentypen der Begleitwerte. Eine Abbildung der S7-Daten-Typen auf  
UA-Datentypen ist nicht möglich, da sie nicht eindeutig abzubilden sind.

**S7AlarmAddDataCount**

Ist die tatsächliche Anzahl der Begleitwerte der S7-Meldungen.

**S7AlarmAddTextn**

$1 \leq n \leq 8$ . Zusatztexte aus der Projektierung.

### 2.7.14.3 S7-Eventtyp mit dem Anzeigenamen "S7ExclusiveDeviationAlarmType"

Nodeld: ns="S7TYPES:", i=41  
Abgeleitet indirekt von: ns="http://opcfoundation.org/UA/", i=9764 mit Anzeigenamen "ExclusiveDeviationAlarm Type"  
Browse-Namen der relevanten Properties:  
"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)

Dieser S7-Eventtyp bildet bei OPC-Server 8.0 alle projektierten Baustein- und symbolbezogene Meldungen mit den Meldeklassen "Toleranz - oben" oder "Toleranz unten" ab. Siehe Hinweise von "S7ExclusiveLevelAlarm Type".

#### 2.7.14.4 S7-Eventtyp mit dem Anzeigenamen "S7OffNormalAlarmType"

Nodeld: ns="S7TYPES.", i=43

Abgeleitet indirekt von: .ns="http://opcfoundation.org/UA/", i=10637 mit Anzeigenamen "OffNormalAlarm Type"

Browse-Namen der relevanten Properties:

- "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)

Dieser Eventtyp bildet bei OPC-Server 7.0 Baustein- und symbolbezogene Meldungen ab.

Bei OPC-Server 8.0 bildet er alle unprojektierten Baustein- und symbolbezogene Meldungen und projektierte Meldungen mit anderen Meldeklassen als "Toleranz - oben", "Toleranz - unten", "Alarm - oben", "Alarm - unten", "Warnung - oben" und "Warnung - unten") ab. Siehe Hinweise von "S7ExclusiveLevelAlarm Type".

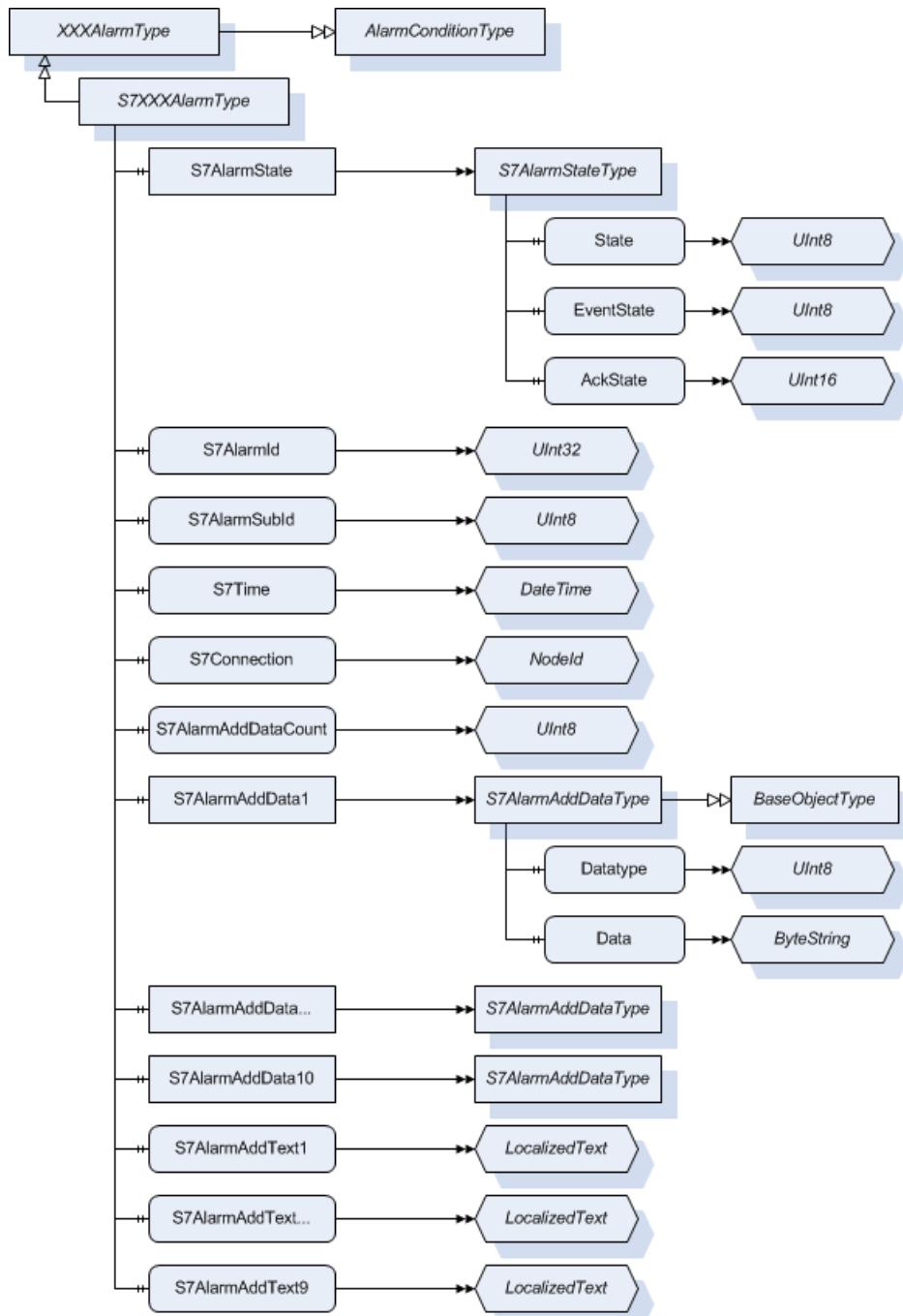


Bild 2-38 Beispielhafte Darstellung der S7-spezifischen Properties für den S7OffNormalAlarm Type und den anderen S7XXXAlarm Types

### 2.7.14.5 S7-Eventtyp mit dem Anzeigenamen "S7DiagnosisEventType"

Nodeld: ns="S7TYPES.", i=60

Abgeleitet indirekt von: .ns="http://opcfoundation.org/UA/", i=2041 mit Anzeigenamen "BaseEvent Type"

Browse-Namen der relevanten Properties:

"S7DiagnosisId" (13|UInt32)

"S7Connection" (13|NodeID)

"S7DiagnosisData" (13|ByteString)

"S7Time" (13|DateTime)

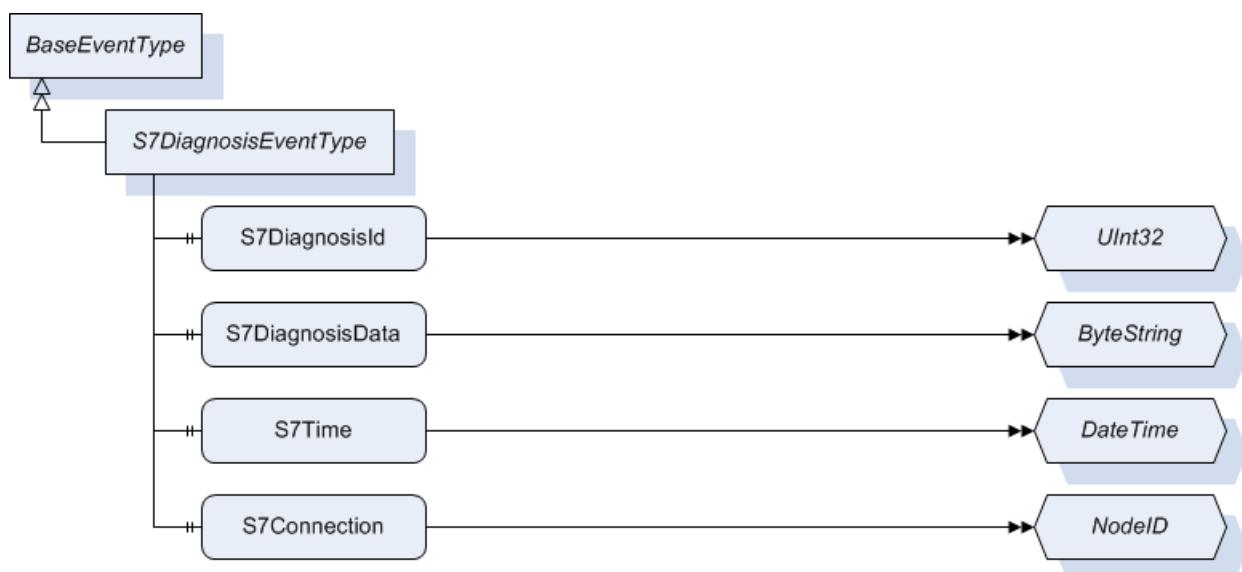


Bild 2-39 Beispielhafte Darstellung des S7DiagnosisEvent Type

#### S7DiagnosisId

Melde-ID der Diagnosemeldung.

#### S7Connection

Gibt die NodeID der Verbindung an, über die die Diagnose Meldung empfangen wird.  
Bsp.: ns="S7:", s="Verbindungsname".

#### S7DiagnosisData

Diagnosedaten

---

#### Hinweis

Eine detaillierte Beschreibung zu SFC 52 (S7DiagnosisData) finden Sie im Dokument "STEP 7 System- und Standardfunktionen für S7-300/400".

## 2.7.15 Bereichsbauum und Herkunftsraum

Bei OPC-Server 7.0 gibt es keinen Bereichsbauum und keinen Herkunftsraum. Bei OPC-Server 8.0 enthalten der Bereichsbauum und der Herkunftsraum die SourceNodes von projektierten S7-Meldungen. Unprojektierte S7-Meldungen haben keine SourceNodes.

S7-UA-Alarming baut den Bereichsbauum auf, um die SourceNodes Bereichen zuzuordnen. Knoten des Bereichsbauums sind spezielle UA-Ordner, die Bereiche der Anlage abbilden. Diese speziellen UA-Ordner werden Bereichsknoten genannt.

Bereiche einer Anlage werden als "Zusatztext 2" aller Baustein- und symbolbezogenen Meldungen definiert. Z. B. definiert der Zusatztext "Maschinenhaus\Kessel" den übergeordneten Bereich "Maschinenhaus" und den untergeordneten Bereich "Kessel" mit den Nodelds der zugehörigen UA-Ordner

- ns="S7AREAS:", s="Maschinenhaus".
- ns="S7AREAS:", s="Maschinenhaus\Kessel".

Der "Zusatztext 1" der Baustein- und symbolbezogenen Meldung generiert einen Herkunfts Node (SourceNode). Z. B. Angenommen der "Zusatztext 1" ist "Überdrucksensor", entspricht diesem Zusatztext der SourceNode mit Nodeld

- ns="S7SOURCES:", s="Überdrucksensor".

Der "Zusatztext 1" kann "\" enthalten, dies führt jedoch nicht zu mehreren SourceNodes.

Die Angabe eines "Zusatztext 1" und "Zusatztext 2" ist nicht erforderlich. Sind diese nicht vorhanden, werden als Ersatz für "Zusatztext 2" der Pfad in STEP 7 im Anlagebaum verwendet und als Ersatz für "Zusatztext 1" der Pfad in STEP 7 im Anlagebaum + Alarminstanzdatenbaustein bzw. Symbolnamen (bei SCAN) verwendet. Z. B. generiert der Alarm im Alarminstanzdatenbaustein DB404 im "S7-Programm(1)" der CPU "CPU 416-3" des AS-Geräts "Station1" die Bereiche "Station1" mit untergeordneten Bereichen "CPU 416-3" und "S7-Programm(1)" mit den Nodelds der zugehörigen UA-Ordner

- ns="S7AREAS:", s="Station"
- ns="S7AREAS:", s="Station1\CPU 416-3"
- ns="S7AREAS:", s="Station1\CPU 416-3\S7-Programm(1)"

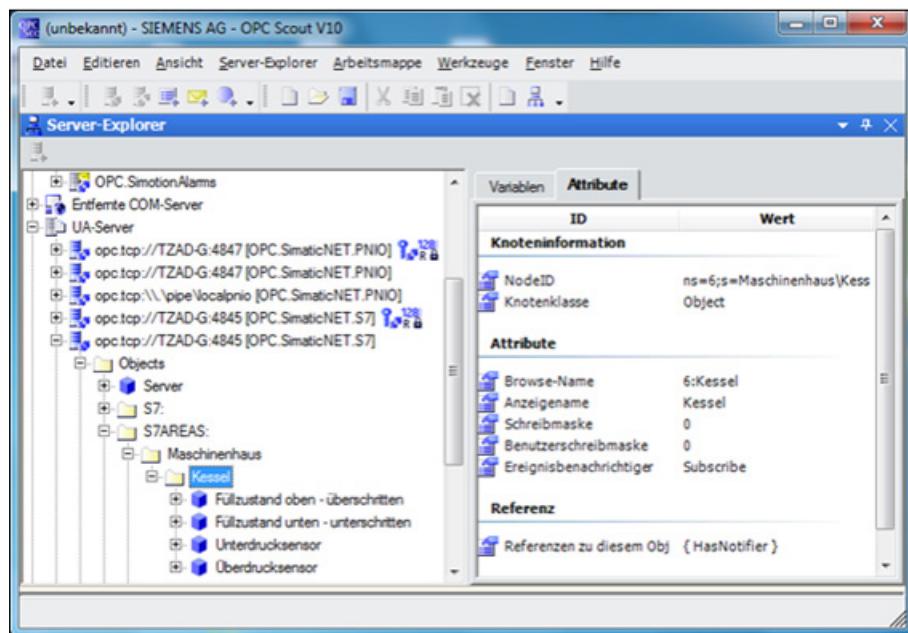
und der SourceNode

- ns="S7SOURCES:", "Station1\CPU 416-3\S7-Programm(1)\DB404"

Diagnosemeldungen weisen keinen "Zusatztext 1" und "Zusatztext 2" auf. Analog zu den Alarmen wird als Ersatz für "Zusatztext 2" der Pfad in STEP 7 im Anlagebaum verwendet und als Ersatz für den Zusatztext 1 der Pfad in STEP 7 im Anlagebaum + Meldebezeichner. Z. B. generiert die Diagnosemeldung mit Meldebezeichner im obenstehenden "S7-Programm(1)" neben den bereits obenstehenden UA-Ordnern zusätzlich noch den SourceNode

- ns="S7SOURCES:", "Station1\CPU 416-3\S7-Programm(1)\WR\_USMSG (1)"

Der Bereichsbbaum und der Herkunftsraum können mit dem OPC Scout V10 durchsucht werden.



Ausgewählt ist der Bereichsknoten mit dem Anzeigenamen "con13 Scan Notify" (Nodeld ns="S7AREAS:", s=" con13-Scan-Notify") im linken Fenster. Im rechten Fenster werden seine Attribute angezeigt, es ist diesem zu entnehmen, dass der Bereichsknoten die Referenz "HasNotifier" aufweist.

## 2.7.16 Empfang von Events

### EventItems und EventNotifier

Die einzige Möglichkeit aktuelle Events von einem Server zu empfangen ist das Erzeugen von einem oder mehreren beobachteten Event-Items in einer Subscription. Bei einem Event-Item wird das Attribut EventNotifier (12) beobachtet. EventNotifier weisen nur Objekte auf, die die Referenz "HasNotifier" haben. Bei S7 UA-Alarming mit OPC-Server 8.0 sind das der Server-Knoten ns="http://opcfoundation.org/UA/", i=2253, alle Knoten der S7-Verbindungen, z.B. ns="S7:", s="S7-Verbindung\_1" und alle Bereichsknoten im ns="S7AREAS:". Die EventNotifier des Server-Knotens und der Knoten der S7-Verbindungen sind die Einzigsten, die Events für unprojektierte S7-Meldungen melden.

Bei Erzeugen eines beobachteten Event-Items, müssen die zurück zuliefernden Properties angegeben werden. Werden keine Properties angegeben, weiß man, dass ein Event aufgetreten ist, jedoch nicht welches. Für die Aufnahme einer Property in die Liste der zurück zuliefernden Properties, muss für jede Property folgendes angegeben werden:

- die Nodeld des definierten Eventtyps (Typeld),
- der Browse-Name ggf. Browse-Pfad von der Property und
- die Attributeld

Nodeld, Browse-Namen ggf. Browse-Pfad und die Attributeld des definierten Eventtyps können Sie dem Kapitel "Eventtyp hierarchie von S7-UA-Alarming-Server (Seite 201)" entnehmen.

---

#### **Hinweis**

An der OPC-UA-Schnittstelle werden nicht alle Zustandswechsel eines Alarms durch ein Event weitergereicht. Es kann vorkommen, dass bei schneller Änderung des Alarmzustands, nicht alle Zustandswechsel mit einem Event mitgeteilt werden, sondern nur der letzte nun aktuelle Zustand gesendet wird. Dies ist abhängig von OPC-Parametern, wie z.B. "PublishingInterval" und "SamplingInterval" sowie der Projektierung und der Rechnerleistung.

---

### **Auswahl der Properties**

Ein Event muss nicht alle zurück zu liefernden Properties aufweisen. Für Properties, die ein Event nicht hat, wird "null" zurückgemeldet. Dabei kann S7-OPC-UA-Alarming nicht zwischen Properties unterscheiden, die das Event nicht hat, oder solchen, die aufgrund von der fehlerhaften Angabe z. B. des Browse-Namen, keine gültige Property bezeichnen.

### **Verwendung von Filterbedingungen**

Wird ein EventNotifier eines Bereichsknoten beobachtet, werden alle Events gemeldet, die in diesem Bereich entstanden sind. Die Anzahl der gemeldeten Events kann je nach Ausprägung der Anlage (die Menge der potentiell verfügbaren EventSourcen kann dem Herkunftsraum kombiniert mit dem Bereichsraum entnommen werden) sehr hoch sein. Daher ist eine sinnvolle Filterung der Events erforderlich.

## **2.7.17        Methoden von UA-Alarmen**

### **Enable()/Disable()**

Für einen Alarm, der disabled ist, werden keine Events an Clients generiert. Zur S7-Steuerung hin werden jedoch unabhängig davon die Alarne immer überwacht, da Alarne dorthin nicht einzeln enabled/disabled werden können.

### **AddComment()**

Für jede Alarminstanz kann jeweils ein Kommentar gespeichert werden.

### **ConditionRefresh()**

Es werden alle aktiven oder unquittierten Alarne gemeldet.

## Acknowledge()

Die Quittierung eines Alarms wird zur S7-Steuerung transportiert. Die Umschaltung des Quittungszustandes wird unabhängig davon von der S7-Steuerung zum OPC-Server explizit zurückgemeldet, erst dann wechselt der Alarm OPC-seitig in den quittierten Zustand und löst entsprechende Events aus.

# 2.8 Optimierte S7-Kommunikation mit OPC UA

## 2.8.1 Eigenschaften der optimierten S7-Kommunikation mit OPC UA

Der SIMATIC NET OPC-Server ermöglicht die Nutzung der S7-Kommunikation über OPC UA und Industrial Ethernet zu den neuen S7-1200- und S7-1500-Stationen.

Der S7OPT-OPC-UA-Server von SIMATIC NET hat folgende Eigenschaften:

- Variabeldienste  
Zugriff und Beobachtung von S7-Variablen über Standardzugriff und optimierten Zugriff.  
Der optimierte Zugriff erfolgt über symbolische Adressierung.
- S7-CPU Schutzstufenkonzept  
Setzen eines Passworts zum geschützten Verbindungsaufbau und Zugriff auf die S7-1200- und S7-1500-Stationen
- OPC-UA-Events, -Conditions und -Alarne  
Verarbeitung von PLC-Meldungen

## 2.8.2 SIMATIC NET OPC-UA-Server für das optimierte S7-Protokoll

### Einleitung

Der folgende Abschnitt beschreibt die Konfiguration für das optimierte S7-Protokoll über Industrial Ethernet und OPC UA.

### Konfiguration

Die Aktivierung des S7OPT-OPC-UA-Servers erfolgt durch die Auswahl von "OPC UA" (in der Zeile "S7 optimiert") im Konfigurationsprogramm "Kommunikations-Einstellungen" im Katalog "OPC-Protokollauswahl". Für "S7 optimiert" wird kein Häkchen angezeigt, da das optimierte S7-Protokoll ausschließlich bei OPC UA zur Verfügung steht. Daher wird durch die Aktivierung des S7OPT-UA-Servers die Inproc/Outproc-Konfiguration der anderen S7-COM-OPC-Data-Access-Server nicht beeinflusst.

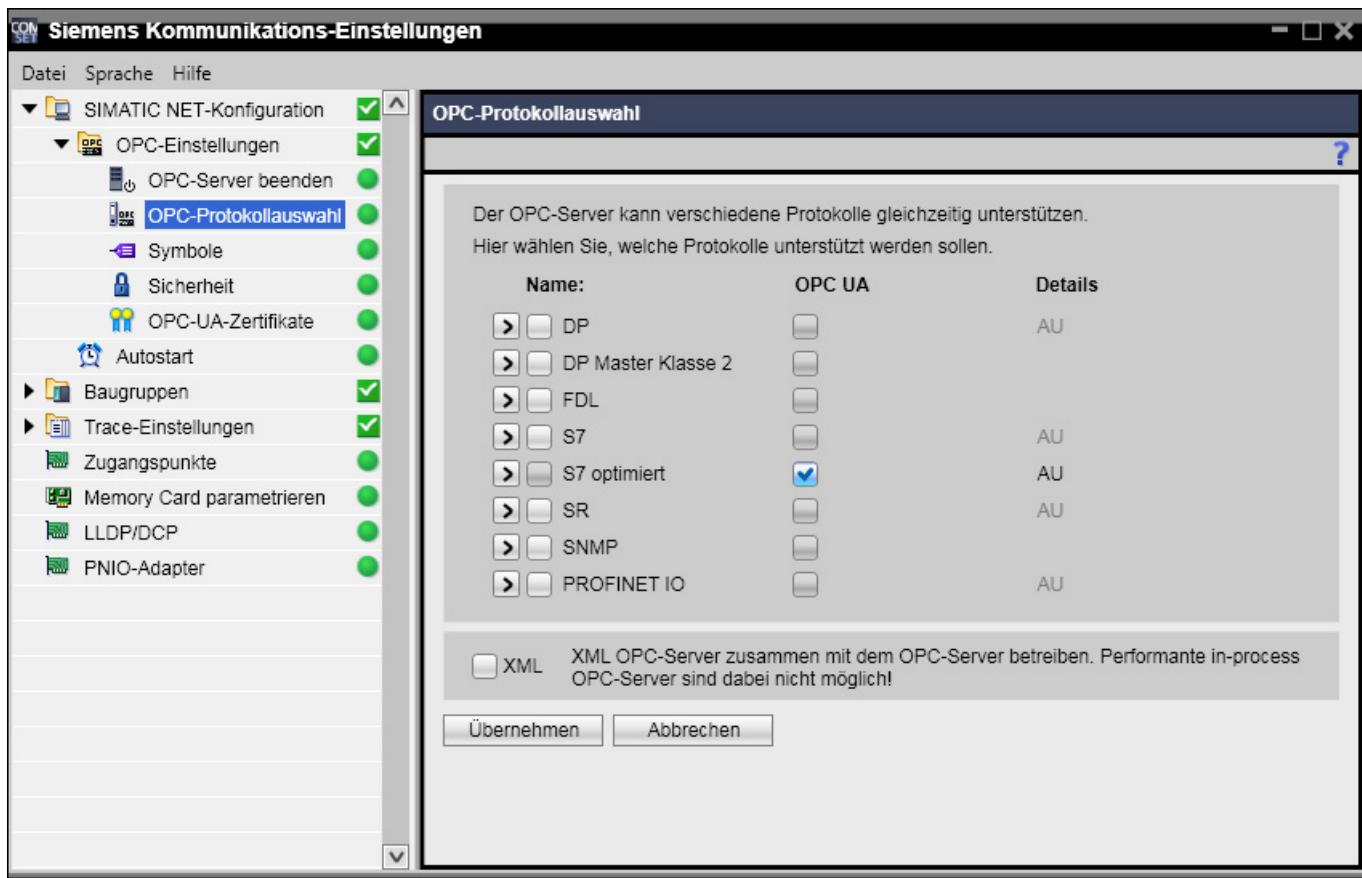


Bild 2-40 Fenster des Konfigurationsprogramms "Kommunikations-Einstellungen" zur Auswahl von OPC UA für das optimierte S7-Protokoll

#### Hinweis

Der optimierte Zugriff auf S7-Variablen der S7-1200- und S7-1500-Stationen ist ausschließlich über symbolische Adressierung möglich.

#### Hinweis

Für die Erstellung von Symbolen mit STEP 7 Professional (TIA Portal) oder dem Symbol-Editor ist die Verwendung folgender Zeichen erlaubt: A-Z, a-z, 0-9, \_, -, ^, !, #, \$, %, &, ', /, (, ), <, >, =, ?, ~, +, \*, ', :, |, @, [ , ], { , }, ". Zusätzlich sollten Sie bei der Erstellung von Symbolen mit STEP 7 darauf achten, dass es bei der Array-Auflösung zu Problemen kommen kann, wenn Ihre Symboldatei gleichzeitig Symbole der Form <Symbolname> und <Symbolname>[<Index>] enthält.

## Vorteile / Nachteile

Bei Verwendung des S7OPT-OPC-UA-Servers ist nur der Outproc-Betrieb von "S7 optimiert" möglich. Der S7OPT-OPC-UA-Server-Prozess muss zum Erhalt der Empfangsbereitschaft in Betrieb sein. Ein Beenden des S7OPT-OPC-UA-Servers auch nach Abmeldung aller OPC-UA-Clients wird nicht ausgeführt und darf nicht ausgeführt werden.

Dem stehen folgende Vorteile gegenüber:

- Es ist keine COM/DCOM-Konfiguration mehr nötig.
- Performante, sichere Kommunikation
- Für PLC-Meldungen und Datenzugriffe ist nur noch ein OPC-Server erforderlich.

### Hinweis

Wenn der S7OPT-OPC-UA-Server für das S7-Protokoll aktiv ist, werden als permanent projektierte S7-Verbindungen aufgebaut, sobald der erste OPC-UA-Client am S7OPT-OPC-UA-Server angemeldet ist. Wenn sich kein OPC-UA-Client verbunden hat, z.B. nach Hochfahren des Rechners, werden noch keine permanenten S7-Verbindungen aufgebaut.

### Hinweis

S7-Verbindungen für das "S7 optimiert"-Protokoll können mit der Eigenschaft "Verbindung bei Anforderung aufbauen" in SIMATIC STEP 7 Professional (TIA Portal) projektiert werden. Beim ersten Zugriff auf eine Variable, die diese Verbindung benötigt, wird zuerst die Verbindung aufgebaut und erst danach erfolgt der eigentliche Zugriff. Somit kann der erste Zugriff auf die Variable länger dauern und etwaige Fehler beim Verbindungsauflauf melden.

## 2.8.3 Wie wird der S7OPT-OPC-UA-Server adressiert?

### Server-URL

Für das native binäre TCP-Protokoll gibt es für den OPC-Client zwei Möglichkeiten der Server-Adressierung:

- Direkte Adressierung:
  - `opc.tcp://<hostname>:55105`
  - oder
  - `opc.tcp://<IP-Adresse>:55105`
  - oder
  - `opc.tcp://localhost:55105`

Der S7OPT-OPC-UA-Server hat den Port 55105.

- Die URL des S7OPT-OPC-UA-Servers kann auch über den OPC-UA-Discovery-Dienst gefunden werden.

Die Eingabe zum Auffinden des Discovery-Servers lautet:

- `opc.tcp://<hostname>:4840`

oder

– opc.tcp://<IP-Adresse>:4840

Der Discovery-Server hat den Port 4840 (für TCP-Verbindungen).

## **IPv6-Adresse**

Für die IP-Adresse kann auch eine IPv6-Adresse verwendet werden. Die Adresse muss in Klammern angegeben werden, z.B. [fe80:e499:b710:5975:73d8:14]

## **Endpunkte und Sicherheitsmodi**

Der SIMATIC NET S7OPT-OPC-UA-Server unterstützt Endpunkte mit dem nativen binären TCP-Protokoll und ermöglicht Authentisierung über Zertifikate und eine verschlüsselte Übertragung.

Der Discovery-Dienst auf dem angesprochenen Host meldet die Endpunkte der Server, sowie deren Sicherheitsanforderungen und Protokollunterstützung.

Die Server-URL "opc.tcp://<hostname>:55105" des S7OPT-OPC-UA-Servers bietet folgende Endpunkte:

- Endpunkt im Sicherheitsmodus "SignAndEncrypt":  
Zur Kommunikation mit dem Server werden Signierung und Verschlüsselung gefordert.  
Die Kommunikation ist durch Zertifikataustausch und Passworteingabe geschützt.  
Zusätzlich zum Sicherheitsmodus wird die Sicherheitsrichtlinie Basic128Rsa15 angezeigt.
- Endpunkt im Sicherheitsmodus "keiner":  
In diesem Modus werden keine Sicherheitsfunktionen vom Server gefordert (Sicherheitsrichtlinie "None").

Weitere Details zu den Sicherheitsfunktionen finden Sie im Kapitel "OPC-UA-Schnittstelle programmieren (Seite 508)".

Die Sicherheitsrichtlinien "Basic128Rsa15" und "None" finden Sie in der UA-Spezifikation der OPC Foundation unter folgender Internet-Adresse:

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

Weitere Informationen finden Sie auf folgender Internetseite:

OPC Foundation (<http://www.opcfoundation.org/profilereporting/index.htm>)  
> "Security Category" > "Facets" > "Security Policy"

## **Die OPC-UA-Discovery des OPC Scout V10**

Der OPC Scout V10 ermöglicht das Öffnen des OPC-UA-Discovery-Dialogs zur Übernahme von OPC-UA-Endpunkten in den Navigationsbereich des OPC Scout V10.

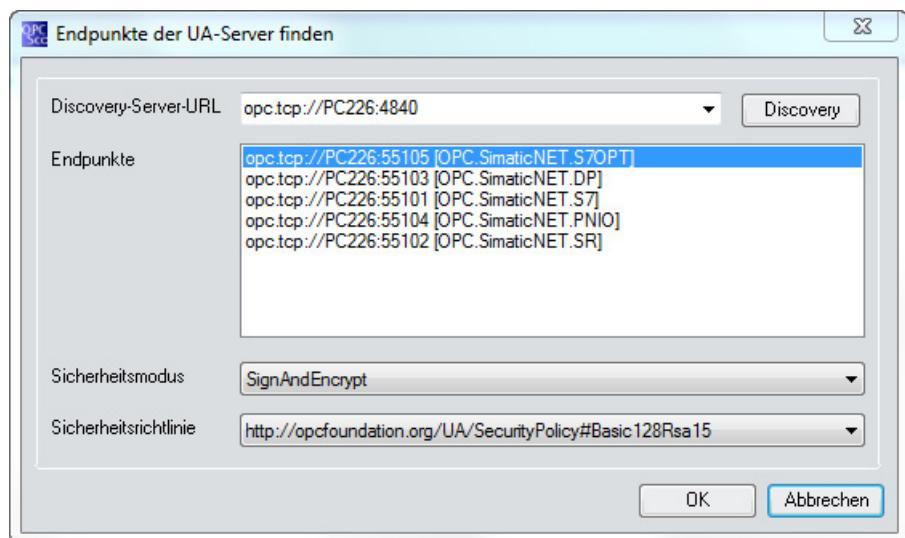


Bild 2-41 Das Dialogfeld "Endpunkte der UA-Server finden" des OPC Scout V10

Der S7OPT-OPC-UA-Server kann über den OPC-UA-Discovery-Dienst gefunden werden. Zur Eingabe siehe oben unter "Server-URL".

Der OPC Scout V10 beinhaltet eine Liste der OPC-UA-Endpunkte. Der Discovery-Dienst auf dem angesprochenen Host meldet dann die registrierten OPC-UA-Server sowie deren Ports und Sicherheitsmodi.

Weitere Details finden Sie in der Online-Hilfe des OPC Scout V10.

## 2.8.4 Welche Namensräume bietet der S7OPT-OPC-UA-Server an?

Der S7OPT-OPC-UA-Server bietet folgende Namensräume an:

Namensraum-Index	"Bezeichner" (Namensraum-URI) / Kommentar
0	"http://opcfoundation.org/UA/" von der OPC Foundation spezifiziert
1	"urn:Siemens.Automation.SimaticNET.S7OPT:(GUID)" Eindeutiger Bezeichner des lokalen performanten S7OPT-OPC-UA-Servers.
2	"S7OPTTYPES:" Definitionen für S7OPT-spezifische Objekttypen.
3	"S7OPT:" Bezeichner des lokalen performanten S7OPT-OPC-UA-Servers.
4	"S7OPTSOURCES:" Bezeichner für Quellen von Alarmereignissen.

Namensraum-Index	"Bezeichner" (Namensraum-URI) / Kommentar
5	"S7OPTAREAS:" Bezeichner für Bereiche einer Alarmhierarchie.
6	"SYM:" Optionaler Server mit ATI-S7OPT-Symbolik; abhängig von der Projektierung und der Konfiguration der PC-Station (durchsuchbar und verwendbar mit OPC UA). Alternativ kann hier ein Präfix stehen, der in der Symbolik-Parametrierung ("Kommunikations-Einstellungen") festgelegt wird.

Die Namensraum-Indizes 0 und 1 sind reserviert und in ihrer Bedeutung von der OPC Foundation spezifiziert.

Die Zuordnung der restlichen Namensraum-Indizes zu den Bezeichnern (Namensraum-URI) muss zu Beginn einer OPC-UA-Session vom Client unter Angabe des Bezeichners über die Datenvariable "NamespaceArray" ermittelt werden. Die Bezeichner "S7OPTTYPES:", "S7OPT:", "S7OPTAREAS:" und "S7OPTSOURCES:" sind beim S7OPT-OPC-UA-Server immer vorhanden.

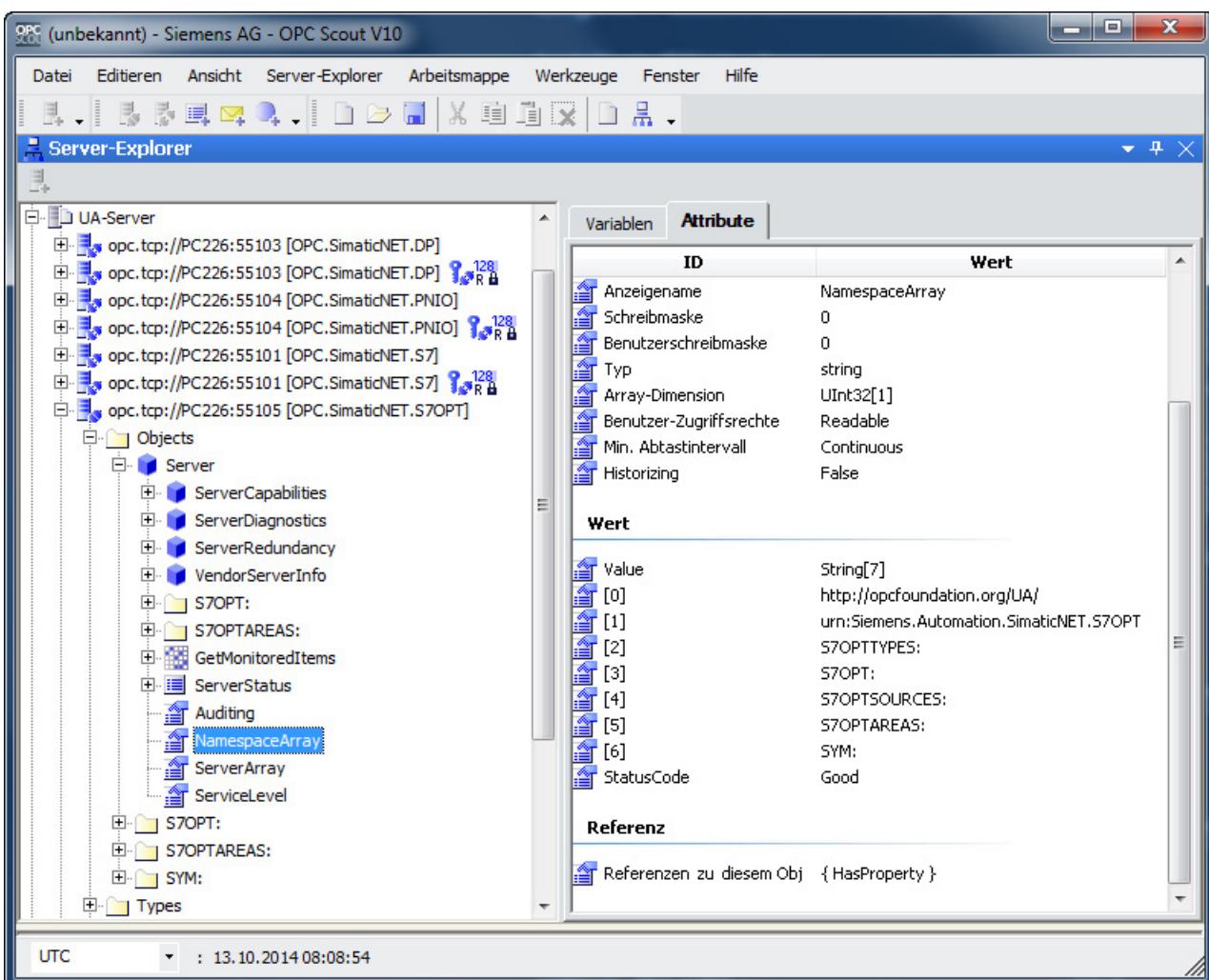


Bild 2-42 Anzeige der S7OPT-OPC-UA-Namensräume mittels der Browse-Funktion des OPC Scout V10

## 2.8.5 Die Nodeld

### Identifikation einer S7OPT-Prozessvariable

Die Nodeld identifiziert mit Hilfe des folgenden Tupels eine S7OPT-Prozessvariable eindeutig:

- Namensraum-Index
- Bezeichner (Zeichenfolge)

## Beispiele

- Nodeld:
  - Namensraum-URI:  
S7OPT:  
(= Namensraum-Index 3)
  - Bezeichner:  
S7-Verbindung\_1.m.10,b

## Wie verhält sich der OPC-UA-Namensraum?

Die OPC-UA-Sicht ist auf Automatisierungsobjekte auch auf verschiedene Eigenschaften der Objekte bezogen. OPC UA greift auf Objekte und deren Unterobjekte zu.

- Datenvariablen, Methoden und zum Teil Ereignisse sind beispielsweise Unterobjekte eines "S7 optimiert"-Verbindungsobjekts. Attribute und Properties definieren die Objekte näher.

Den qualifizierten Bezeichnern der Nodelds kommt unter OPC UA eine große Bedeutung zu. Jeder einzelne Zugriff auf ein Objekt, Unterobjekt, Property und Attribut erfolgt über dessen Nodeld.

Unter anderem für die Unterstützung durch lokale Sprachen sieht OPC UA den Anzeigenamen vor. So kann ein und dasselbe Objekt beispielsweise in unterschiedlichen Sprachumgebungen, die der OPC-UA-Client vorgibt, unterschiedlich durchsucht werden, wobei aber jedes Mal dieselbe Nodeld präsentiert wird. Der Anzeigename wird analog zur jeweiligen Nodeld gewählt. Die Texte des gesamten Namensraums sind in Englisch.

## Syntax der S7OPT-Prozessvariablen

Mit OPC UA wird eine optimierte Syntax eingeführt. Die Nodelds aller OPC-UA-Objekte haben folgenden Aufbau:

<verbindungsobjekt>."<unterobjekt>".<property>

Ein Unterobjekt kann weitere Unterobjekte beinhalten.

Eine nicht interpretierbare Nodeld wird mit einem Fehler zurückgewiesen. Die Groß- oder Kleinschreibung der Buchstaben "A-Z" wird bei allen Nodelds ignoriert.

## Symbolische Objektdarstellung

Die OPC-UA-Spezifikation empfiehlt zur hierarchischen Beschreibung des Adressraums eine einheitliche Symboldarstellung. Folgende Symbole werden in diesem Dokument verwendet:

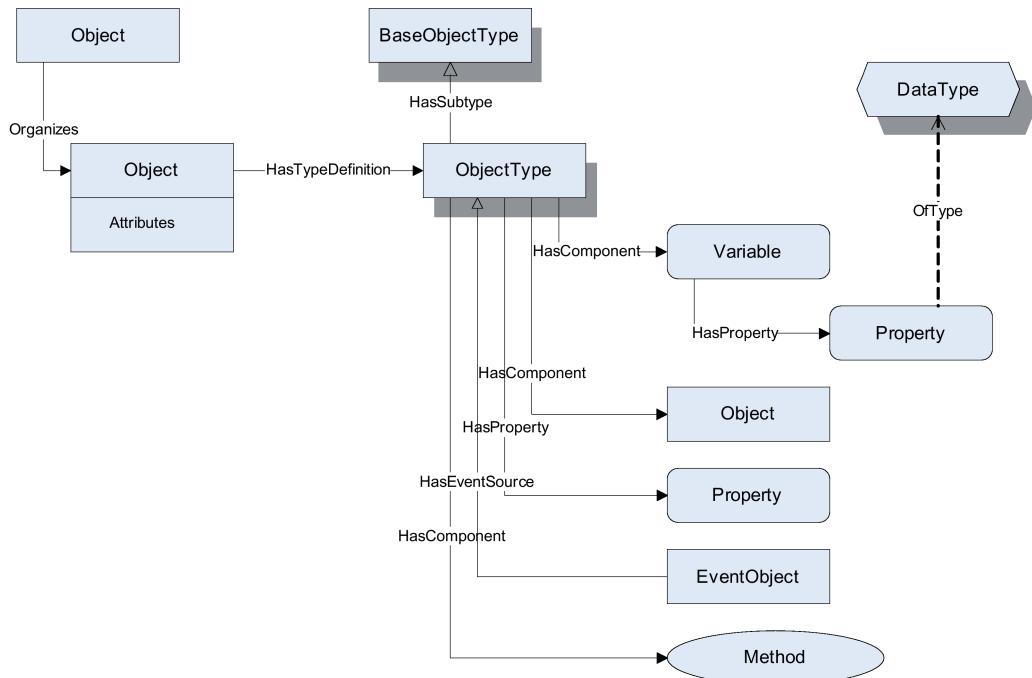


Bild 2-43 Symbole des OPC-UA-Adressraums

## 2.8.6 Verbindungsobjekte

### S7OPT-Verbindungs-Objekte

Es gibt folgende S7OPT-Verbindungs-Objekte:

- Produktive S7-Verbindungen  
S7-Verbindungen für die optimierte S7-Kommunikation werden zum Datenaustausch zwischen Automatisierungsgeräten genutzt und über STEP 7 Professional (TIA Portal) projektiert.
- Das CpuSubscription–Objekt  
Es dient zum Anmelden der zyklischen Überwachung von Datenänderungen direkt in der CPU, ohne dass ein Pollen erforderlich ist. Die S7-1200- / S7-1500-Stationen unterstützen ein begrenztes Mengengerüst an CpuSubscriptions.

#### Hinweis

Informationen zu den Mengengerüsten der CPUs finden Sie in den jeweiligen Gerätehandbüchern im Kapitel "Technische Daten".

### CpuSubscription-Objekt

<Verbindungsobjekt>:= "cpusubscription"

### Hinweis

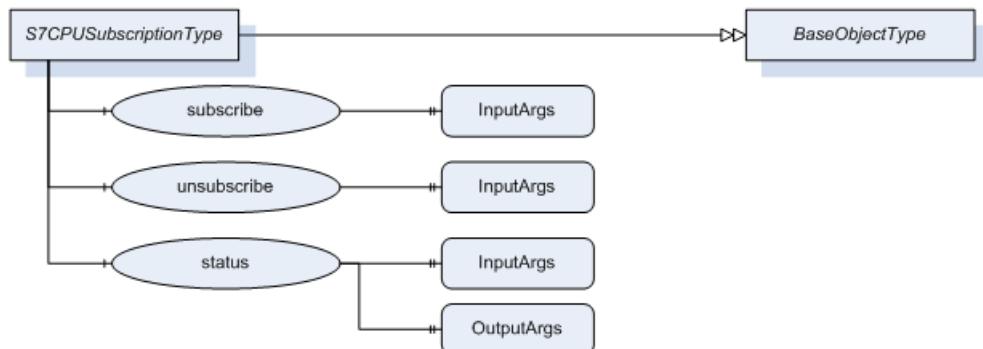
Der Verbindungsname "CpuSubscription" ist reserviert und darf niemals als Name einer S7-Verbindung eingesetzt werden. STEP 7 Professional (TIA Portal) verhindert dies beim Anlegen der Verbindungsprojektierung.

Der S7OPT-OPC-UA-Server bietet zur S7-1200- und S7-1500-Station die Überwachung von Datenänderungen direkt in der CPU an, ohne dass auf der S7-Verbindung zyklische Leseaufträge erforderlich sind. Durch die Verwendung dieser CpuSubscriptions können Sie die Kommunikationslast Ihrer CPU erheblich reduzieren.

Das zur Verfügung stehende Mengengerüst an CpuSubscriptions ist streng limitiert und vom Typ der verwendeten S7-Station abhängig. Es liegt im Bereich von einigen hundert Objekten mit wenigen tausend Bytes und kann zur Laufzeit vom S7OPT-OPC-UA-Server nicht ermittelt werden. Weiterhin steht es dem S7OPT-OPC-UA-Server auch nicht ausschließlich zur Verfügung, sondern kann auch von weiteren Applikationen genutzt werden.

Daher bietet der S7OPT-OPC-UA-Server seinen UA-Clients die Möglichkeit an, über OPC-UA-Methoden, von ihm bevorzugte Datenvariablen für CpuSubscriptions abzufragen und den aktuellen Status dieser CpuSubscriptions in Erfahrung zu bringen.

Diese Methoden sind nicht an eine einzige S7-Verbindung gebunden. Sie nehmen als Parameter OPCUA\_Nodelds für die Datenvariablen an, über die auf den jeweiligen Verbindungsweg zur CPU geschlossen werden kann. Weiterhin können auch Datenvariablen der Symbolik, deren OPCUA\_Nodelds normalerweise keinen Verbindungsnamen enthalten, in die CpuSubscriptions eingebunden werden.



### Syntax CpuSubscription-Methoden

Namensraum-URI: S7OPT: (Namensraum-Index: 3)

`cpusubscription.subscribe(OpcUa_UInt32 cycletime, OpcUa_NodeId[] nodeids)`  
`cpusubscription.unsubscribe(OpcUa_NodeId[] nodeids)`  
`cpusubscription.status(OpcUa_NodeId[] nodeids, OpcUa_Status[] status)`

### Beispiele:

```

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)
```

### Erklärung der OPC-UA-Methoden

- **subscribe()**

Legt die CpuSubscription für mehrere Datenvariablen an. (Im Fehlerfall werden entsprechende OPC-UA-Fehlercodes angeboten, siehe auch Kapitel "Meldungen der OPC-UA-Server (Seite 526)")

- InputArgument1: "cycletime"; Intervall, das Datenänderungen von der PLC an den S7OPT-OPC-UA-Server meldet.

Datenvariable vom OPC-UA-Datentyp "UInt32", Angabe in ms, mindestens 100 ms, nur ganzzahlige Vielfache von 100 ms

- InputArgument2: "nodeids"; Angabe von einer oder mehreren Nodelds, die über die CpuSubscription-Zugriffsmöglichkeit angelegt werden sollen.

Datenvariable vom OPC-UA-Datentyp "OpcUa\_Nodeld[]"

### Hinweis

Wenn für eine Nodeld schon eine CpuSubscription angelegt wurde, wird diese mit geänderter Zykluszeit modifiziert.

- **unsubscribe()**

Entfernt die CpuSubscription. (Im Fehlerfall werden entsprechende OPC-UA-Fehlercodes angeboten, siehe auch Kapitel "Meldungen der OPC-UA-Server (Seite 526)")

- InputArgument1: "nodeids"; Liste der Nodelds kann angezeigt werden

Datenvariable vom OPC-UA-Datentyp "OpcUa\_Nodeld[]"

- **status()**

Gibt den CpuSubscription-Status zurück. (Im Fehlerfall werden entsprechende OPC-UA-Fehlercodes angeboten, siehe auch Kapitel "Meldungen der OPC-UA-Server (Seite 526)")

- InputArgument1: "nodeids"; Liste der Nodelds kann angezeigt werden

Datenvariable vom OPC-UA-Datentyp "OpcUa\_Nodeld[]"

- OutputArgument1: "status"

Datenvariable vom OPC-UA-Datentyp "OpcUa\_Status[]"

Der Aufruf der CpuSubscription-Methoden fügt die jeweils übergebenen Nodelds der internen CpuSubscription-Verwaltung des S7OPT-OPC-UA-Servers hinzu oder entfernt diese, auch unabhängig vom jeweiligen Verbindungszustand der Verbindungspartner. Der (erfolgreiche) Abschluss eines Methodenaufrufs zeigt deshalb nicht an, dass die CpuSubscription in der CPU tatsächlich erzeugt werden konnte. Sind falsche Nodelds verwendet worden, liefern die Funktionen einen Parameterfehler und dann ist keine der übergebenen Nodelds an die CpuSubscription-Verwaltung übergeben worden.

### Hinweis

Wenn die Anzahl der verfügbaren CpuSubscriptions oder Datenvariablen überschritten wurde, dann kann die Auswertung des Methodenrückgabeparameters keinen Aufschluss über das erfolgreiche Anlegen der CpuSubscription in der CPU geben. Überprüfen Sie aus diesem Grund den erfolgreichen Abschluss einer Subscribe- oder Unsubscribe-Methode durch einen anschließenden Aufruf der Status-Methode und reagieren Sie im Fehlerfall durch Anpassung beziehungsweise Reduzierung der Datenvariablen/CpuSubscriptions darauf.

Nach erfolgreichem Verbindungsaufbau oder nach Hinzufügen beziehungsweise Entfernen einer CpuSubscription optimiert der S7OPT-OPC-UA-Server alle bis dahin angemeldeten CpuSubscriptions auf der jeweiligen Verbindung und meldet diese beim Verbindungspartner an oder wieder ab.

Falls die Anmeldung beim Verbindungspartner erfolgreich war, liefert dieser nach einer Datenänderung automatisch die geänderten Daten. Beim S7OPT-OPC-UA-Server eingerichtete Monitored-Items mit gleicher oder größerer Aktualisierungsrate, die sich auf diese CpuSubscriptions abbilden lassen, werden nun nicht mehr gepollt.

Falls die Anmeldung beim Verbindungspartner nicht erfolgreich war, werden die auf diese CpuSubscription abgebildeten Monitored-Items gepollt, was dem Standardverhalten ohne Verwendung der CpuSubscriptions entspricht. Der Status dieser CpuSubscription wird auf „Bad“ gesetzt, was mit der Status-Methode ausgelesen werden kann, siehe auch Kapitel „Meldungen der OPC-UA-Server (Seite 526)“.

Die CpuSubscriptions wirken über alle UA-Clients gemeinsam, das heißt ein UA-Client kann eine CpuSubscription entfernen, die ein anderer Client eingerichtet hat. Nachdem der letzte UA-Client abgemeldet ist, werden alle zu diesem Zeitpunkt noch bestehenden CpuSubscriptions entfernt.

Für eine optimale Verwendung und Auslastung der CpuSubscription-Dienste sollten Sie bei der Einrichtung und Verwendung der CpuSubscriptions folgendes beachten:

- Melden Sie die CpuSubscriptions nur auf die wichtigsten Datenvariablen an
- Melden Sie größere, zusammenhängende Bereiche an, wenn Sie in diesem Datenbereich viele kleine Datenvariablen beobachten wollen.
- Melden Sie möglichst solche Datenvariablen an, die sich selten ändern, deren Änderungen dann aber eine sehr schnelle Rückmeldung erfordern
- Gleichen Sie die Zykluszeit der CpuSubscription an die Aktualisierungszeit Ihrer Monitored-Items an.

Für den OPC-Client verhält sich die Verwendung von CpuSubscriptions transparent. In jedem Fall (ohne CpuSubscription, erfolgreich angemeldete CpuSubscription, nicht erfolgreich angemeldete CpuSubscription) werden als Monitored-Items angemeldete

Datenvariablen entweder durch Pollen oder durch die CpuSubscriptions auf Datenänderungen überwacht.

Der S7OPT-OPC-UA-Server hat auf die Reihenfolge der Quittungsverarbeitung bzw. das Melden von Datenänderungen durch den Verbindungspartner keinen Einfluss. Es können sozusagen parallel sowohl Quittungen auf Leseaufträge als auch Datenänderungsmeldungen unterwegs sein. Der S7OPT-OPC-UA-Server aktualisiert deshalb seinen Speicher immer mit dem zuletzt bei ihm angekommenen Wert.

## 2.8.7 Verbindungsnamen

### Der Verbindungsname einer S7-Verbindung

Der Verbindungsname ist der in STEP 7 Professional (TIA Portal) projektierte Name zur Identifikation der Verbindung. Dieser Name heißt bei STEP 7 Professional (TIA Portal) "Lokale ID". Die Lokale ID ist innerhalb des OPC-Servers eindeutig.

### Verbindungstypen

Der OPC-Server unterstützt als Verbindungstyp die S7-Verbindung.

### Welche Zeichen sind für S7-Verbindungsnamen erlaubt?

Für den <Verbindungsname> sind Ziffern "0-9", alphabetische Zeichen in Groß- und Kleinschreibung "A-z" und Sonderzeichen erlaubt. Leerzeichen am Anfang und am Ende des Verbindungsnamen sind genauso verboten wie die Sonderzeichen „„, „|“, „\“, „/“, „[“, „]“. Der Verbindungsname darf maximal 24 Zeichen lang sein. Groß- und Kleinschreibung wird nicht unterschieden.

Weitere sichtbare Zeichen mit ASCII-Code > 126 sowie ASCII-Code < 30 sind nicht erlaubt.

Die Verbindungsnamen "SYSTEM" bzw. der Verbindungsname „cpusubscription“ sind reserviert und dürfen nicht verwendet werden.

### Beispiele für S7-Verbindungsnamen:

Typische Beispiele sind:

- S7-Verbindung\_1
- S7OPT-OPC-Verbindung

## 2.8.8 Aufbau und Funktionen des produktiven S7OPT-Verbindungsobjekts

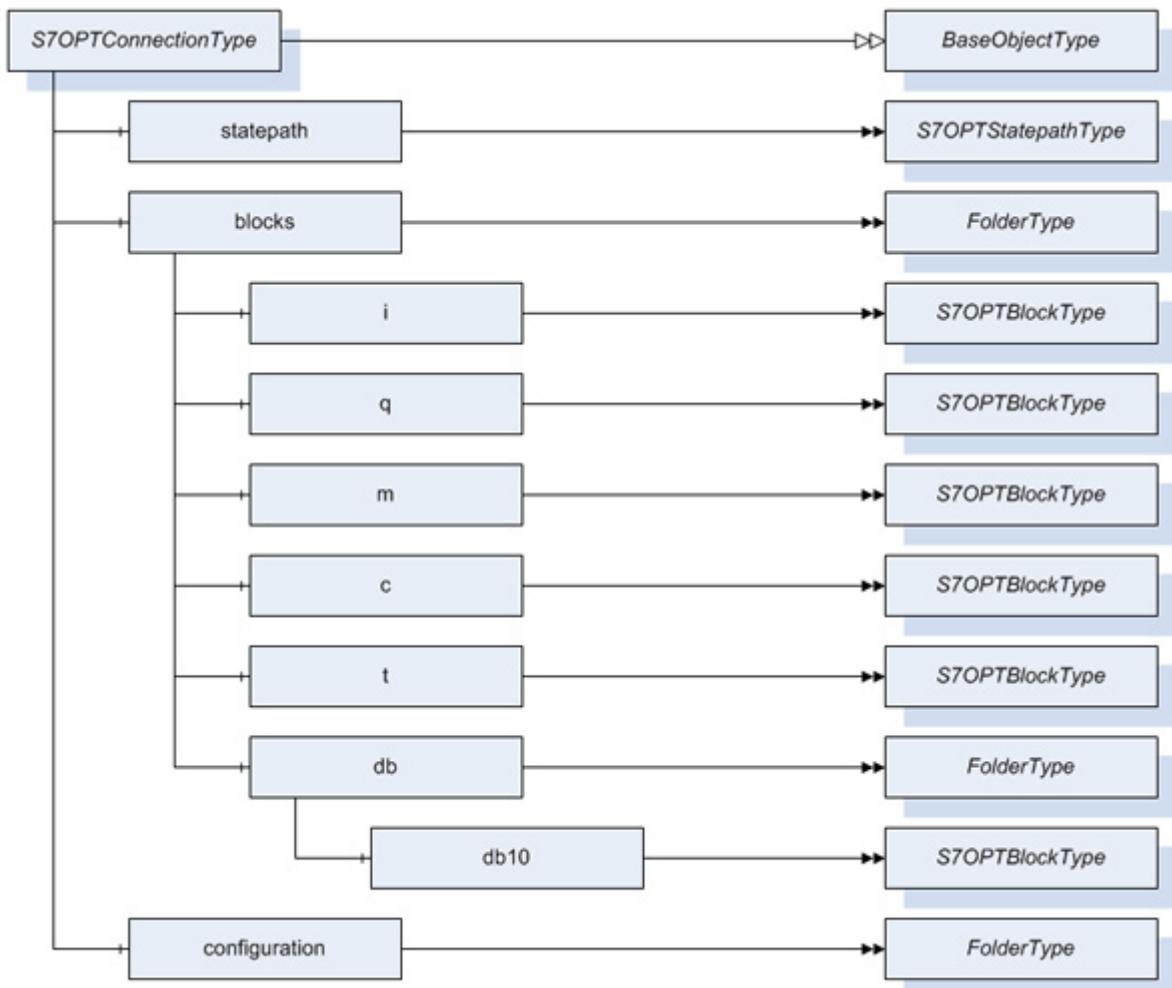
### Was sind S7OPT-Verbindungsobjekte?

Alle produktiven protokollspezifischen Objekte sind immer einer Verbindung zugeordnet. Bei "S7 optimiert" sind dies die Verbindungen zu Kommunikationspartnern (S7-Verbindung). Ausnahmen hiervon bildet die CpuSubscription.

## 2.8.9 Typ-Definition des S7OPT-Verbindungsobjekts

### Typ-Definition des S7OPT-Verbindungsobjekts

Für die Objekte und Funktionalitäten, die über eine produktive S7-Verbindung verwendbar sind, ist ein spezifischer OPC-UA-Objekttyp definiert:



Im OPC-UA-Namensraum für Objekte werden Instanzen dieses Typs angezeigt. Der Typ selbst kann unter "Typen" strukturiert ausgelesen werden.

## 2.8.10 S7OPT-Verbindungs-Informationsobjekte

### S7OPT-spezifische Datenvariablen für Informationen

Es gibt S7OPT-spezifische Datenvariablen, mit denen Sie Informationen über die optimierte S7-Kommunikation und die aufgebauten Verbindungen abfragen können.

Folgende Information kann ermittelt werden:

- Status einer S7-Verbindung mit optimiertem Zugriff

### Syntax der S7OPT-spezifischen Informationsvariable

Nodeld:

*Namensraum-Index: 3*

*<Verbindungsobjekt>.<Informationsparameter>*

### Erklärungen

*<Informationsparameter>:= "statepath"*

alarm	Instanz des zum Statepath-Objekt gehörenden Statepath-Alarms.		
statepath	<p>Zustand einer Kommunikationsverbindung zum Partnergerät</p> <p>Der Wert der Variable wird als Zahl ausgelesen und kann durch zusätzliches Auslesen des zugehörigen Enumstring {UNKNOWN, DOWN, UP, RECOVERY, ESTABLISH} einem Text zugeordnet werden.</p> <p>Variable vom UA-Typ "MultistateDiscreteType", nur lesbar</p>		
0	UNKNOWN	Für zukünftige Erweiterungen reserviert	
1	DOWN	Verbindung ist nicht aufgebaut	
2	UP	Verbindung ist aufgebaut	
3	RECOVERY	Verbindung ist nicht aufgebaut. Es wird versucht, die Verbindung aufzubauen.	
4	ESTABLISH	Für zukünftige Erweiterungen reserviert	

## 2.8.11 Beispiele für S7OPT-spezifische Informationsvariablen und Rückgabewerte

Hier finden Sie Beispiele, welche die Syntax der Namen von S7OPT-spezifischen Informationsvariablen verdeutlichen.

### Prioritätsstufe eines Statepath-Alarms

- Nodeld:
  - Namensraum-URI:  
S7OPT: (Namensraum-Index: 3)
  - Bezeichner:  
S7-Verbindung\_1.statepath.alarm.Severity

**Mögliche Rückgabewerte:** 500

### Mögliche Zustandstexte einer Kommunikationsverbindung

- Nodeld:
  - Namensraum-URI:  
S7OPT: (Namensraum-Index: 3)
  - Bezeichner:  
S7-Verbindung\_1.statepath.statepath.EnumStrings

**Mögliche Rückgabewerte:** { UNKNOWN | DOWN | UP | RECOVERY | ESTABLISH }

### Zustand einer Kommunikationsverbindung als Wert

- Nodeld:
  - Namensraum-URI:  
S7OPT: (Namensraum-Index: 3)
  - Bezeichner:  
S7-Verbindung\_1.statepath.statepath

**Mögliche Rückgabewerte:** 3 (RECOVERY) -Die Verbindung wird momentan aufgebaut.

## 2.8.12 Variabldienste

### 2.8.12.1 Standardzugriff

#### Was machen Variabldienste, die über den Standardzugriff aufgerufen werden?

Variabldienste, die über den Standardzugriff aufgerufen werden, ermöglichen den Zugriff und die Beobachtung von S7-Variablen im Automatisierungsgerät. Die Adressierung der S7-Variablen erfolgt als Namenskürzel der adressierten Objekte. Die Art des Zugriffs orientiert sich an der Notation der S7-Werkzeuge.

## Objekte im Automatisierungsgerät

Der S7OPT-OPC-UA-Server unterstützt folgende Objekte:

- Datenbausteine (Standardzugriff)
- Instanzdatenbausteine und Multiinstanzdatenbausteine (Standardzugriff)
- Eingänge
- Ausgänge
- Merker
- Timer (nur S7-1500)
- Counter (nur S7-1500)
- UDTs

Nicht jedes S7-Automatisierungsgerät unterstützt alle Objekttypen.

---

### Hinweis

Der Standardzugriff auf Variabeldienste kann auch über Symbole erfolgen. Alle Informationen dazu erhalten Sie im Handbuch "PC-Station in Betrieb nehmen"

---

## Syntax der Variabeldienste

### Syntax der Prozessvariablen

Vereinfachte Syntax der Prozessvariablen der S7OPT-OPC-UA-Nodeld:

*Namensraum-URI: S7OPT: (Namensraum-Index: 3)*

### Klassische Syntax

Es gibt drei Möglichkeiten:

```
<Verbindungsname>.<S7OPTObjekt>.<Adresse>{,<S7OPTTyp>{,<Anzahl>}}  
<Verbindungsname>.<S7OPTTimerObjekt>.<Adresse>  
{,<S7OPTTimerTyp>{,<Anzahl>}}  
<Verbindungsname>.<S7OPTCounterObjekt>.<Adresse>  
{,<S7OPTCounterTyp>{,<Anzahl>}}
```

### Erklärungen

Namensraum-URI: S7OPT: (Namensraum-Index: 3)

#### **<Verbindungsname>**

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

Das folgende Trennzeichen ist der Punkt (".").

**<S7OPTObjekt>**

Angabe des Bereichstyps in der S7. Mögliche Werte sind:

Parameter	Bedeutung
db<Nr>	Datenbaustein oder Instanzdatenbaustein. Kennzeichen für eine S7OPT-Variablen aus einem Datenbaustein. Instanzdatenbausteine unterscheiden sich in der S7-Kommunikation nicht von normalen Datenbausteinen. Deshalb kann auf die Vergabe einer zusätzlichen Kennung zur Vereinfachung verzichtet werden. <Nr> Nummer des Datenbausteins oder Instanzdatenbausteins ohne führende Nullen. Es können nur die nicht-optimierten Datenbausteine in der S7-1200 / S7-1500 über diese Datenvariablen angesprochen werden. Für Datenvariablen in optimierten Datenbausteinen lässt sich zur Laufzeit keine Adressinformation bilden. Der Zugriff auf diese optimierten Datenvariablen erfolgt über den symbolischen Zugriff.
m	Merker
i	Eingang Schreib- und lesbar Es gibt vereinfachend nur diese englische Bezeichnung.
q	Ausgang Schreib- und lesbar Es gibt vereinfachend nur diese englische Bezeichnung.

Das folgende Trennzeichen ist der Punkt (".").

**<S7OPTTimerObjekt>**

Typ-Bezeichner	S7-Datentyp	OPC-UA-Datentyp	Beschreibung
t	TIMER	UInt16	Die folgende Adressangabe ist eine Timer-Nummer.

**<S7OPTCounterObjekt>**

Typ-Bezeichner	S7-Datentyp	OPC-UA-Datentyp	Beschreibung
c	COUNTER	UInt16	Die folgende Adressangabe ist eine Counter-Nummer.

**<Adresse>**

Byte-Adresse der ersten Variablen, die angesprochen werden soll. Es werden keine führenden Nullen (z.B. 001) unterstützt.

Der Wertebereich für die Byte-Adresse ist 0...65534. Geräte- und typabhängig kann der tatsächlich verwendbare Wert der Adresse geringer sein.

### <S7OPTTyp>

Ein S7-Datentyp wird im OPC-UA-Server in den entsprechenden OPC-UA-Datentyp umgewandelt. Die folgende Tabelle listet den Typ-Bezeichner und den entsprechenden OPC-UA-Datentyp auf, in dem der Variablenwert dargestellt werden kann.

Typ-Bezeichner	S7-Datentyp	OPC-UA-Datentyp	Beschreibung
b	BYTE	Byte	8 Bit (Bitstring)
	USINT	Byte	8 Bit (unsigned)
c	CHAR	SByte	8 Bit (Charakter)
	SINT	SByte	8 Bit (signed)
w	WORD	UInt16	16 Bit (Bitstring)
	UINT	UInt16	16 Bit (unsigned)
dw	DWORD	UInt32	32 Bit (Bitstring)
	UDINT	UInt32	32 Bit (unsigned)
lw	LWORD	UInt64	64 Bit (Bitstring); nur für S7-1500 verfügbar
	ULINT	UInt64	64 Bit (unsigned); nur für S7-1500 verfügbar
i	INT	Int16	16 Bit (signed)
di	DINT	Int32	32 Bit (signed)
li	LINT	Int64	64 Bit (signed); nur für S7-1500 verfügbar
r	REAL	Float	Fließkomma (4 Byte) IEEE 754
lr	LREAL	Double	Fließkomma (8 Byte) IEEE 754
dt	DATE_TIME	UtcTime	Datum und Uhrzeit, Wertebereich ab 01.01.1990; nur für S7-1500 verfügbar
ldt	LDT	UtcTime	Datum und Uhrzeit nanosekundengenau, Wertebereich ab 01.01.1990; nur für S7-1500 verfügbar
dtl	DTL	UtcTime	Datum und Uhrzeit nanosekundengenau, Wertebereich 01.01.1970 – 31.12.2553; nur für S7-1500 verfügbar
date	DATE	UtcTime	Datum und Uhrzeit (8 Byte), wobei die Uhrzeit immer 00:00:00 ist, Wertebereich ab 01.01.1990. Abbildung des CPU-Datentyps "DATE" (unsigned, 16 Bit).
t	TIME	Int32	Vorzeichenbehafteter Zeitwert in Millisekunden
lt	LTIME	Int64	Vorzeichenbehafteter Zeitwert in Nanosekunden; nur für S7-1500 verfügbar
tod	TOD	UInt32	Tageszeit, 0 ... 86399999 ms ab Mitternacht

Typ-Bezeichner	S7-Datentyp	OPC-UA-Datentyp	Beschreibung
Itod	LTOD	UInt64	Tageszeit, 0 ... 86399999999999 ns ab Mitternacht; nur für S7-1500 verfügbar
s5bcd	S5TIME	UInt16	Abbildung des CPU-Datentyps "S5TIME" auf UInt 16 (unsigned, 16 Bit) mit eingeschränktem Wertebereich, 0...9990000 ms.*); nur für S7-1500 verfügbar
x<Bitadresse>	BOOL	Boolean	Bit (bool) Zusätzlich zum Byte-Offset im Bereich ist noch die <Bitadresse> im jeweiligen Byte anzugeben. Wertebereich 0 ... 7
s<Stringlänge>	STRING	String	Es ist noch die für den String reservierte <Stringlänge> anzugeben. Wertebereich 1 ... 254 Beim Schreiben können auch kürzere Strings geschrieben werden, wobei die übertragene Datenlänge immer die reservierte Stringlänge in Byte zuzüglich 2 Byte ist. Die nicht benötigten Bytes werden mit dem Wert 0 gefüllt. Das Lesen und Schreiben von Strings und String-Arrays wird intern auf das Lesen und Schreiben von Byte-Arrays abgebildet. Der String muss auf der S7 mit gültigen Werten initialisiert sein.

\*) Siehe nachfolgende Tabelle "Zeitraster und Wertebereich für den S7-Datentyp "s5bcd""

#### Hinweis

Der Typbezeichner "<b>" wird vom S7OPT-OPC-UA-Server als Defaultwert verwendet, falls kein "<S7OPTTyp>" angegeben ist.

#### Hinweis

Auf den S7-Datentyp "DTL" kann vom SIMATIC NET OPC-UA-Server nur lesend zugegriffen werden.

#### Hinweis

Der S7-Datentyp "Byte[]" wird vom SIMATIC NET OPC-UA-Server auf einen Bytestring abgebildet.

**Zeitraster und Wertebereich für den S7-Datentyp "s5tbcd":**

Der Wertebereich der Zeit-Variable vom Datentyp "s5tbcd" ist BCD-kodiert. Der Wertebereich ergibt sich entsprechend der folgenden Tabelle:

Bit-Nr.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bedeutungssymbol	0	0	x	x	z	z	z	z	z	z	z	z	z	z	z	z
<b>Erläuterung:</b>																
Bedeutungssymbol "0"	nicht relevant															
Bedeutungssymbol "x"	Angabe des Zeitrasters															
	Bit 13 und 12										Zeitraster in Millisekunden					
	00										10					
	01										100					
	10										1000					
	11										10000					
Bedeutungssymbol "z"	BCD-kodierter Zeitwert (0...999)															

**Beispiel:**

Bit-Nr.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Wert	0	0	0	1	0	0	0	0	0	1	1	1	0	1	0	1

Bit 0-11 legen die Zahl 075 fest. Bit 12 und 13 legen das Zeitraster 100 ms fest.  
 $75 * 100 \text{ ms} = 7500 \text{ ms}$ . Dieser taktet 75 mal in 100-ms-Stufen herunter.

Der OPC-Datentyp der Zeit-Variable vom Datentyp "s5tbcd" ist ein Wort (UInt16). Beim Schreiben ist der Wertebereich entsprechend eingeschränkt.

**<S7OPTTimerTyp>**

Ein S7-Timer Datentyp wird im OPC-UA-Server in den entsprechenden OPC-UA-Datentyp umgewandelt. Die folgende Tabelle listet den Typ-Bezeichner und den entsprechenden OPC-UA-Datentyp auf, in dem der Variablenwert dargestellt werden kann.

Typ-Bezeichner	S7-Datentyp	OPC-UA-Datentyp	Beschreibung
tbcd	TIMER	UInt16	BCD-codiert Wird als Defaultwert verwendet, falls kein <S7TimerTyp> angegeben ist.
tda	TIMER	UInt16[2]	Dezimale Zeitbasis und Zeitwert

**Zeitraster und Wertebereich von S7OPTTimerObjekt "t" und S7OPTTimerTyp "tbcd":**

Der Wertebereich einer S7-Timer-Variablen vom Typ "tbcd" ist BCD-kodiert. Aus dem Wertebereich ergibt sich (für das Schreiben) das Zeitraster entsprechend der folgenden Tabelle:

Bit-Nr.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bedeutungssymbol	0	0	x	x	z	z	z	z	z	z	z	z	z	z	z	z
<b>Erläuterung:</b>																
Bedeutungssymbol "0"	nicht relevant															
Bedeutungssymbol "x"	Angabe des Zeitrasters															
	Bit 13 und 12										Zeitraster in Millisekunden					
	00										10					
	01										100					
	10										1000					
	11										10000					
Bedeutungssymbol "z"	BCD-kodierter Zeitwert (0 ... 999)															

**Beispiel:**

Bit-Nr.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Wert	0	0	0	1	0	0	0	0	0	1	1	1	0	1	0	1

Bit 0-11 legen die Zahl 075 fest. Bit 12 und 13 legen das Zeitraster 100 fest.  
 $75 * 100 = 7500$  ms. Dieser taktet 75 mal in 100-ms-Stufen herunter.

**Zeitraster und Wertebereich für den S7-Timer-Datentyp "tda":**

Datentyp: Feld von zwei Worten {Zeitraster in Millisekunden als UInt16 | Zeitwert UInt16}

Zeitraster [ms]	10, 100, 1 000, 10 000
Zeitwert	0 ... 999
Zeitbereiche [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

Bei dem Objekt "TDA" ist keine Eingabe der <Anzahl> möglich.

**Beispiel:**

Beschreiben des Timers "t.3,tda" mit Wert {100|50} initialisiert den Timer 3 mit dem Wert  $50 * 100$  ms = 5000 ms und dieser taktet 50 mal in 100-ms-Stufen herunter.

### <S7OPTCounterTyp>

Typ-Bezeichner	S7-Datentyp	OPC-UA-Datentyp	Beschreibung
c	COUNTER	UInt16	Wertebereich für S7: 0...999, decimal kodiert. Wird als Defaultwert verwendet, falls kein <S7OPTCounterTyp> angegeben ist.

### <Anzahl>

Anzahl der Variablen eines Typs, die ab dem im Parameter "Adresse" angegebenen Offset angesprochen werden sollen (Wertebereich 1...65 535).

Die Angabe einer Anzahl von Arrayelementen führt immer zur Bildung eines Feldes vom entsprechenden Typ, auch wenn nur ein einziges Feldelement adressiert wird.

Das Trennzeichen ist ein Komma (",").

Beim Datentyp "x" ist die Eingabe der Anzahl für Schreibzugriff nur in Vielfachen von 8 möglich. Die Bit-Adresse muss dann Null sein.

Beim Datentyp "x" ist die Eingabe der Anzahl für Lesezugriff nicht eingeschränkt. Der Wertebereich der Bit-Adresse erlaubt dann 0...7.

#### Beispiele:

Namensraum-URI: S7OPT: (Namensraum-Index: 3)

S7-OPC-1.db1.10,x0,64, Zugriffsrechte RW

S7-OPC-1.db1.10,x3,17, Zugriffsrechte R

## Beispiele für Prozessvariablen für nicht-optimierte S7OPT-OPC-UA-Variablen-Dienste

Hier finden Sie Beispiele, die die Syntax von Variablennamen für Variablen-Dienste verdeutlichen.

### Nicht-optimierter Datenbaustein DB Einzel-Byte

*Namensraum-URI: S7OPT: (Namensraum-Index: 3)*

*S7-Verbindung-1.db5.12,b*

bezeichnet Datenbyte 12 im Datenbaustein 5 über S7-Verbindung-1.

### Nicht-optimierter Datenbaustein DB, Feld von Wörtern

*Namensraum-URI: S7OPT: (Namensraum-Index: 3)*

*S7-Verbindung-1.db5.10,w,9*

bezeichnet 9 Datenwörter ab Byteadresse 10 im Datenbaustein 5 über S7-Verbindung-1.

### Nicht-optimierter Datenbaustein DB, Feld von Strings

*Namensraum-URI: S7OPT: (Namensraum-Index: 3)*

*S7-Verbindung-1.db100.50,s32,3*

bezeichnet 3 Strings der Länge 32 ab Byteadresse 50 im Datenbaustein 100 über S7-Verbindung-1.

### Eingang 0

*Namensraum-URI: S7OPT: (Namensraum-Index: 3)*

*S7-Verbindung-1.i.0*

bezeichnet das Eingangsbyte 0 über S7-Verbindung-1.

### Ausgang 0 Bit 0

*Namensraum-URI: S7OPT: (Namensraum-Index: 3)*

*S7-Verbindung-1.q.0,x0*

bezeichnet in Ausgangsadresse 0 das Bit 0 über S7-Verbindung-1.

### Feld von lesbaren Merkerbits

*Namensraum-URI: S7OPT: (Namensraum-Index: 3)*

*S7-Verbindung-1.m.3,x4,12 // Zugriffsrechte R*

bezeichnet 12 Bits ab Merkeradresse 3 und dort ab Bitadresse 4 über S7-Verbindung-1. Nur lesbar.

### Timer 22 BCD-kodiert

*Namensraum-URI: S7OPT: (Namensraum-Index: 3)*

*S7-Verbindung-1.t.22*

bezeichnet Timer 22, TBCD-Default über S7-Verbindung-1.

## 2.8.12.2 Optimierter Zugriff

### Was machen Variabldienste, die über den optimierten Zugriff aufgerufen werden?

Variabldienste, die über den optimierten Zugriff aufgerufen werden, ermöglichen den optimierten Zugriff und die Beobachtung von S7-Variablen im Automatisierungsgerät. Die Adressierung der S7-Variablen erfolgt symbolisch. Die Art des Zugriffs orientiert sich an der Notation der S7-Symbolik.

## Objekte im Automatisierungsgerät

Der S7OPT-OPC-UA-Server unterstützt folgende Objekte:

- Datenbausteine (optimierter Zugriff)
- Instanzdatenbausteine (optimierter Zugriff)

---

### Hinweis

Die optimierte S7-Kommunikation mit OPC UA funktioniert mit den S7-1200- (ab V4.0) und den S7-1500-Stationen.

---

## Syntax der optimierten Variabeldienste

### Syntax der symbolischen Adressierung

Vereinfachte Syntax der Prozessvariablen der S7OPT-OPC-UA-Nodeld:

*Namensraum-URI: SYM: (Namensraum-Index: 4)*

### Syntax Symbolik

<Stationsname>. <CPU-Name> { .<Ordner> } .<Symbol>

### Erklärungen

Namensraum-URI: SYM: (Namensraum-Index: 4)

<Stationsname>

Der Stationsname wird bei der Projektierung in STEP 7 Professional (TIA Portal) festgelegt. Das folgende Trennzeichen ist der Punkt (".").

<CPU-Name>

Der Name der S7-CPU wird in STEP 7 Professional (TIA Portal) festgelegt. Das folgende Trennzeichen ist der Punkt (".").

<Ordner>

Name des Datenbausteines oder einer Datenstruktur in der S7-CPU. Der Name wird in STEP 7 Professional (TIA Portal) festgelegt. Das folgende Trennzeichen ist der Punkt ("."). Bei Datenstrukturen in Datenbausteinen erweitert sich die Ordnerstruktur entsprechend.

<Symbol>

Name der Variablen in der S7-CPU. Der Variablenname wird in STEP 7 Professional (TIA Portal) im Datenbaustein oder in der Symboltabelle festgelegt.

S7-Datentyp	OPC-UA-Datentyp	Beschreibung
BYTE	Byte	8 Bit (Bitstring)
USINT	Byte	8 Bit (unsigned)
CHAR	SByte	8 Bit (Charakter)
SINT	SByte	8 Bit (signed)

S7-Datentyp	OPC-UA-Datentyp	Beschreibung
WORD	UInt16	16 Bit (Bitstring)
UINT	UInt16	16 Bit (unsigned)
DWORD	UInt32	32 Bit (Bitstring)
UDINT	UInt32	32 Bit (unsigned)
LWORD	UInt64	64 Bit (Bitstring); nur für S7-1500 verfügbar
ULINT	UInt64	64 Bit (unsigned); nur für S7-1500 verfügbar
INT	Int16	16 Bit (signed)
DINT	Int32	32 Bit (signed)
LINT	Int64	64 Bit (signed); nur für S7-1500 verfügbar
REAL	Float	Fließkomma (4 Byte) IEEE 754
LREAL	Double	Fließkomma (8 Byte) IEEE 754
DATE_TIME	UtcTime	Datum und Uhrzeit, Wertebereich ab 01.01.1990; nur für S7-1500 verfügbar
LDT	UtcTime	Datum und Uhrzeit nanosekundengenau, Wertebereich ab 01.01.1990; nur für S7-1500 verfügbar
DTL	UtcTime	Datum und Uhrzeit nanosekundengenau, Wertebereich 01.01.1970 – 31.12.2553; nur für S7-1500 verfügbar
DATE	UtcTime	Datum und Uhrzeit (8 Byte), wobei die Uhrzeit immer 00:00:00 ist, Wertebereich ab 01.01.1990. Abbildung des CPU-Datentyps "DATE" (un-signed, 16 Bit).
TIME	Int32	Vorzeichenbehafteter Zeitwert in Millisekunden
LTIME	Int64	Vorzeichenbehafteter Zeitwert in Nanosekunden; nur für S7-1500 verfügbar
TOD	UInt32	Tageszeit, 0 ... 86399999 ms ab Mitternacht
LTOD	UInt64	Tageszeit, 0 ... 86399999999999 ns ab Mitternacht; nur für S7-1500 verfügbar
S5TIME	UInt16	Abbildung des CPU-Datentyps "S5TIME" auf UInt 16 (unsigned, 16 Bit) mit eingeschränktem Wertebereich, 0...9990000 ms.*); nur für S7-1500 verfügbar

S7-Datentyp	OPC-UA-Datentyp	Beschreibung
BOOL	Boolean	<p>Bit (bool)</p> <p>Zusätzlich zum Byte-Offset im Bereich ist noch die &lt;Bitadresse&gt; im jeweiligen Byte anzugeben.</p> <p>Wertebereich 0 ... 7</p>
STRING	String	<p>Es ist noch die für den String reservierte &lt;Stringlänge&gt; anzugeben.</p> <p>Wertebereich 1 ... 254</p> <p>Beim Schreiben können auch kürzere Strings geschrieben werden, wobei die übertragene Datenlänge immer die reservierte Stringlänge in Byte zuzüglich 2 Byte ist. Die nicht benötigten Bytes werden mit dem Wert 0 gefüllt.</p> <p>Das Lesen und Schreiben von Strings und String-Arrays wird intern auf das Lesen und Schreiben von Byte-Arrays abgebildet.</p> <p>Der String muss auf der S7 mit gültigen Werten initialisiert sein.</p>

## Hinweis

Auf den S7-Datentyp "DTL" kann vom SIMATIC NET OPC-UA-Server nur lesend zugegriffen werden.

## Hinweis

Bei anwenderdefinierte Datentypen (UDT) unterstützt der SIMATIC NET OPC-UA-Server nur den Zugriff auf die Unterelemente des UDTs. Auf den UDT als Ganzes kann nicht zugegriffen werden.

## Hinweis

Der S7-Datentyp "Byte[]" wird vom SIMATIC NET OPC-UA-Server auf einen Bytestring abgebildet.

Beispiel für eine Prozessvariable für optimierte S7OPT-OPC-UA-Variablen-Dienste

Hier finden Sie ein Beispiel, das die Syntax von Variablennamen für Variabldienste verdeutlicht.

Optimierter Datenbaustein DB Einzel-Byte

Namensraum-URI: SYM: (Namensraum-Index: 4)

*S7-1500-Station 1.S7-1500.datatypes optimized.byte*

bezeichnet eine S7-Datenvariable "byte" im Datenbaustein "datatypes\_optimized" in der CPU "S7-1500" der Station "S7-1500-Station 1".

Die Syntax-Festlegung trifft der Anwender innerhalb seines STEP 7-Professional-Projektes durch Vergabe von eindeutigen Namen für Stationen, CPUs, Datenbausteinen und Variablen. Die Auswahl der im OPC-Adressraum vorhandenen Symbole erfolgt über das Dialogfeld "Symbol-Konfiguration" des OPC-Servers im TIA Portal.

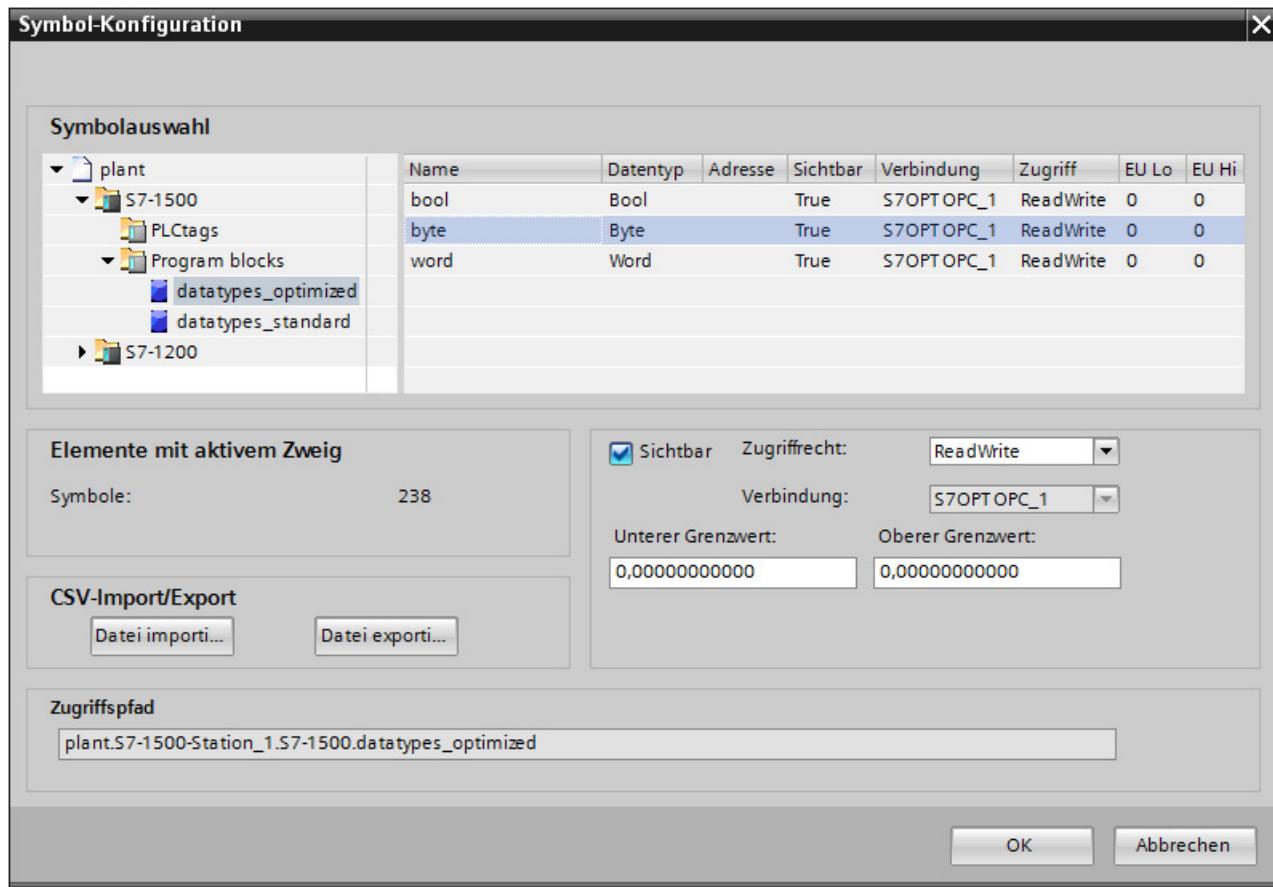


Bild 2-44 Symbol-Konfiguration des OPC-Servers in STEP 7 Professional (TIA Portal)

## 2.8.13 Baustein-Informations-Objekte einer S7-Verbindung

### 2.8.13.1 Längeninformationen

#### Die Bausteinobjekte unter einem S7OPT-Verbindungsobjekt

Die Art und Größe des Baustinaufbaus in einem S7-Automatisierungsgerät wird zur Laufzeit ermittelt. Die Ermittlung dieser Detailinformationen ist nur für die nicht-optimierten Bausteine der S7-1200- / S7-1500-Station möglich. Es gibt folgende Bausteinobjekte unter einem S7-Verbindungsobjekt, die diese Informationen einem OPC-UA-Client zur Verfügung stellen:

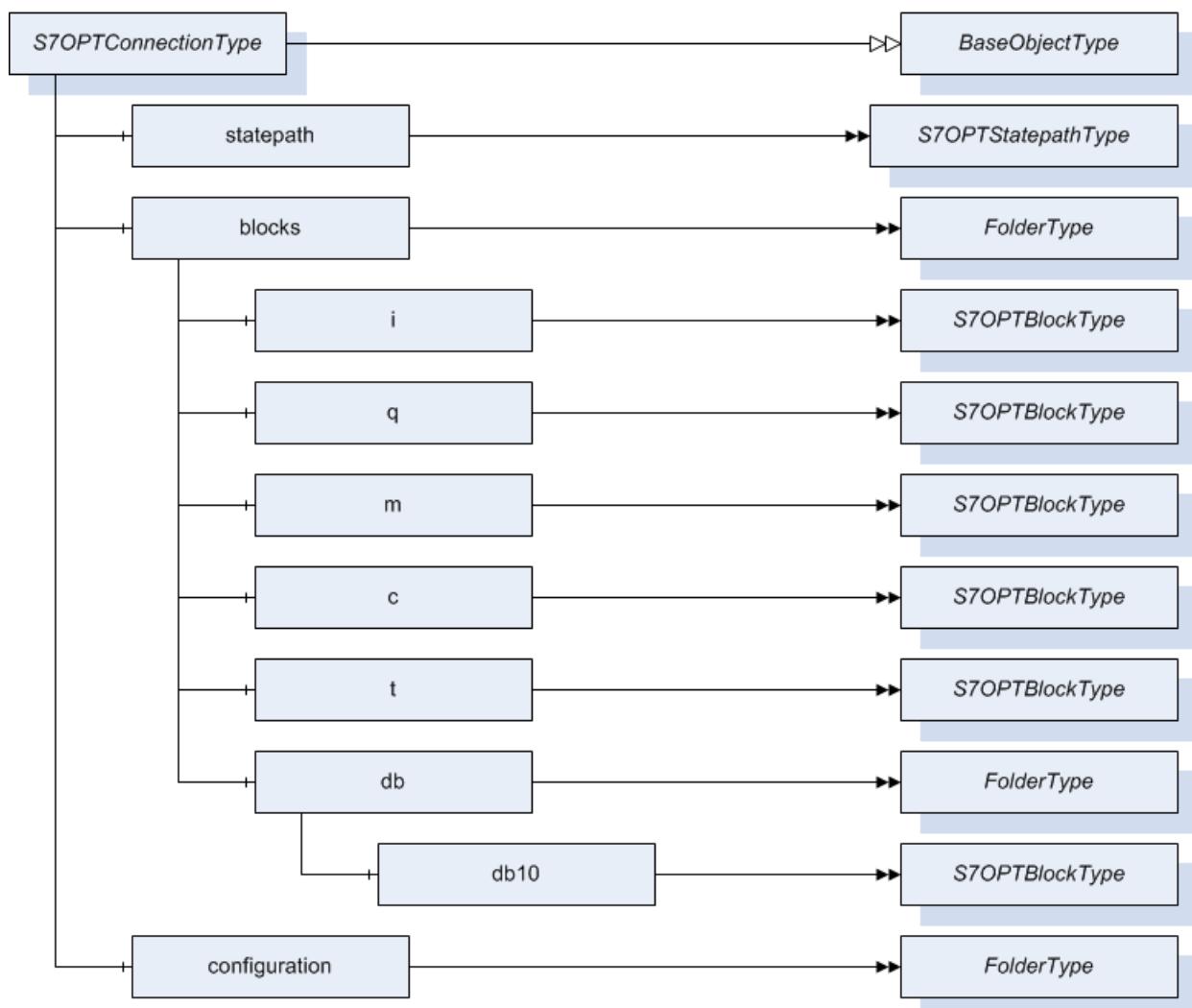


Bild 2-45 Bausteinobjekte unter einem S7-Verbindungsobjekt

Jedes Bausteinobjekt enthält dabei eine OPC-UA-Property mit einer Längen- bzw. Größenangabe über den Baustein (length).

**Beispiel:**

Namensraum-URI: S7OPT: (-->Namensraum-Index: 3)

S7-Verbindung\_1.db10.length //Property-Länge des DB10-Bausteins (Länge in Byte)

**Hinweis**

Das Längeninformationsfeld steht Ihnen bei Datenbausteinen zur Verfügung, die über den Standardzugriff aufgerufen werden.

## 2.8.13.2 Musterobjekte

### Das Musterobjekt eines Bausteinobjekts

Für jedes beim Durchsuchen angezeigte Bausteinobjekt wird eine Musterdatenvariable generiert, dessen Nodeld als Vorlage für weitere benutzerdefinierte Datenobjekte verwendet werden kann. Die Musterdatenvariable besitzt den Standard-Datentyp B (bzw. c oder tbcd) für das jeweilige Bausteinobjekt und beginnt immer ab Adresse 0 (bei Datenbausteinen für den Standardzugriff). Sollte dieses Item beim Verbindungspartner nicht zugreifbar sein, dann wird dies über entsprechende Zugriffsergebnisse und Quality-Codes angezeigt.

**Beispiel:**

Namensraum-URI: S7OPT: (Namensraum-Index: 3)

S7-Verbindung\_1.db10.0,b

S7-Verbindung\_1.m.0,b

## 2.8.13.3 Diagnose- und Konfigurationsinformationen

### Die Properties eines S7OPT-Verbindungsobjekts

Im Allgemeinen werden die Eigenschaften einer S7-Verbindung mit dem Projektierungswerkzeug STEP 7 Professional (TIA Portal) projektiert. Zur Laufzeit kann es sinnvoll sein, einige Projektierungsparameter auszuwerten.

Einige Projektierungsparameter werden für OPC UA als Properties zum S7OPT-Verbindungsobjekt bereitgestellt:

### Syntax zu Diagnose- und Konfiguration-Informationen

Namensraum-URI: S7OPT: (-->Namensraum-Index: 3)

<Verbindungsname>. <S7OPTVerbindungsProperty>

<S7OPTVerbindungsProperty>:= "vfd"|"cp"|"remoteaddress"|"connect"|"modelversionid"|"fastconnectionstatereturnenable"|"connecttimeout"|"timeout"|"abortconnectionafter"|"optimizebuffer"|"defaultalarmseverity"

S7-Verbindungsdiagnose-Property (nur lesbar)	Beschreibung				
vfd	Name des OPC-Servers, dem die Verbindung zugeordnet ist. Üblicherweise hat dieses bei über TIA Portal projektierte STEP 7-Verbindungen den Text "OPC Server_1". Datentyp String, nur lesbar.				
cp	Name der Schnittstellenparametrierung, dem die Verbindung zugeordnet ist. Datentyp String, nur lesbar.				
remoteaddress	Adresse des Verbindungspartners. Datentyp String, nur lesbar. Die Adresse des Verbindungspartners ist ein Datenpuffer mit einer vom Verbindungstyp abhängigen Datenlänge. Für die übersichtlichere Auswertung durch den Anwender wird der Datenpuffer formatiert in einem String dargestellt. IP-Adresse (ISOonTCP) Format: "ddd.ddd.ddd.ddd" (je 1-3 dezimale Ziffern) oder über die MAC-Adresse				
modelversionid	Typ und Version des Verbindungspartner Datentyp UInt32,nur lesbar				
connect	Art des Verbindungsaufbaus. Datentyp UInt32, nur lesbar. <table border="1" data-bbox="921 1354 1429 1503"> <tr> <td>1</td><td>Aktiv, Verbindungsaufbau wird erst bei Bedarf hergestellt, Verbindungsabbau ohne Benutzung nach Wartezeit.</td></tr> <tr> <td>2</td><td>Aktiv, Verbindung wird permanent aufrecht erhalten.</td></tr> </table>	1	Aktiv, Verbindungsaufbau wird erst bei Bedarf hergestellt, Verbindungsabbau ohne Benutzung nach Wartezeit.	2	Aktiv, Verbindung wird permanent aufrecht erhalten.
1	Aktiv, Verbindungsaufbau wird erst bei Bedarf hergestellt, Verbindungsabbau ohne Benutzung nach Wartezeit.				
2	Aktiv, Verbindung wird permanent aufrecht erhalten.				
fastconnectionstatereturnable	Schnelle Rückgabe eines Schreib-/Lesezugriffs bei unterbrochener Verbindung. Datentyp Boolean, Lesen und Schreiben. True: aktiviert False: deaktiviert				
connecttimeout	Verbindungsaufbau-Timeout in ms Datentyp UInt32, Lesen und Schreiben. 0:kein Timeout >0:Timeout in ms				

S7-Verbindungsdiagnose-Property (nur lesbar)	Beschreibung
timeout	Auftrags-Timeout für den Produktivverkehr in ms. Datentyp UInt32, Lesen und Schreiben. 0:kein Timeout >0:Timeout in ms
abortconnectionafter	Automatischer Verbindungsabbau. Verzögerungszeit für den automatischen Verbindungsabbau: Der OPC-Server baut die Verbindung nach dieser Zeit selbstständig wieder ab, sofern in dieser Zeit kein erneuter Variablenzugriff erfolgt. Auf diese Weise können bei Zugriffen auf Variablen in sehr großen Zeitabständen die Anzahl der benötigten Verbindungen reduziert werden. Datentyp UInt32, Lesen und Schreiben. 0:kein Abbau >0:Leerlaufzeit bis zum Abbau in ms
optimizebuffer	Größe des Optimierungspuffers für die Baustein-Kommunikation zu Datenbausteinen über den Standardzugriff. Datentyp UInt16, Lesen und Schreiben. 0: Keine Optimierung >0: Grösse des Optimierungspuffers in Bytes
defaultalarmseverity	Vorgabe-Priorität für den Statepath Alarm. Datentyp UInt16, Lesen und Schreiben. 1:niederprior ... 1000:hochprior

## 2.8.14 S7OPT-OPC-UA-Template-Datenvariablen

Sie haben mit den Prozessvariablen für das OPC-UA-S7OPT-Protokoll flexible Einstellungsmöglichkeiten, um die „nicht optimierten“ Prozessdaten Ihrer Anlage in den gewünschten Datenformaten zu erhalten.

Die Vielfalt der Adressierungsmöglichkeiten lässt sich allerdings nicht in einen vollständig durchsuchbaren Namensraum fassen. Bereits ein über Standardzugriff aufgerufener Datenbaustein, mit der Länge eines einzelnen Bytes besitzt etwa 40 verschiedene Datenformatoptionen – angefangen vom Byte, SByte, Felder mit einem Element davon, einzelne Bits, Felder von Bits mit bis zu 8 Feldelementen an unterschiedlichen Bitoffsets beginnend.

Der OPC-UA-Server unterstützt den Anwender deshalb mit den sogenannten Template-Datenvariablen im S7OPT-Namensraum. In einem für einen OPC-Client typischen Texteingabefeld können diese Templates durch Ändern einiger weniger Zeichen in gültige ItemIDs verwandelt werden.

### Beispiel:

S7-Verbindung1.db<db>.<o>, dw

Durch Ersetzen von <db> mit der Datenbausteinnummer und <o> dem Offset innerhalb des Datenbausteins erhalten Sie eine gültige Nodeld.

-> S7-Verbindung1.db10.4,dw

Der Vorteil dieses Konzepts ist, dass es von nahezu allen OPC-UA-Clients eingesetzt werden kann, ohne dass Anpassungen der Clients erforderlich sind.

#### Hinweis

Die Verwendbarkeit von OPC-UA-S7OPT-Template-Datenvariablen kann im Konfigurationsprogramm "Kommunikations-Einstellungen" unter "OPC-Protokollauswahl" > Klicken des Pfeilsymbols bei "S7 optimiert" aktiviert und deaktiviert werden.

### Template-Datenvariablen innerhalb der Browse Hierarchie

Die Template-Datenvariablen sind neben den ihnen entsprechenden Ordnern in der Namensraum-Darstellung eingesortiert, so dass sie bei Bedarf leicht genutzt werden können.

### Spezielle Nutzung einiger Attribute der Template-Datenvariablen

Die Verwendung der OPC-UA-Attribute ist durch die UA-Spezifikation vorgegeben und bedarf keiner weiteren Erläuterung.

Beispiel für das Template einer Datenbaustein-Bytevariablen:

Nodeld:	S7-Verbindung1.db<db>.<o>,b
Browse-Name:	Template Byte
Beschreibung:	<db>Adresse des Datenbausteins
	<o> offset innerhalb des Datenbausteins

## 2.8.15 OPC-UA-Events, -Conditions und -Alarne

### 2.8.15.1 Welche OPC-UA-Alarne gibt es?

Folgende OPC-UA-Event- und Alarmtypen werden vom S7OPT-OPC-UA-Server unterstützt:

- Statepath-Alarm  
Meldungen zum S7-Verbindungsstatus
- Consistency-Alarm  
Meldung zu einer inkonsistenten Alarmprojektierung
- Programmalarne (Bausteinbezogene PLC-Meldungen)  
Der Meldebaustein „Program\_Alarm“ generiert aus einem Signalwechsel eine PLC-Meldung mit oder ohne Quittierpflicht mit bis zu zehn Begleitwerten.

---

**Hinweis**

Programmmeldungen werden vom S7OPT-OPC-UA-Server nur zur S7-1500 unterstützt.

---

**Hinweis**

Bevor Programmmeldungen ausgegeben werden können, müssen sie in SIMATIC STEP 7 Professional (TIA Portal) projektiert werden. Sie werden im Programmiereditor erstellt und im Meldungseditor bearbeitet. Siehe Systemhandbuch „SIMATIC STEP 7 Professional“ bzw. Informationssystem in SIMATIC STEP 7 Professional (TIA Portal).

---

### **2.8.15.2 Was sind OPC-UA-Events?**

Eine Anlage ist charakterisiert durch die Stati seiner Hardware- und Software-Komponenten. UA-Alarming bietet die Möglichkeit, Statusänderungen einer Auswahl von Stati dem dafür angemeldeten Anwender als Ereignisse zu melden. Die Informationen des Events sind in seinen Properties abgelegt. Welche Properties ein Event aufweist, wird durch den Eventtyp definiert.

Der Eventtyp weist eigene oder von einem anderen Eventtyp (der seinerseits vererbte Properties aufweisen kann) vererbte Properties auf. Die Möglichkeit der einfachen Vererbung führt zu einer Eventtypenhierarchie. Die UA-Alarming-Spezifikation weist eine Vielzahl von vordefinierten Eventtypen auf, die in der vordefinierten Typenhierarchie strukturiert sind. Des Weiteren macht die UA-Alarming-Spezifikation Angaben zum Typ der Properties und zu der Semantik dieser Properties. Alle vordefinierten Events befinden sich im Namensraum ns="http://opcfoundation.org/UA/".

### **2.8.15.3 Welche S7OPT-Eventtypen werden vom S7OPT-OPC-UA-Server unterstützt?**

Der S7OPT-OPC-UA-Server lehnt sich an vordefinierten Eventtypen an und leitet die eigenen Eventtypen von den vordefinierten Eventtypen ab. Es sind für S7OPT-Events eigene Eventtypen definiert. Alle S7OPT-Eventtypen befinden sich im Namensraum ns="S7OPTTYPES:".

S7OPT-UA-Alarming dient zur Darstellung von PLC-Meldungen. Die untenstehende Tabelle gibt an, welche Eventtypen der S7OPT-OPC-UA-Server meldet. Mit Ausnahme der ersten beiden S7OPT-Eventtypen, treten Events mit dem angegebenen Eventtyp erst nach dem Empfang einer projektierten Programmmeldung auf.

Nodeld und Anzeigenamen des S7OPT-Eventtypen	Bedeutung des S7OPT-Eventtypen
ns= S7OPTTYPES; i=14 "S7OPTStatepathAlarmType"	"S7OPTStatepathAlarmType" bildet den inversen Zustand einer S7-Verbindung ab ("Inaktiv", wenn S7-Verbindung aufgebaut ("UP") ist; und "Aktiv", wenn S7-Verbindung nicht aufgebaut ist ("Down")). Der Zustand wird nur PC-seitig ermittelt und kann hierdurch auch dann gemeldet werden, wenn keine physikalische Verbindung zur PLC besteht.
ns= S7OPTTYPES; i=15 "S7OPTConsistencyAlarmType"	"S7OPTConsistencyAlarmType" zeigt eine inkonsistente Alarmprojektierung an.
ns= S7OPTTYPES; i=43 "S7OPTOffNormalAlarmType"	Projektierte Programmmeldungen mit oder ohne Quittierpflicht. Die Programmmeldungen diesen Typs haben einen Zustand bzw. Bedingung. Diese können den Status „kommend“ oder „gehend“ annehmen.
ns= S7OPTTYPES; i=61 "S7OPTInfoReportEventType"	Projektierte Programmmeldungen, die nur zu Informationszwecken dienen. Hierzu muss das Optionskästchen in der Spalte "Nur Information" im Meldungseditor von SIMATIC STEP 7 Professional (TIA Portal) aktiviert sein.

#### 2.8.15.4 Eventtypenhierarchie von S7OPT-OPC-UA-Server

Die Eventtypenhierarchie des S7OPT-OPC-UA-Server besteht aus der Standard-Eventtypenhierarchie, wobei aus manchen Eventtypen S7OPT-Eventtypen abgeleitet werden.

Die Eventtypenhierarchie kann mit dem OPC Scout V10 durchsucht werden.

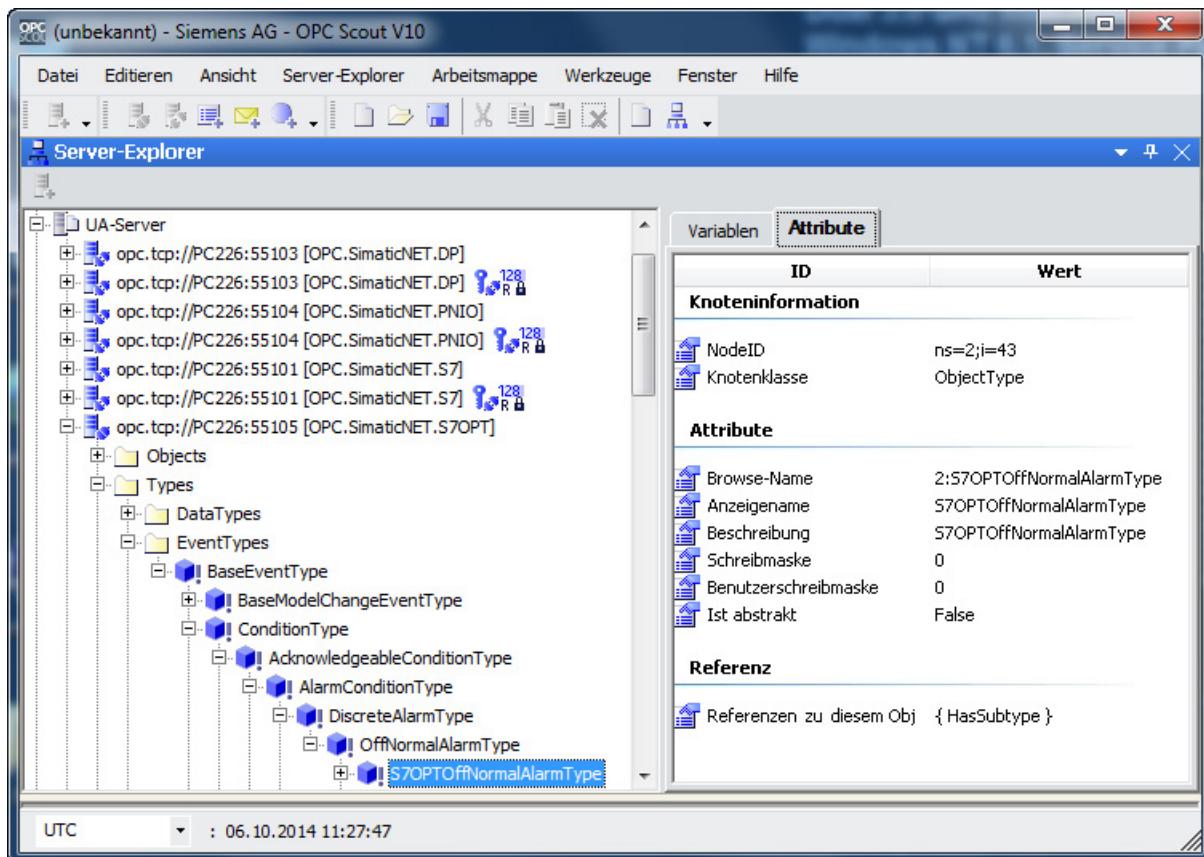


Bild 2-46 Abbildung der Standard-Eventtypenhierarchie

Die Vererbung der Standard-Eventtypen ist wie folgt:

Standard-Eventtyp mit Anzeigenamen	erbt von ...
"ConditionType"	"BaseEventType"
"AcknowledgeableConditionType"	"ConditionType"
"AlarmConditionType"	"AcknowledgeableConditionType"
"DiscreteAlarmType"	"AlarmConditionType"
"OffNormalAlarmType"	"DiscreteAlarmType"
"SystemOffNormalAlarmType"	"OffNormalAlarmType"

Der Standard-Eventtyp mit dem Anzeigenamen "BaseEventType" ist der Typ von dem alle Eventtypen abgeleitet sind. Dieser Typ definiert alle Properties, die in allen Events verwendet werden sowie auch deren Verhalten.

Ein Eventtyp hat eine numerische NodId (z. B. ns="http://opcfoundation.org/UA/", i=2041 und Anzeigename ="BaseEventType").

Die Vererbung der S7OPT-Eventtypen von den Standard-Eventtypen ist wie folgt:

S7OPT-Eventtyp mit Anzeigenamen	erbt von ...
"S7OPTStatepathAlarmType"	"SystemOffNormalAlarmType"
"S7OPTConsistencyAlarmType"	"SystemOffNormalAlarmType"

S7OPT-Eventtyp mit Anzeigenamen	erbt von ...
"S7OPTOffNormalAlarmType"	"OffNormalAlarmType"
"S7OPTInfoReportEventType"	"BaseEventType"

Beim S7OPT-UA-Alarming werden die S7OPT-Eventtypen "S7OPTStatepathAlarmType" , "S7OPTConsistencyAlarmType" und "S7OPTOffNormalAlarmType" instanziert. Eine PLC-Meldung ist deshalb mit all seinen Properties in einem Alarmobjekt abgebildet. Die Properties der jeweiligen Instanz können hilfsweise auch über Data Access-Zugriffe gelesen oder beobachtet werden.

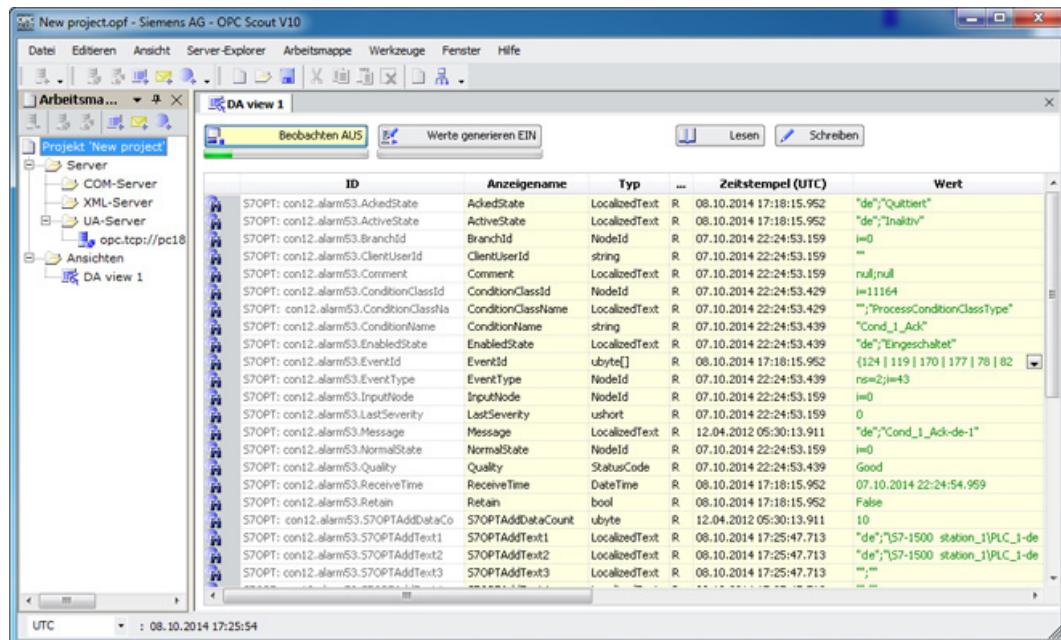


Bild 2-47 Beobachten der Alarmobjekte über Data Access

In den Kapiteln "S7OPT-Eventtypen (Seite 272)" und "Bereichsbau und Herkunftsraum (Seite 278)" werden die für den Anwender bei der Programmierung einer UA-Applikation relevanten Properties beschrieben. Sie sind gruppiert nach den Eventtypen, in denen sie definiert werden. Bei der Referenzierung einer Property verwendet man den Datentyp "QualifiedName". "QualifiedName" beinhaltet einen Namensraum und einen Browse-Namen (gelegentlich Browse-Pfad).

Da jedoch im Kontext des Dokumentes nur Properties referenziert werden, deren "QualifiedName" einen Namensraum haben, der identisch mit dem Namensraum des definierenden Typen ist, wird auf die Angabe des Namensraums verzichtet. Statt "QualifiedName" wird der Browse-Name des Properties angegeben. Zusätzlich wird in Klammern das angeforderte Attribut getrennt durch ":" von dem Datentyp angegeben. In den meisten Fällen ist das angeforderte Attribut numerisch 13, was den "Value" angibt. Gelegentlich tritt der Wert numerisch 1 auf, was die Nodeld angibt.

## 2.8.16 Standard-Eventtypen und die Verwendung ihrer Properties

### 2.8.16.1 Standard-Eventtyp mit dem Anzeigenamen "BaseEventType"

Nodeld: ns="http://opcfoundation.org/UA/", i=2041

Abgeleitet von: ist von keinem anderen Eventtyp abgeleitet.

Browse-Namen der relevanten Properties

- "EventId" (13|ByteString)
- "EventType" (13|Nodeld)
- "SourceNode" (13|Nodeld)
- "SourceName" (13|String)
- "Time" (13|DateTime)
- "ReceiveTime" (13|DateTime)
- "Message" (13|LocalizedText)
- "Severity" (13|UInt16)

Direkt von diesem Typen abgeleitete S7OPT-Eventtypen: ns="S7OPTTYPES:", i =61, Anzeigenamen="S7OPTInfoReportEventType".

Ein Event von diesem Eventtyp wird eindeutig durch den SourceName identifiziert.

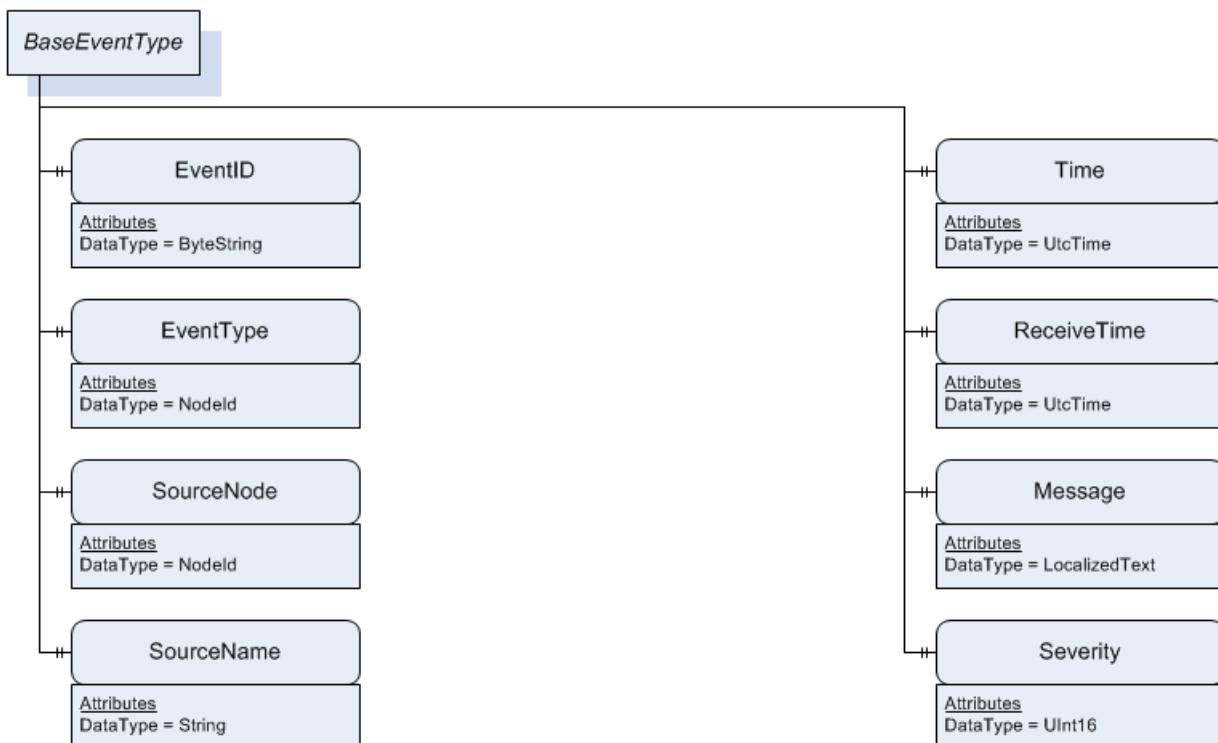


Bild 2-48 Darstellung des BaseEventType

#### EventId

Eine Kennung, welche einen Event eindeutig bestimmt und referenziert. Der Client benötigt die EventId z.B. zur Quittierung von Alarmen.

## EventType

ist immer einer der bereits aufgelisteten S7OPT-Eventtypen.

## SourceNode

Herkunftsnode eines Ereignisses, bei S7OPT-UA-Alarming ein durch eine NodeID spezifiziertes Objekt im ns="S7OPTSOURCES:" oder das S7-Verbindungsobjekt im ns="S7OPT:". Siehe auch im Kapitel "Bereichsbaum und Herkunftsraum (Seite 278)".

## SourceName

Siehe Kapitel "Bildung von SourceName, Message und Severity (Seite 265)".

## Time

Zeitstempel, der so nahe wie möglich am Prozess ist. Dies wird durch die Projektierung bestimmt, wobei es folgende Möglichkeiten gibt:

- PLC-Zeitstempel
- PLC-Zeit + Offset
- PC-Zeit (UTC)

Bei PC-Zeit (UTC) ist "Time" identisch mit "ReceiveTime".

## ReceiveTime

Zeitstempel des PC.

## Message

Siehe Kapitel "Bildung von SourceName, Message und Severity (Seite 265)".

## Severity

Siehe Kapitel "Bildung von SourceName, Message und Severity (Seite 265)".

## 2.8.16.2 Bildung von SourceName, Message und Severity

Die Namen der Properties werden teilweise durch das STEP 7-Projekt vorgegeben.

Die im STEP 7-Projekt verwendeten Namen, z.B. S7-Verbindungsname, Stationsname, PLC-Name werden zur Bildung der Properties herangezogen.

## Bildung des SourceName

Nodeld und Anzeigenamen des S7OPT-Eventtypen	Bildungsregel für den SourceName
ns= S7OPTTYPES; i=14 "S7OPTStatepathAlarmType"	S7-Verbindungsname + ".statepath" Beispiel: connection2.statepath Es existiert ein entsprechender SourceNode.
ns= S7OPTTYPES; i=15 "S7OPTConsistencyAlarmType"	S7-Verbindungsname + ".statepath" Beispiel: connection2.statepath Es existiert ein entsprechender SourceNode.
ns= S7OPTTYPES; i=43 "S7OPTOffNormalAlarmType"	Projektierte Programmmeldungen: Der SourceName besteht aus dem Stationsname (z.B. „station_1“) + „\“ PLC-Name (z.B. „PLC_1516“) + „\“ Symbolischen Namen des Instanz-Datenbausteins der PLC-Meldung (z.B. „motor1“) Beispiel: „station_1\PLC_1516\motor1“
ns= S7OPTTYPES; i=61 "S7OPTInfoReportEventType"	Projektierte Programmmeldungen: Der SourceName besteht aus dem Stationsname (z.B. „station_1“) + „\“ PLC-Name (z.B. „PLC_1516“) + „\“ Symbolischen Namen des Instanz-Datenbausteins der PLC-Meldung (z.B. „motor1“) Beispiel: „station_1\PLC_1516\motor1“

## Bildung der Message

Nodeld des S7OPT-Eventtypen	Bildungsregel für die Message
ns= S7OPTTYPES; i=14 "S7OPTStatepathAlarmType"	"statepath"
ns= S7OPTTYPES; i=15 "S7OPTConsistencyAlarmType"	"consistency"
ns= S7OPTTYPES; i=43 "S7OPTOffNormalAlarmType"	Projektierte Programmmeldungen: Meldetext der Programmmeldungen. Der Meldetext kann dynamische Parameter (Variablen und Textlisten) enthalten.
ns= S7OPTTYPES; i=61 "S7OPTInfoReportEventType"	Projektierte Programmmeldungen: Meldetext der Programmmeldungen. Der Meldetext kann dynamische Parameter (Variablen und Textlisten) enthalten.

## Bildung der Severity

Nodeld und Anzeigenamen des S7OPT-Eventtypen	Bildungsregel für die Severity
ns= S7OPTTYPES; i=14 "S7OPTStatepathAlarmType"	Für jede S7-Verbindung erfasste Vorgabe-Priorität.
ns= S7OPTTYPES; i=15 "S7OPTConsistencyAlarmType"	Für jede S7-Verbindung erfasste Vorgabe-Priorität.
ns= S7OPTTYPES; i=43 "S7OPTOffNormalAlarmType"	Projektierte Programmmeldungen: Wird aus der PLC-Meldungspriorität (0 bis 16) der Programmmeldungen abgeleitet.
ns= S7OPTTYPES; i=61 "S7OPTInfoReportEventType"	Projektierte Programmmeldungen: Wird aus der PLC-Meldungspriorität (0 bis 16) der Programmmeldungen abgeleitet.

Tabelle 2- 6 Umformungstabelle PLC-Meldungspriorität - Severity

PLC-Meldungspriorität	Severity
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
13	813
14	875
15	938
16	1000

### 2.8.16.3 Standard-Eventtyp mit dem Anzeigenamen "ConditionType"

Nodeld: ns:"<http://opcfoundation.org/UA/>", i=2782  
 Abgeleitet von: ns:"<http://opcfoundation.org/UA>", i=2041 mit Anzeigenamen "BaseEventType"

Browse-Namen der relevanten Properties

"" (1|Nodeld)  
 "ConditionName" (13|String)  
 "EnableState|ID" (13|Boolean)(Browse-Path)  
 "EnableState" (13|LocalizedText)  
 "Quality" (13|StatusCode)

Direkt von diesem Typen abgeleitete S7OPT-Eventtypen: keine direkt oder indirekt davon abgeleitete Eventtypen haben eine Condition-Instanz.

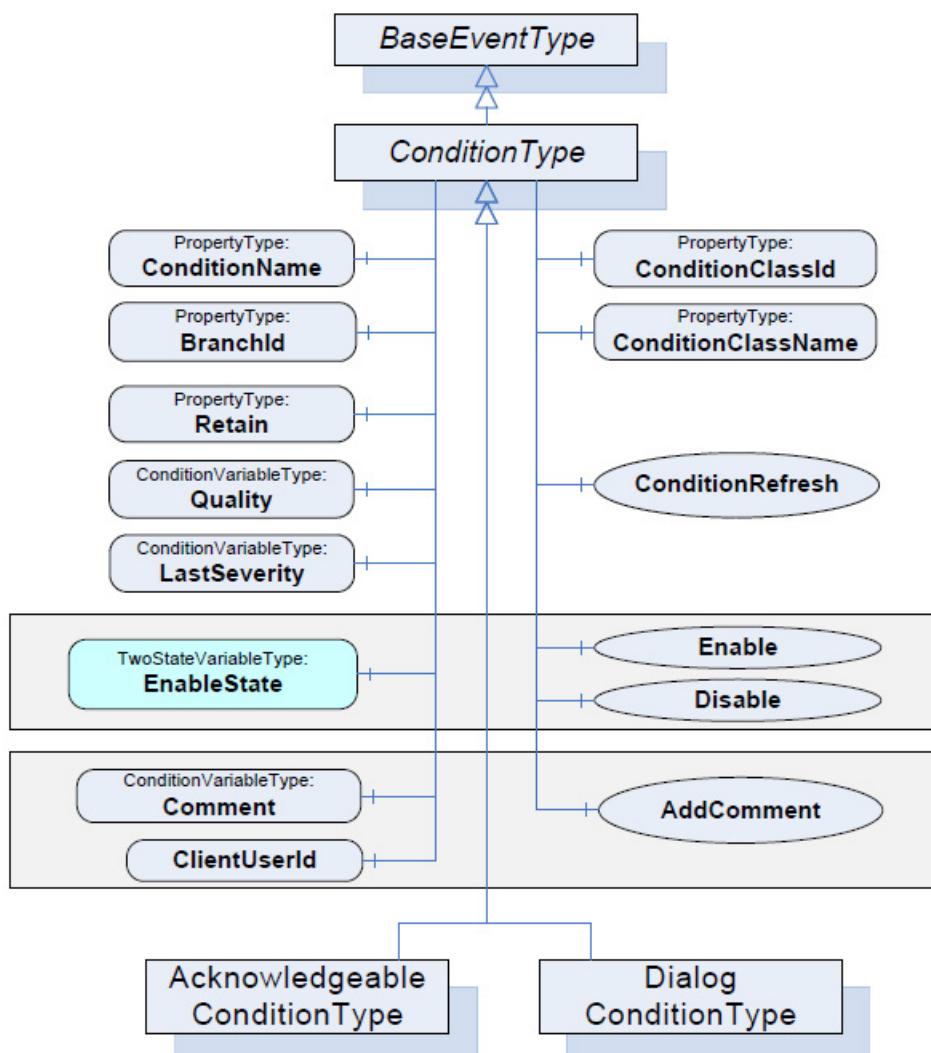


Bild 2-49 Darstellung des ConditionType

### ConditionName

Ein Event einer Condition wird durch die Kombination "SourceName" und "ConditionName" eindeutig identifiziert. Siehe unten, Abschnitt "Bildung von ConditionName".

### EnableState

Ist Teil einer Zustandsmaschine. Mögliche Werte gemäß UA-Alarming-Spezifikation für EnableState sind "de";"Eingeschaltet" und "de";"Ausgeschaltet".

### Quality

Die möglichen Quality-Werte siehe UA-Spezifikation.

#### 2.8.16.4 Bildung von ConditionName

Die Namen der Properties werden teilweise durch das STEP 7-Projekt vorgegeben.

Nodeld und Anzeigenamen des S7OPT-Eventtypen	Bildungsregel für den ConditionName
ns=S7OPTTYPES:, i=14 "S7OPTStatepathAlarmType"	"statepath"
ns= S7OPTTYPES:, i=15 "S7OPTConsistencyAlarmType"	"consistency"
ns= S7OPTTYPES:, i=43 "S7OPTOffNormalAlarmType"	Projektierte Programmmeldungen: Name der Programmmeldungen Beispiel: „MyAlarm“
ns= S7OPTTYPES:, i=61 "S7OPTInfoReportEventType"	Projektierte Programmmeldungen: Name der Programmmeldungen Beispiel: „MyAlarm“

### 2.8.16.5 Standard-Eventtyp mit dem Anzeigenamen "AcknowledgeableConditionType"

Nodeld: ns="http://opcfoundation.org/UA/", i=2881

Abgeleitet von: ns="http://opcfoundation.org/UA", i=2782 mit Anzeigenamen "ConditionType"

Browse-Namen der relevanten Properties

"AckedState|Id" (13|Boolean)

"AckedState" (13|LocalizedText)

Direkt von diesem Typen abgeleitete S7OPT-Eventtypen: keine

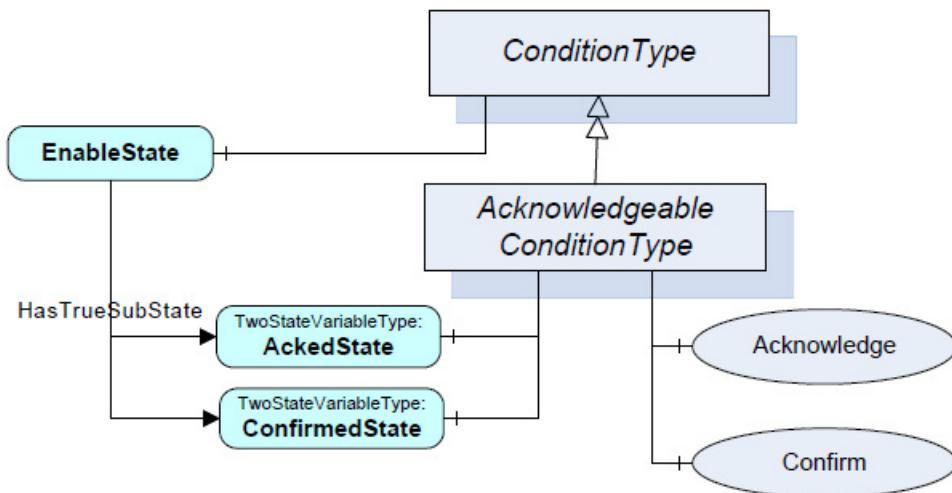


Bild 2-50 Darstellung des AcknowledgeableConditionType

#### AckedState

Ist ein Teil einer Zustandsmaschine. Mögliche Werte sind "de";"Quittiert" und "de";"Unquittiert". Jede Werteveränderung von AckedState wird mit einem Event gemeldet. Events, die mit dem AckedState "de", "Unquittiert" gemeldet werden, müssen mit der Acknowledge-Funktion quittiert werden. Siehe Systemhandbuch "SIMATIC STEP 7 Professional" bzw. Informationssystem in SIMATIC STEP 7 Professional (TIA Portal).

### 2.8.16.6 Standard-Eventtyp mit dem Anzeigenamen "AlarmConditionType"

Nodeld: ns="http://opcfoundation.org/UA/", i=2915

Abgeleitet von: ns="http://opcfoundation.org/UA", i=2881 mit Anzeigenamen

"AcknowledgeableConditionType"

Browse-Namen der relevanten Properties

"ActiveState|Id" (13|Boolean)

"ActiveState" (13|LocalizedText)

Direkt von diesem Typen abgeleitete S7OPT-Eventtypen: keine

Mögliche Werte gemäß UA-Alarming-Spezifikation für ActiveState sind "de";"Aktiv" und "de";"Inaktiv".

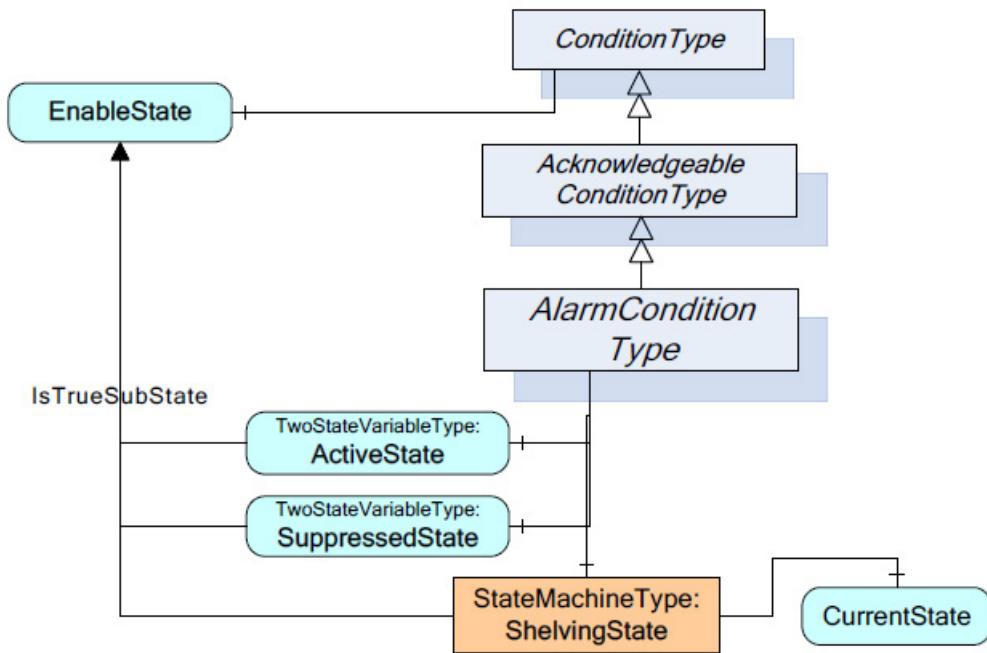


Bild 2-51 Darstellung des AlarmConditionType

## ActiveState

Ist Teil einer Zustandsmaschine. Bei S7OPT-UA-Alarming wird der ActiveState durch PLC-Meldungen gesteuert. Wenn das meldungsauslösende Signal den Wert "1" hat, wird ActiveState "Aktiv" gemeldet, wenn es den Wert "0" hat, dann wird "Inaktiv" gemeldet. Jede Werteänderung von ActiveState wird durch einen Event gemeldet.

### ActiveState bei Eventtyp mit dem Anzeigenamen="S7OPTStatepathAlarmType"

Hier wird der Zustand "Aktiv" erreicht, wenn die S7-Verbindung nicht aufgebaut ist. Der Zustand "Inaktiv" ist erreicht, wenn die S7-Verbindung aufgebaut ist.

#### 2.8.16.7 Standard-Eventtyp mit dem Anzeigenamen "OffNormalAlarmType"

NodeID: ns="http://opcfoundation.org/UA/", i=10637

Abgeleitet indirekt von: ns="http://opcfoundation.org/UA/", i=2915 mit Anzeigenamen "AlarmConditionType"

Browse-Namen der relevanten Properties: keine

Direkt von diesem Typen abgeleiteten S7OPT-Eventtypen:

ns=S7OPTTYPES;, i =14, Anzeigenamen="S7OPTStatepathAlarmType"

ns=S7OPTTYPES;, i =15, Anzeigenamen="S7OPTConsistencyAlarmType"

ns=S7OPTTYPES;, i =43, Anzeigename="S7OPTOffNormalAlarmType"

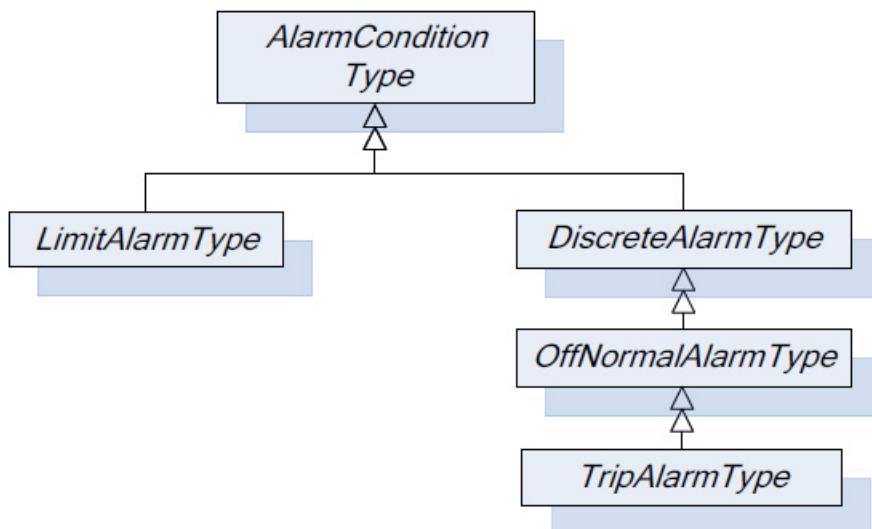


Bild 2-52 Einordnung des OffNormalAlarmType in die Alarmtyp-Hierarchie

## 2.8.17 S7OPT-Eventtypen

### 2.8.17.1 S7OPT-Eventtyp mit dem Anzeigenamen "S7OPTStatepathAlarmType"

NodId: ns="S7OPTTYPES:", i=14

Abgeleitet indirekt von: ns="http://opcfoundation.org/UA/", i=2915 mit Anzeigenamen "AlarmConditionType"

Browse-Namen der relevanten Properties:  
"S7OPTConnection" (13|NodId)

Dieser S7OPT-Eventtyp bildet den inversen Zustand einer S7-Verbindung ab ("Inaktiv", wenn S7-Verbindung aufgebaut ("UP") ist; und "Aktiv", wenn S7-Verbindung nicht aufgebaut ist ("DOWN" oder "RECOVERY")).

Der Zustand wird nur PC-seitig ermittelt und kann hierdurch auch dann gemeldet werden, wenn keine physikalische Verbindung zur PLC besteht.

Im Zustand „Aktiv“ können keine weiteren PLC-Meldungen empfangen werden.

#### S7OPTConnection

Gibt die NodId der Verbindung an, über die eine PLC-Meldung empfangen wird.

Beispiel: ns="S7OPT:", s="Verbindungsname".

### 2.8.17.2 S7OPT-Eventtyp mit dem Anzeigenamen "S7OPTConsistencyAlarmType"

NodId: ns="S7OPTTYPES:", i=15

Abgeleitet indirekt von: ns="http://opcfoundation.org/UA/", i=2915 mit Anzeigenamen "AlarmConditionType"

Browse-Namen der relevanten Properties:  
"S7OPTConnection" (13|NodId)

Der S7OPT-Eventtyp "S7OPTConsistencyAlarmType" hat keine Quittierungspflicht. Er zeigt eine inkonsistente Alarmprojektierung zwischen einer PLC und der PC-Station an. Solche inkonsistente PLC-Meldungen werden nicht an den OPC-Client gemeldet. In diesem Fall wird für diese PLC einmalig die Condition-Instanz "consistency" auf "Aktiv" gesetzt. Es wird empfohlen die Alarmprojektierung zuerst auf der PLC und anschließend auf der PC-Station zu aktualisieren.

## S7OPTConnection

Gibt die Nodeld der Verbindung an, über die eine PLC-Meldung empfangen wird.  
Beispiel: ns="S7OPT:", s="Verbindungsname".

### 2.8.17.3 S7OPT-Eventtyp mit dem Anzeigenamen "S7OPTOffNormalAlarmType"

Nodeld: ns="S7OPTTYPES:", i=43

Abgeleitet indirekt von: .ns="http://opcfoundation.org/UA/", i=10637 mit Anzeigenamen "OffNormalAlarmType"

Browse-Namen der relevanten Properties:

- "S7OPTAlarmId" (13|UInt32)
- "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)

Dieser S7OPT-Eventtyp bildet die projektierten "Programmmeldungen" mit oder ohne Quittierungspflicht mit bis zu zehn Begleitwerten ab.

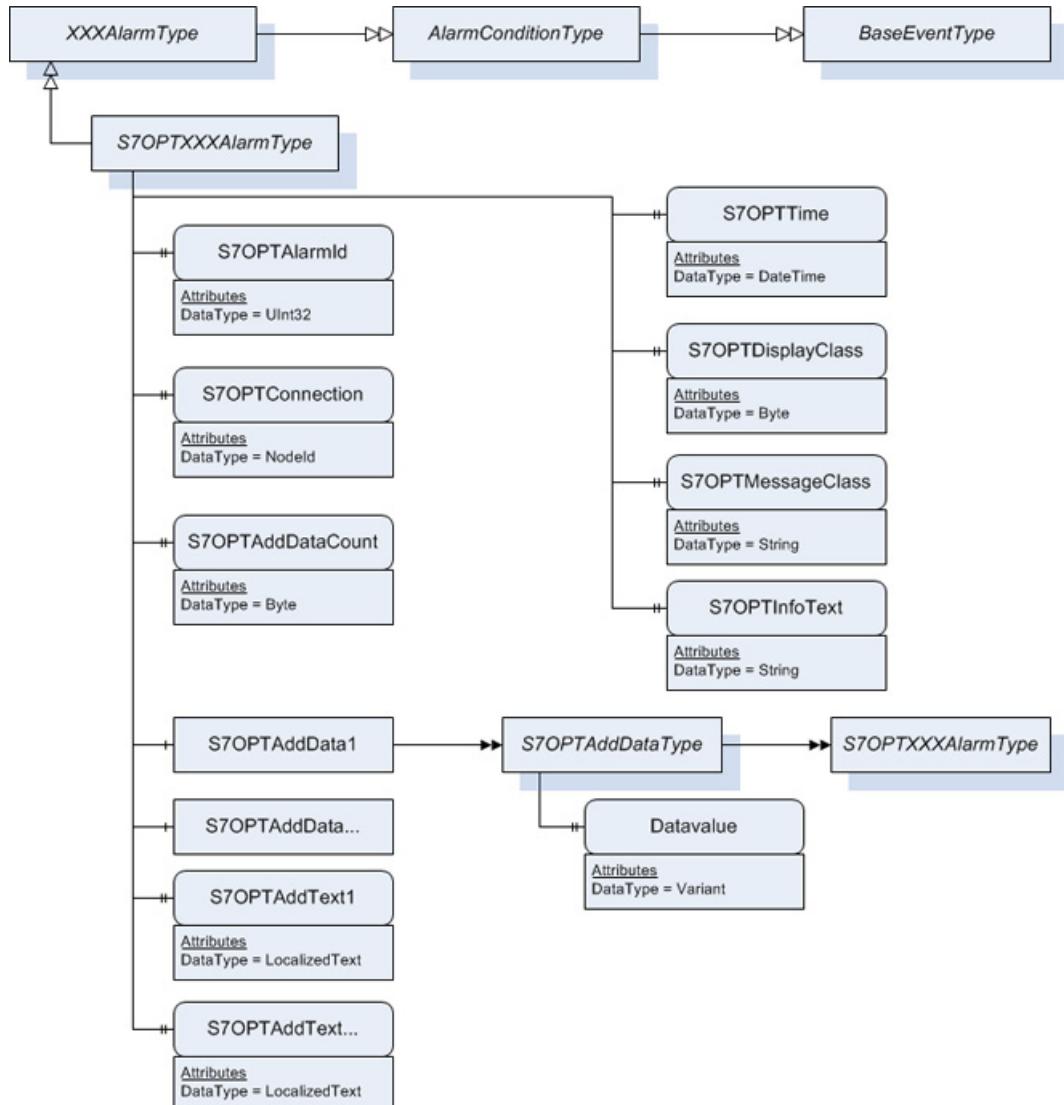


Bild 2-53 Darstellung der S7-spezifischen Properties für den S7OPTOffNormalAlarmType

### S7OPTAlarmId

Meldungsnummer der PLC-Meldung. Sie ist PLC-weit eindeutig.

### S7OPTConnection

Gibt die Nodeld der Verbindung an, über die die PLC-Meldung empfangen wird.  
Beispiel: ns="S7OPT:", s="Verbindungsname".

### S7OPTTime

Der Zeitstempel zeigt an, wann die PLC-Meldung in der PLC aufgetreten ist (PLC-Zeit).

### S7OPTDisplayClass

Anzeigeklasse der PLC-Meldung.

### S7OPTAlarmClass

Meldeklasse der PLC-Meldung. Die Meldeklasse bestimmt, ob die Meldung quittierungspflichtig ist oder nicht.

### S7OPTInfoText

Infotext der PLC-Meldungen aus der Projektierung. Der Infotext kann dynamische Parameter (Variablen und Textlisten) enthalten.

### S7OPTAddDataCount

Ist die tatsächliche Anzahl der Begleitwerte der PLC-Meldungen.

### S7OPTAddData[n]||Datavalue

Sind die Begleitwerte der generierenden PLC-Meldungen.  
 $1 \leq n \leq 10$ .

### S7OPTAddText[n]

$1 \leq n \leq 9$ . Zusatztexte der PLC-Meldungen aus der Projektierung.

### 2.8.17.4 S7OPT-Eventtyp mit dem Anzeigenamen "S7OPTInfoReportEventType"

Nodeld: ns="S7OPTTYPES:", i=61  
Abgeleitet indirekt von: .ns="http://opcfoundation.org/UA/", i=2041 mit Anzeigenamen "BaseEventType"  
Browse-Namen der relevanten Properties:  
"S7OPTInfoReportId" (13|UInt32)  
"S7OPTInfoReportName" (13|String)  
"S7OPTConnection" (13|Nodeld)  
"S7OPTTTime" (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)

Dieser S7OPT-Eventtyp bildet die projektierten Programmmeldungen, die nur zu Informationszwecken dienen, mit bis zu zehn Begleitwerten ab.

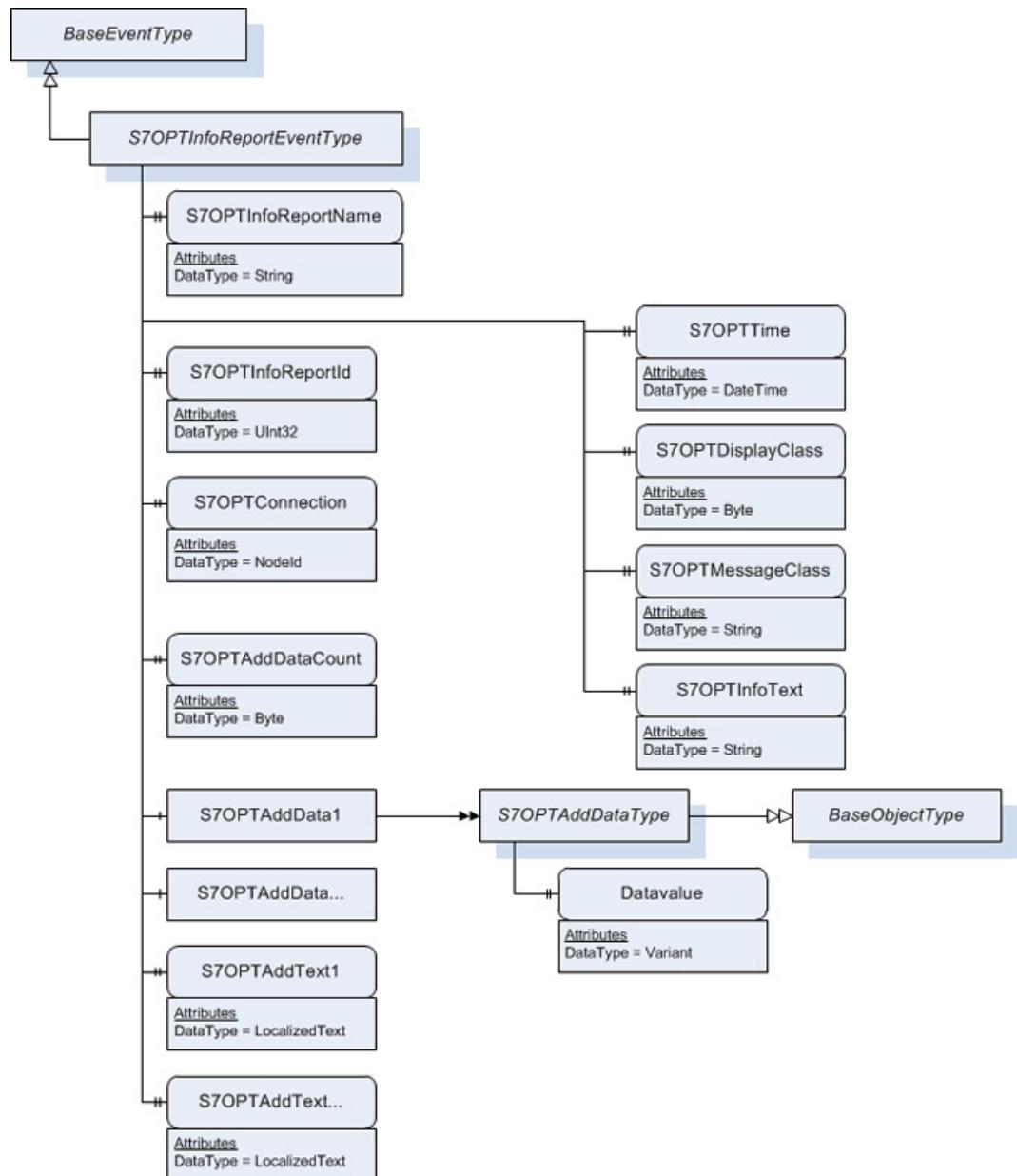


Bild 2-54 Darstellung der S7-spezifischen Properties für den S7OPTInfoReportEventType

### S7OPTInfoReportId

Meldungsnummer der PLC-Meldung. Sie ist PLC-weit eindeutig.

### S7OPTInfoReportName

Setzt sich aus "inforeport" + Meldungsnummer der PLC-Meldung zusammen.  
Beispiel: inforeport56

### **S7OPTConnection**

Gibt die Nodeld der Verbindung an, über die die PLC-Meldung empfangen wird.  
Beispiel: ns="S7OPT:", s="Verbindungsname".

### **S7OPTTime**

Der Zeitstempel zeigt an, wann die PLC-Meldung in der PLC aufgetreten ist.

### **S7OPTDisplayClass**

Anzeigeklasse der PLC-Meldung.

### **S7OPTAlarmClass**

Meldeklasse der PLC-Meldung. Die Meldeklasse bestimmt, ob die Meldung quittierungspflichtig ist oder nicht.

### **S7OPTInfoText**

Infotext der PLC-Meldungen aus der Projektierung. Der Infotext kann dynamische Parameter (Variablen und Textlisten) enthalten.

### **S7OPTAddDataCount**

Ist die tatsächliche Anzahl der Begleitwerte der PLC-Meldungen.

### **S7OPTAddData[n]|Datavalue**

Sind die Begleitwerte der generierenden PLC-Meldungen.  
 $1 \leq n \leq 10$ .

### **S7OPTAddText[n]**

$1 \leq n \leq 9$ . Zusatztexte der PLC-Meldungen aus der Projektierung.

## **2.8.18**

### **Bereichsbau und Herkunftsraum**

Bei S7OPT-OPC-UA-Server enthalten der Bereichsbau und der Herkunftsraum die SourceNodes von projektierten PLC-Meldungen.

S7-UA-Alarming baut den Bereichsbau auf, um die SourceNodes Bereichen zuzuordnen. Knoten des Bereichsbau sind spezielle UA-Ordner, die Bereiche der Anlage abbilden. Diese speziellen UA-Ordner werden Bereichsknoten genannt.

Als Bereichsbau wird der Pfadname vom STEP 7-Projekt verwendet. Er setzt sich aus folgenden Komponenten innerhalb der Projektnavigation zusammen:

- Stationsname
- PLC-Name
- Gruppen

Die benutzerdefinierten Gruppen erlauben eine topologische Abbildung der bestehenden Kunden-Anlage. Sie können in der Projektnavigation des STEP 7-Projekts eingefügt werden.

Der Stationsname und der PLC-Name sind fest miteinander verbunden, sie können nicht getrennt werden. Gruppen können nur davor oder danach eingefügt werden.

Der Stationsname wird automatisch vor dem Stationsnamen eingefügt.

Beispiel:

```

plant1 (Gruppe)
    station_1 (Stationsname)
        PLC_1516 (PLC-Name)
            house1 (Gruppe)
                motor1 (Instanz-Datenbaustein der PLC-Meldung)
            house2 (Gruppe)
                motor2 (Instanz-Datenbaustein der PLC-Meldung)

```

Aus dem Beispiel leiten sich der Bereichsbau und die SourceNodes wie folgt ab:

Knoten des Bereichsbau:

- 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".

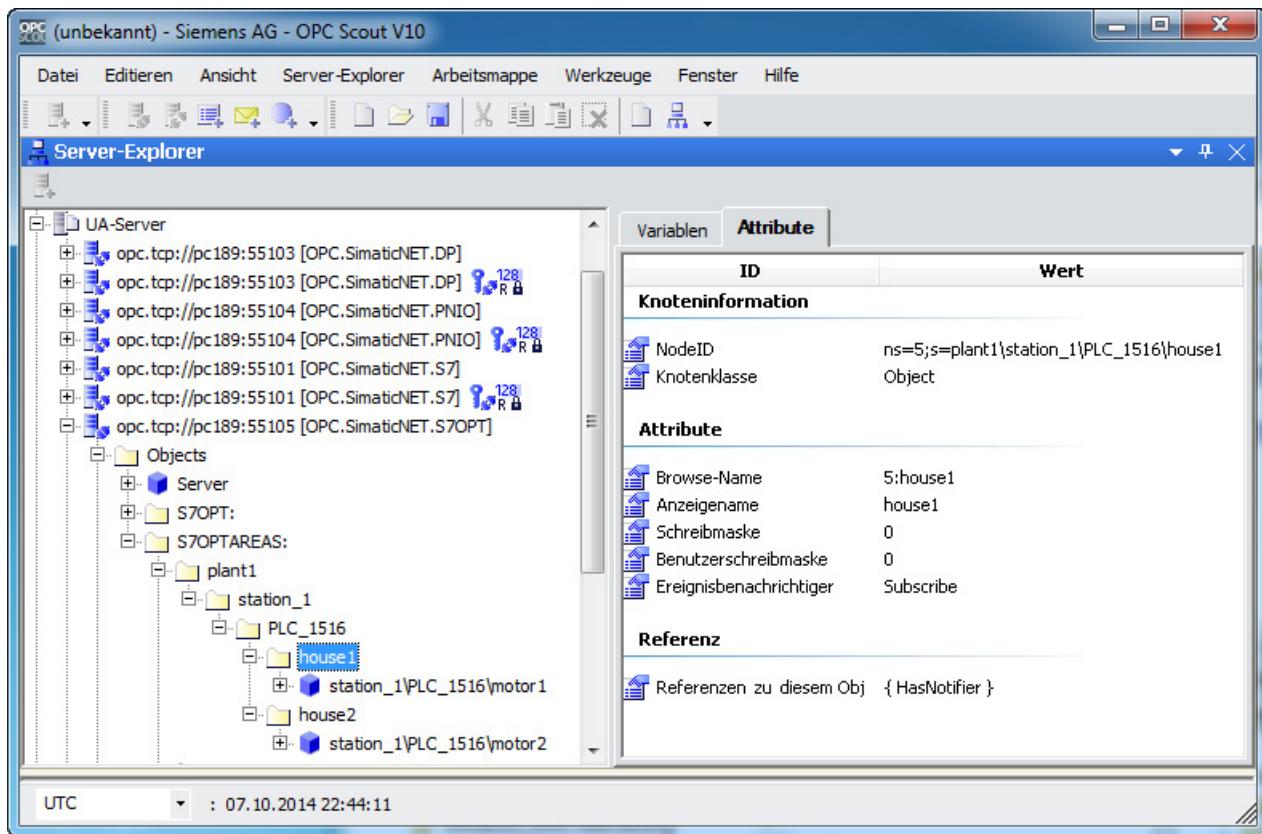
Die SourceName der SourceNode setzen sich wie folgt zusammen:

- Stationsname + „\“
- PLC-Name (Name der projektierten PLC) + „\“
- Symbolische Namen des Instanz-Datenbausteins der PLC-Meldung

Entsprechend dem Beispiel:

- ns="S7OPTSOURCES:", s="station\_1\PLC\_1516\motor1".
- ns="S7OPTSOURCES:", s="station\_1\PLC\_1516\motor2".

Der Bereichsbau und der Herkunftsraum können mit dem OPC-Client (z. B. OPC Scout V10) durchsucht werden.



Ausgewählt ist im linken Fenster der Bereichsknoten mit dem Anzeigenamen "house1" (`NodeID ns="S7OPTAREAS:", s="plant1\station_1\PLC_1516\house1"`). Im rechten Fenster werden seine Attribute angezeigt, es ist diesem zu entnehmen, dass der Bereichsknoten die Referenz "HasNotifier" aufweist.

## 2.8.19 Empfang von Events

### EventItems und EventNotifier

Die einzige Möglichkeit, aktuelle Events von einem Server zu empfangen, ist das Erzeugen von einem oder mehreren beobachteten Event-Items in einer Subscription. Bei einem Event-Item wird das Attribut EventNotifier (12) beobachtet. EventNotifier weisen nur Objekte auf, die die Referenz "HasNotifier" haben. Bei S7OPT-OPC-UA-Alarming sind das der Server-Knoten `ns="http://opcfoundation.org/UA/", i=2253`, alle Knoten der S7-Verbindungen, z.B. `ns="S7OPT:", s="S7-Verbindung_1"` und alle Bereichsknoten im `ns="S7OPTAREAS:"`.

Wird ein EventNotifier eines Bereichsknoten beobachtet, werden alle Events gemeldet, die in diesem Bereich entstanden sind. Die Anzahl der gemeldeten Events kann je nach Ausprägung der Anlage (die Menge der potentiell verfügbaren EventSourcen kann dem Herkunftsraum kombiniert mit dem Bereichsraum entnommen werden) sehr hoch sein. Daher ist eine sinnvolle Auswahl der Events erforderlich.

## Filterung von Events

Neben der geeigneten Auswahl des Event-Items besteht an der OPC UA-Schnittstelle auch die Möglichkeit, nach Events mit bestimmten Properties und ihren Werten zu filtern.

---

### Hinweis

Die Auswahl des Event-Items sowie die Einstellung von Filterbedingungen bestimmen maßgeblich, ob ein Event von einem OPC-Client empfangen werden kann.

---

## Auswahl der Properties

Beim Erzeugen eines beobachteten Event-Items, müssen die zurück zuliefernden Properties angegeben werden. Werden keine Properties angegeben, weiß man, dass ein Event aufgetreten ist, jedoch nicht welches. Für die Aufnahme einer Property in die Liste der zurück zuliefernden Properties, muss für jede Property folgendes angegeben werden:

- die Nodeld des definierten Eventtyps (Typeld),
- der Browse-Name ggf. Browse-Pfad von der Property und
- die Attributeld

Nodeld, Browse-Namen ggf. Browse-Pfad und die Attributeld des definierten Eventtyps können Sie dem Kapitel "Eventtyp hierarchie von S7OPT-OPC-UA-Server (Seite 261)" entnehmen.

Ein Event muss nicht alle zurück zu liefernden Properties aufweisen. Für Properties, die ein Event nicht hat, wird "null" zurückgemeldet. Dabei kann S7OPT-OPC-UA-Alarming nicht zwischen Properties unterscheiden, die das Event nicht hat, oder solchen, die aufgrund von der fehlerhaften Angabe z. B. des Browse-Namen, keine gültige Property bezeichnen.

---

### Hinweis

An der OPC-UA-Schnittstelle werden nicht alle Zustandswechsel eines Alarms durch ein Event weitergereicht. Es kann vorkommen, dass bei schneller Änderung des Alarmzustands, nicht alle Zustandswechsel mit einem Event mitgeteilt werden, sondern dass nur der letzte nun aktuelle Zustand gesendet wird. Dies ist abhängig von OPC-Parametern, wie z.B. "PublishingInterval" sowie der Projektierung und der Rechnerleistung.

---

## **2.8.20      Methoden von UA-Alarmen**

### **Enable()/Disable()**

Für einen Alarm, der "disabled" ist, werden keine Events an Clients generiert. Zur PLC hin werden jedoch unabhängig davon die Alarne immer überwacht, da Alarne dorthin nicht einzeln "enabled"/"disabled" werden können.

### **ACHTUNG**

#### **Deaktivierung von Events**

Bei Verwendung dieser Methoden, müssen Sie beachten, dass ein Client die Events zu anderen Clients deaktivieren kann.

### **AddComment()**

Für jede Alarinstanz kann jeweils ein Kommentar gespeichert werden.

### **ConditionRefresh()**

Es werden alle aktiven oder unquittierten Alarne gemeldet (Property "Retain" = true).

### **Acknowledge()**

Die Quittierung eines Alarms wird zur PLC übertragen. Die Umschaltung des Quittungszustandes wird unabhängig davon von der PLC zum OPC-Server explizit zurückgemeldet, erst dann wechselt der Alarm OPC-seitig in den quittierten Zustand und löst entsprechende Events aus.

## **2.9**

## **Offene Kommunikationsdienste (SEND/RECEIVE)**

Je nach Kommunikationsnetz werden zwei Ausprägungen der offenen Kommunikationsdienste (SEND/RECEIVE) unterschieden:

- Offene Kommunikationsdienste (SEND/RECEIVE) über Industrial Ethernet  
Protokoll-ID: SR
- Offene Kommunikationsdienste (SEND/RECEIVE) über PROFIBUS  
Protokoll-ID: FDL

## **2.9.1**

## **Offene Kommunikationsdienste (SEND/RECEIVE) über Industrial Ethernet**

Der offene Kommunikationsdienst (SEND/RECEIVE) über Industrial Ethernet wird auch als SEND/RECEIVE-Protokoll bezeichnet. Es ermöglicht die Kommunikation mit S5- und S7-Geräten, sowie mit Fremdgeräten.

## Eigenschaften der offenen Kommunikationsdienste (SEND/RECEIVE) über Industrial Ethernet

- Kommunikation über SIMATIC S5-Hantierungsbausteine und S7-Funktionsbausteine.
- Kopplung zweier PC-Stationen möglich.
- Unterstützung der WRITE- und FETCH-Funktion zum Zugriff auf Objekte des Partnergeräts.
- Schneller Zugriff auf große Datenpakete über die Betriebsarten SEND und RECEIVE.
- Teilzugriff innerhalb eines Datenpakets.
- Anzeige und Überwachung des Verbindungszustands.

### 2.9.1.1 Performanter SIMATIC NET OPC-Server für das SR-Protokoll

#### Einleitung

Der folgende Abschnitt beschreibt eine Konfigurationsvariante für das SR-Protokoll, die höhere Performanceanforderungen erfüllt. Hierfür werden alle unterlagerten SR-Protokollbibliotheken und der COM-Server als Inproc-Server in den Outproc-OPC-Server geladen. Die Protokollbearbeitung läuft im Prozess des OPC-Servers ab, weitere Laufzeiten für Prozesswechsel und Multiprotokollbetrieb fallen weg. Der Prozesswechsel zwischen OPC-Client und OPC-Server ist allerdings noch vorhanden.

## Konfiguration

Die Aktivierung dieser performanten Variante erfolgt implizit dadurch, dass im Konfigurationsprogramm "Kommunikations-Einstellungen" das Protokoll "SR" als einziges Protokoll ausgewählt wird (bei Auswahl weiterer Protokolle oder der OPC-UA-Schnittstelle entfällt der beschriebene Performance-Vorteil):

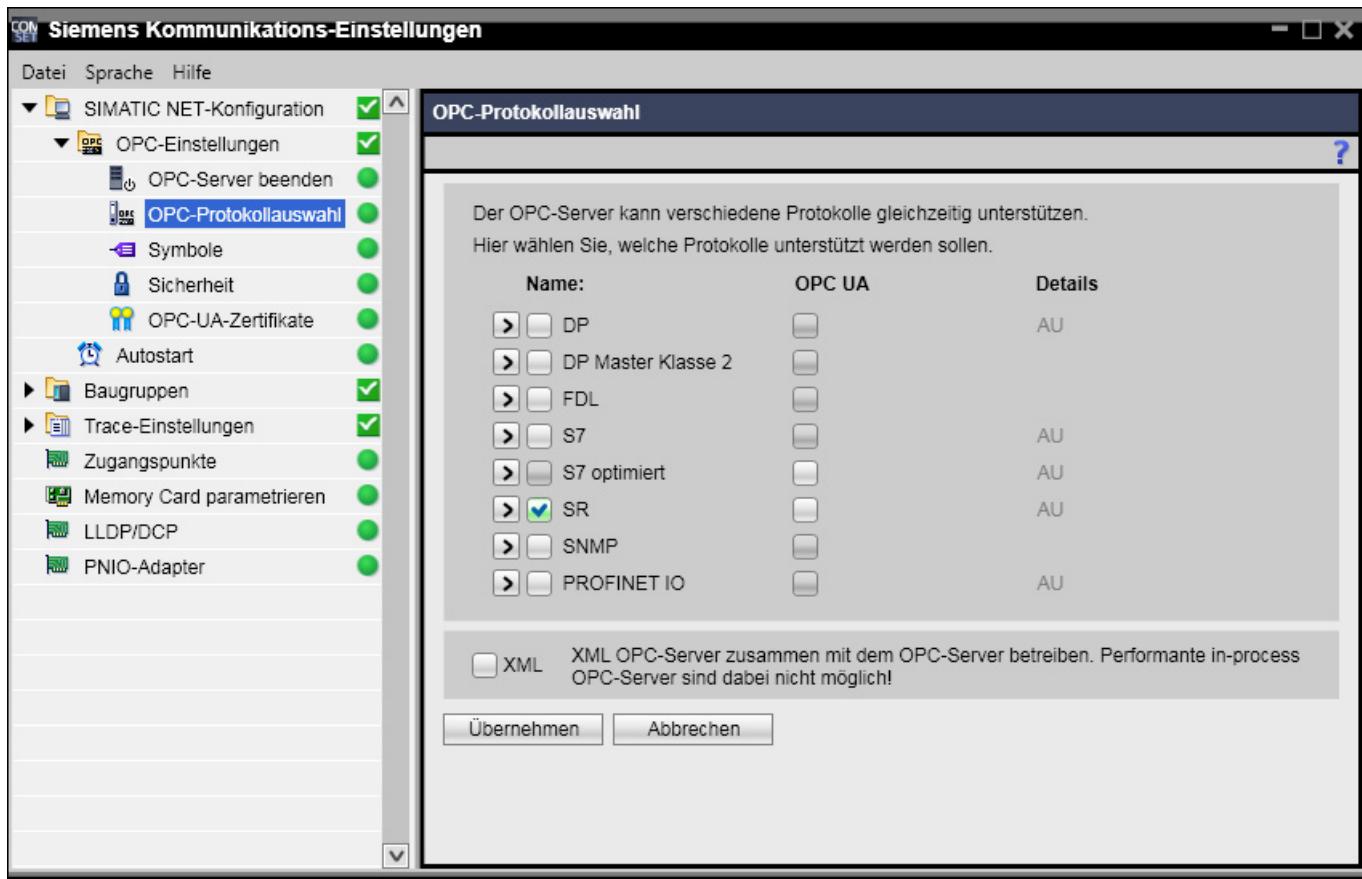


Bild 2-55 Fenster des Konfigurationsprogramms "Kommunikations-Einstellungen" zur Auswahl des SR-Protokolls

"Symbolik" darf zusätzlich ausgewählt werden.

### Hinweis

Für die Erstellung von Symbolen mit dem Symbol-Editor ist die Verwendung folgender Zeichen erlaubt: A-Z, a-z, 0-9, \_, -, ^, !, #, \$, %, &, ', /, (, ), <, >, =, ?, ~, +, \*, ',', :, |, @, [, ], {, }, ". Zusätzlich sollten Sie bei der Erstellung von Symbolen mit STEP 7 darauf achten, dass es bei der Array-Auflösung zu Problemen kommen kann, wenn Ihre Symboldatei gleichzeitig Symbole der Form <Symbolname> und <Symbolname>[<Index>] enthält.

## Vorteile / Nachteile

Die Verwendung des performanten SR-OPC-Servers hat jedoch den Nachteil, dass nur der Einzelprotokollbetrieb von SR möglich ist. Dem stehen folgende Vorteile gegenüber:

- Höhere Performance als beim Multiprotokollbetrieb.
- Einfache Konfiguration.
- Mehrere Clients können den Server zur gleichen Zeit nutzen.
- Die Stabilität des OPC-Servers ist nicht von den Clients abhängig.

### 2.9.1.2 Protokoll-ID

Die Protokoll-ID für das SEND/RECEIVE-Protokoll lautet "SR".

### 2.9.1.3 Verbindungsnamen

Der Verbindungsname ist der in STEP 7 projektierte Name zur Identifikation der Verbindung.

Es werden folgende Verbindungstypen vom OPC-Server unterstützt:

- ISO-Transportverbindung
- ISO-on-TCP-Verbindung
- TCP-Verbindung

### Beispiele für Verbindungsnamen

Typisches Beispiel ist:

*SR\_Connection*

### 2.9.1.4 Variabldienste

Variabldienste ermöglichen den direkten Zugriff und die Beobachtung von Variablen im Automatisierungsgerät. Die Adressierung der Variablen erfolgt symbolisch, die Notation der Variablennamen orientiert sich an den Programmierwerkzeugen. Zum direkten Lesezugriff auf Variablen überträgt der OPC-Server die gewünschte Adressinformation an den Empfänger und dieser sendet daraufhin die angeforderten Daten zurück. Bei einem Schreibzugriff überträgt der OPC-Server die Adressinformation zusammen mit dem zu schreibenden Wert an das Partnergerät.

---

#### Hinweis

Die angeforderte Variable muss mit der Projektierung des Partnergeräts konsistent sein. Andernfalls führt der Zugriff auf bestimmte Bereiche zu Kommunikationsfehlern. Der OPC-Server kann bei der Anmeldung einer Variablen nur die Syntax überprüfen. Es ist nicht möglich, festzustellen, ob auf Grund der Projektierung des Partners die Variable im Partnergerät gültig ist.

---

## Syntax der Prozessvariablen für SEND/RECEIVE-Variabldiensten

SR: [<Verbindungsname>]<Bereich>{,}<Typ><Adresse>{,}<Anzahl>}

### Erklärungen

#### SR

SEND/RECEIVE-Protokoll für den Zugriff auf die Prozessvariable.

#### <Verbindungsname>

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

#### <Bereich>

Objekt, das angesprochen werden soll.

DB $nn$	Datenbaustein Nr. nn
A	Ausgang
E	Eingang
M	Merker
P	Peripherie (für S5-CPUs)
PAE	Peripherie (für S7-CPUs)
BS	Systembereich
AS	Absolute Anfangsadresse
DX $nn$	Erweiterter Datenbaustein
DE $nn$	Datenbaustein im Externspeicher
QB	Erweiterte Peripherie

#### <Typ>

Datentyp.

Der Datentyp (Formatbezeichner) wird im OPC-Server in den entsprechenden OLE-Datentyp umgewandelt.

Formatbezeichner	Beschreibung	OLE-Datentyp	Visual Basic-Typ
X	Bit (Boolean) Sie müssen die Bit-Nummer (0...7) angeben.  Hinweis: Nur bei den Objekten E, A, M, P, PAE, DB und QB (nur Lesezugriff)	VT_BOOL	Boolean
B oder BYTE	Byte (unsigned8)  Hinweis: Nur bei den Objekten E, A, M, P, PAE, QB verfügbar	VT_UI1	Byte
CHAR	Byte (signed8)  Hinweis: Nur bei den Objekten E, A, M, P, PAE, QB verfügbar	VT_I1	Integer
W oder WORD	Wort (unsigned16)	VT_UI2	Long
INT	Wort (signed16)	VT_I2	Integer

Formatbezeichner	Beschreibung	OLE-Datentyp	Visual Basic-Typ
D oder DWORD	Doppelwort (unsigned32)	VT_UI4	Double
DINT	Doppelwort (signed32)	VT_I4	Long
REAL	Fließkomma, IEEE Darstellung	VT_R4	Single
S5REAL	Fließkomma, S5 Darstellung	VT_R4	Single

**<Adresse>**

Adresse der Variablen im Bereich.

Abhängig vom Bereich ist die Adresse, die angegeben werden muss, eine Byte- oder eine Wortadresse. Bei Zugriff auf die folgenden Bereiche wird die angegebene Adresse als Wortadresse interpretiert: DB $nn$

Wortadressen:

- DX $nn$
- DE $nn$

Für die anderen Bereiche ist die Adresse eine Byte-Adresse. Für Datentyp X ist die Syntax der <Adresse> = <Byte-Offset.Bit>

Der Wertebereich Bit ist 0...7.

Der Wertebereich für den Byte-Offset ist bei Byte-Adressen 0...65534 und bei Wortadressen 0...32767. Geräte- und Typabhängig kann der tatsächlich verwendbare Wert der Adresse geringer sein.

Wort- und Byteadressen werden folgendermaßen adressiert:

- Wortadressen: <Wort-Nummer>{.Bit-Nummer}
- Byteadressen: <Byte-Nummer>{.Bit-Nummer}

**<Anzahl>**

Anzahl der Variablen eines Typs, die ab der im Parameter *Adresse* angegebenen Adresse angesprochen werden (Wertebereich 0...65535).

Beim Datentyp X ist die Eingabe der Anzahl für

- Schreibzugriff (WRITE): nur Vielfache von 8 bei Bitnummer 0 beginnend.
- Lesezugriff (FETCH): beliebige Anzahl bei beliebigem Offset beginnend.

**Beispiel:**

SR:[ISO-WRITE-1]DB1,X10.0,16, Zugriffsrechte W, Offset 10, ab Bitnummer 0, Anzahl 16 Bit

SR:[ISO-FETCH-1]DB1,X10.4,9, Zugriffsrechte R, Offset 10, ab Bitnummer 4, Anzahl 9 Bit

**Syntax für Timer und Zähler**

SR: [<Verbindungsname>] <Nummer>

- T $nn$   
Timer
- Z $nn$   
Zähler

### 2.9.1.5 Blockorientierte Dienste

Die blockorientierten Dienste ermöglichen eine programmgesteuerte Übertragung größerer Datenblöcke. Diese Dienste werden auch als SEND/RECEIVE-Dienste bezeichnet. Die Übertragung mit dem OPC-Server wird durch Variablen realisiert:

- Variablen, die Datenblöcke empfangen
- Variablen, die Datenblöcke senden

Eine Standardgröße der Datenblöcke wird in der Projektierung festgelegt, beim Senden von Variablen kann die Größe eingeschränkt werden. Ein Teilzugriff innerhalb der Datenblöcke ist möglich.

#### Fest definierte Variablennamen

Folgende Variablennamen sind für jede Verbindung fest definiert:

- *RECEIVE*
- *SEND*

#### Syntax der Prozessvariablen für blockorientierte Dienste

Es gibt folgende Möglichkeiten:

`SR: [<Verbindungsname>] receive{,<Typ><Adresse>{,<Anzahl>}}`

`SR: [<Verbindungsname>] send{<n>}{,<Typ><Adresse>{,<Anzahl>}}`

Zusätzlich wurde ab SIMATIC NET PC Software Edition 2006 eine zweite Möglichkeit zum Empfang von Daten eingerichtet. Bei aktiviertem Item werden Daten gemeldet, auch wenn sich diese nicht geändert haben:

`SR: [<Verbindungsname>] receivedata{,<Typ><Adresse>{,<Anzahl>}}`

Dieses Item ist nicht konform zur OPC-Spezifikation.

#### Erklärungen

##### SR

SEND/RECEIVE-Protokoll für den Zugriff auf die Prozessvariable.

##### <Verbindungsname>

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

##### receive

Zuletzt vom Partner empfangener Datenpuffer.

Die Struktur des Datenpuffers ist nicht vorgegeben. Daher wird der Puffer immer als Feld von Bytes geliefert.

OLE-Datentyp	Visual Basic Typ
VT_ARRAY VT_UI1	Byte

---

### Hinweis

Die RECEIVE-Variable entspricht einem Empfangspuffer. Deshalb kann die Variable nur gelesen werden. Bei einem Lesezugriff auf das Gerät (DEVICE) wird explizit ein Empfangspuffer im Kommunikationssystem bereitgestellt. Wird nicht innerhalb einer Fehlerwartezeit für diesen Puffer ein Datenblock empfangen, so wird ein Fehler erzeugt. Deshalb sollten Sie diese Variablen nur beobachten oder aus dem Cache lesen.

---

### **receivedata**

Das Item "receivedata" ist nicht OPC-konform.

Funktionell entspricht "receivedata" dem zuvor beschriebenen Item "receive". Es gibt aber folgenden Unterschied:

Bei aktivem Item dieser Art werden neue Daten gemeldet, auch wenn sich die Inhalte nicht geändert haben. Dadurch wird es einem Client ermöglicht, auch unveränderte gesendete Datenpuffer vom Partner zu erhalten. Ebenso erfolgt der OnDataChange-Callback nicht schneller als die ausgehandelte Aktualisierungszeit (Update rate). Stellen Sie also hierfür immer Aktualisierungszeiten schneller als die Senderate der SEND/RECEIVE-Daten ein.

### **send**

Ein Sendepuffer, der an den Verbindungspartner übertragen werden kann.

Für die Größe des Sendepuffers ist durch die Projektierung ein Defaultwert festgelegt. Der Puffer wird immer als Feld von Bytes geliefert.

Schreibzugriffe auf diese Variable bewirken, dass der Sendepuffer an den Partner übertragen wird.

OLE-Datentyp	Visual Basic Typ
VT_ARRAY VT_UI1	Byte

---

### Hinweis

Diese Variable und daraus abgeleitete Variablen dürfen nicht gelesen und nicht aktiviert werden. Lesezugriffe auf diese Variable werden unter Umständen vom Partnergerät mit einem Verbindungsabbau quittiert.

---

### Hinweis

Beim quittierten Schreiben auf send-Variablen bedeutet ein Ergebnis "S\_OK" nur, dass der Sendepuffer erfolgreich an das Kommunikationssystem abgegeben wurde und zur Abholung für den Receive-Partner bereit steht. Es wird keine Aussage darüber getroffen, ob der Puffer tatsächlich von einer Partnerapplikation entgegen genommen und verarbeitet wurde.

---

### **<n>**

Größe des Sendepuffers in Byte.

Sie können *n* verwenden, wenn auf einer Verbindung Sendepuffer verschiedener Größe

verwendet werden sollen. Wenn *n* weggelassen wird, wird die in der Projektierung eingetragene Puffergröße verwendet.

Bei TCP/IP native mit abgeschaltetem Miniprotokoll (siehe Projektierungswerkzeug SIMATIC STEP 7 oder SIMATIC NCM PC) ist die Angabe der Größe des Sendepuffers nicht möglich. In diesem Fall wird die in der Projektierung eingetragene Puffergröße verwendet.

Die Größe des Sendepuffers ist von der Projektierung abhängig. Beachten Sie, dass die <Anzahl> multipliziert mit der Größe des <Typ> in Byte der Variablen nicht größer als der Sendepuffer <n> gewählt werden darf.

#### <Typ>

Datentyp.

Der Datentyp (Formatbezeichner) wird im OPC-Server in den entsprechenden OLE-Datentyp umgewandelt.

Formatbezeichner	Beschreibung	OLE-Datentyp	Visual Basic-Typ
X	Bit (Boolean)	VT_BOOL	Boolean
B oder BYTE	Byte (unsigned8)	VT_UI1	Byte
CHAR	Byte (signed8)	VT_I1	Integer
W oder WORD	Wort (unsigned16)	VT_UI2	Long
INT	Wort (signed16)	VT_I2	Integer
D oder DWORD	Doppelwort (unsigned32)	VT_UI4	Double
DINT	Doppelwort (signed32)	VT_I4	Long
REAL	Fließkomma, IEEE Darstellung	VT_R4	Single
S5REAL	Fließkomma, S5 Darstellung	VT_R4	Single

#### <Adresse>

Byteadresse der Variablen im Bereich *Byte-Nummer*.

Für den Datentyp Bit wird das angeforderte Bit durch *Byte-Nummer.Bit-Nummer* adressiert.

**<Anzahl>**

Anzahl der Variablen eines Typs, die ab der im Parameter *Adresse* angegebenen Adresse angesprochen werden sollen.

**Hinweis**

Bitte beachten Sie folgende Hinweise:

- Durch die optionale Angabe von Typ, Adresse und Anzahl können Sie strukturiert auf Teilbereiche von Datenblöcken zugreifen.
- Variablen für Sendedaten oder Empfangsdaten mit unterschiedlicher Länge oder unterschiedlichem Verbindungsnamen verfügen über unabhängige Speicherbereiche.
- Der Sendedatenpuffer wird allokiert und mit Null initialisiert, wenn ein Item für einen unabhängigen Speicherbereich angelegt wird. Ein Schreibauftrag auf ein send-Item wird in einen internen Schreibpuffer geschrieben und übertragen.
- Beim quittierten Schreiben auf send-Variablen bedeutet ein Ergebnis "S\_OK" nur, dass der Sendepuffer erfolgreich an das Kommunikationssystem abgegeben wurde und zur Abholung für den Receive-Partner bereit steht. Es wird keine Aussage darüber getroffen, ob der Puffer tatsächlich von einer Partnerapplikation entgegen genommen und verarbeitet wurde.
- Die Übertragung der Datenblöcke erfolgt azyklisch. Es wird immer der vollständige Sendedatenblock übertragen. Dies gilt auch bei Subelementzugriff oder wenn mehrere Clients gleichzeitig dieses Item beschreiben.

**Beispiele für Prozessvariablen für blockorientierte Dienste**

Hier finden Sie Beispiele, die die Syntax von Variablennamen für blockorientierte Dienste verdeutlichen.

**Empfangsvariablen**

**SR:[MyConnection]receive,w4,6**  
receive,w4,6  
6 Datenworte ab Offset 4 im Empfangspuffer.

**SR:[MyConnection]receive,dword7**  
receive,dword7  
Ein Doppelwort ab Offset 7 im Empfangspuffer

**SR:[MyConnection]receive,REAL0,2**  
receive,REAL0,2  
Ein Feld mit 2 Fließkommawerten ab Offset 0 im Empfangspuffer.

Beispiel für die Übertragung von Daten, die sich nicht geändert haben müssen (siehe "receivedata"):

**SR:[MyConnection]receivedata,w4,6**  
receivedata,w4,6  
6 Datenworte ab Offset 4 im Empfangspuffer.

## Sendevariablen

**SR:[MyConnection]send30,dword7**

send30,dword7

Ein Doppelwort ab Byte 7 in einem 30 Byte großen Sendepuffer. Wenn der Standardwert der Größe des Sendpuffers nicht gleich 30 ist, greift diese Variable auf einen separaten Puffer zu.

**SR:[MyConnection]send,B20,6**

send,B20,6

Ein Feld mit 20 Bytes ab Offset 6 in einem Sendepuffer mit Standardgröße. Die Standardgröße des Sendepuffers wird in der Projektierung festgelegt.

**SR:[MyConnection]send8,DINT0**

send8,DINT0

Ein vorzeichenbehaftetes Doppelwort ab Adresse 0 in einem Sendepuffer mit der Größe 8.

### 2.9.1.6 SEND/RECEIVE-spezifische Informationsvariablen

Mit den SEND/RECEIVE-spezifischen Informationsvariablen können Sie Informationen über den Verbindungsstatus abfragen.

Folgende Information kann ermittelt werden:

*Status einer Verbindung*

## Syntax der Informationsvariablen für offene Kommunikationsdienste (SEND/RECEIVE)

SR: [<Verbindungsname>] &<Informationsparameter>()

## Erklärungen

**SR**

SEND/RECEIVE-Protokoll für den Zugriff auf die Prozessvariable.

**<Verbindungsname>**

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

**<Informationsparameter>**

Es gibt folgende Möglichkeiten:

- statepath
- Zustand einer Kommunikationsverbindung zu einem Partnergerät.

Das Ergebnis wird als String dargestellt.

Rückgabewerte:

**DOWN**

Verbindung ist nicht aufgebaut

**UP**

Verbindung ist aufgebaut

***RECOVERY***

Verbindung wird aufgebaut

***ESTABLISH***

Reserviert für zukünftige Erweiterungen

OLE-Datentyp	Visual Basic Typ
VT_BSTR	String

***statepathval***

Zustand einer Kommunikationsverbindung zu einem Partnergerät.

Das Ergebnis wird als Zahl dargestellt.

Rückgabewerte:

1

Verbindung ist nicht aufgebaut

2

Verbindung ist aufgebaut

3

Verbindung wird gerade aufgebaut

4

Reserviert für zukünftige Erweiterungen

OLE-Datentyp	Visual Basic Typ
VT_UI1	Byte

### Beispiele für SEND/RECEIVE-spezifische Informationsvariablen

Hier finden Sie einige Beispiele für Rückgabewerte von Informationsvariablen für die offene Kommunikation (SEND/RECEIVE):

*SR:[SR\_CONNECTION]&statepath()*

**&statepath()**

kann beispielsweise folgenden Wert zurückgeben:

*UP*

Die Verbindung ist aufgebaut

*SR:[SR\_CONNECTION]&statepathval()*

**&statepathval()**

kann beispielsweise folgenden Wert zurückgeben:

*2*

Die Verbindung ist aufgebaut.

### 2.9.1.7 Syntax der systemspezifischen Informationsvariablen

*SR:[SYSTEM]&version()*

**&version()**

Liefert eine Versionskennung für den SR-OPC-Server, hier die Zeichenfolge, z.B.  
SIMATIC NET Core Server SR V 7.xxxx.yyyy.zzzz Copyright 2012

Datentyp: VT\_BSTR

Zugriffsrecht: nur lesbar

*S7:[SYSTEM]&sapiversion()*

**&winsockversion()**

Versionskennung der Winsocket-Schnittstelle.

Datentyp: VT\_BSTR

Zugriffsrecht: nur lesbar

## 2.9.2 Offene Kommunikationsdienste (SEND/RECEIVE) über PROFIBUS

### Das ISO/OSI-Referenzmodell

Der offene Kommunikationsdienst (SEND/RECEIVE) über PROFIBUS ist ein einfaches, telegrammorientiertes Protokoll für SIMATIC Geräte. Es verwendet die Dienste des Fieldbus Data Links (FDL), der Ebene 2 des ISO/OSI-Referenzmodell bei PROFIBUS.

### Eigenschaften der offenen Kommunikationsdienste (SEND/RECEIVE) über PROFIBUS

Der OPC-Server von SIMATIC NET bietet folgende Eigenschaften:

- Kommunikation über SIMATIC S5 und SIMATIC S7-Hantierungsbausteine auf einem Automatisierungsgerät
- Kopplung zweier PC-Stationen möglich
- Schnelle Übertragung von Datenpaketen
- Strukturierter Zugriff innerhalb eines Datenpakets
- Anzeige und Überwachung des Verbindzungszustandes

#### 2.9.2.1 Protokoll-ID

Die Protokoll-ID für das FDL-Protokoll lautet FDL.

#### 2.9.2.2 Verbindungsnamen

Der Verbindungsname ist der in STEP 7 projektierte Name zur Identifikation der Verbindung. Sie müssen den FDL-Verbindungsnamen für den OPC-Server eindeutig wählen.

### Beispiele für Verbindungsnamen

Ein typisches Beispiel ist:

### FDL-Connection

#### 2.9.2.3 Blockorientierte Dienste

Blockorientierte Dienste ermöglichen die programmgesteuerte Übertragung von Datenblöcken. Die empfangenen und die zu sendenden Datenpuffer werden auf OPC-Variablen abgebildet.

#### Fest definierte Variablennamen

Folgende Variablennamen sind für jede Verbindung fest definiert:

- *RECEIVE*
- *SEND*
- *SendSDA*
- *SendSDN*

#### Syntax der Prozessvariablen für blockorientierte Dienste (FDL)

Es gibt zwei Möglichkeiten:

FDL: [<Verbindungsname>] receive{,<Typ><Adresse>{,<Anzahl>}}

FDL: [<Verbindungsname>] send{<n>} {,<Typ><Adresse>{,<Anzahl>}}

#### Erklärungen:

##### FDL

FDL-Protokoll für den Zugriff auf die Prozessvariable.

##### <Verbindungsname>

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

##### receive

Zuletzt vom Partner empfangener Datenpuffer.

Die Struktur des Datenpuffers ist nicht vorgegeben. Daher wird der Puffer immer als Feld von Bytes geliefert.

---

##### Hinweis

Die RECEIVE-Variablen entspricht einem Empfangspuffer. Deshalb kann die Variable nur gelesen werden. Bei einem Lesezugriff auf das Gerät (DEVICE) wird explizit ein Empfangspuffer im Kommunikationssystem bereitgestellt. Wird nicht innerhalb einer Fehlerwartezeit für diesen Puffer ein Datenblock empfangen, so wird ein Fehler erzeugt. Deshalb sollten Sie diese Variablen nur beobachten oder aus dem Cache lesen.

---

OLE-Datentyp	Visual Basic Typ
VT_ARRAY VT_UI1	Feld mit Elementen vom Typ Byte

**send**

Immer vorhandener Sendepuffer, der an den Verbindungspartner übertragen werden kann. Für die Größe des Sendepuffers ist durch die Projektierung ein Defaultwert festgelegt. Der Puffer wird immer als Feld von Bytes geliefert.

Schreibzugriffe auf diese und daraus abgeleitete Variablen bewirken, dass der Sendepuffer an den Partner übertragen wird.

Diese Variable und daraus abgeleitete Variablen dürfen nicht gelesen und nicht aktiviert werden.

In Abhängigkeit von der Kombination der in der Verbindungsprojektierung angegebenen Stationsadresse und der SAPs wird beim Schreiben eines Send-Items ein spezieller FDL-Dienst verwendet:

LocalSAP	Remote Station	Remote SAP	Bedeutung/ verwendeter Dienst bei Send
0...62, 255	0...126	0...62, 255	Senden und Empfangen / SDA
0...62, 255	0...126	63	Nur Senden / SDA
63	0...126	0...62, 255	Nur Empfangen (kein Send)
0...62, 255	127	63	Broadcast: Nur Senden an Alle / SDN

LocalSAP	Remote Station	Remote SAP	Bedeutung/ verwendeter Dienst bei Send
63	127	0...62, 255	Broadcast Empfang (kein Send)
0..62, 255	127	0...62, 255	Multicast: Nur Senden an Alle, die den RemoteSAP aktiviert haben / SDN

OLE-Datentyp	Visual Basic Typ
VT_ARRAY VT_UI1	Feld mit Elementen vom Typ Byte

**Hinweis**

Beim quittierten Schreiben auf send-Variablen bedeutet ein Ergebnis "S\_OK" nur, dass der Sendepuffer erfolgreich an das Kommunikationssystem abgegeben wurde und zur Abholung für den Receive-Partner bereit steht. Es wird keine Aussage darüber getroffen, ob der Puffer tatsächlich von einer Partnerapplikation entgegen genommen und verarbeitet wurde.

<n>

Größe des Sendepuffers.

Mit blockorientierten Diensten können Sie Datenpuffer senden und empfangen. Die empfangenen und die zu sendenden Datenpuffer werden auf OPC-Variablen abgebildet. Ein entsprechend großes Feld kann beispielsweise den gesamten Empfangspuffer enthalten. Die Zuordnung von Teilabschnitten der Puffer zu einzelnen Variablen ist auch möglich.

Die Übertragung der Datenblöcke erfolgt azyklisch. Ein Sendeaufruf wird in einen internen, mit "Null" vorbelegten, Schreibpuffer geschrieben. Es wird immer der vollständige Datenblock des Schreibpuffers übertragen. Dies gilt auch bei Teilzugriff oder wenn mehrere Clients gleichzeitig dieses Item beschreiben.

#### <Typ>

Datentyp.

Der Datentyp (Formatbezeichner) wird im OPC-Server in den entsprechenden OLE-Datentyp umgewandelt.

Formatbezeichner	Beschreibung	OLE-Datentyp	Visual Basic-Typ
X	Bit (Boolean)	VT_BOOL	Boolean
B oder BYTE	Byte (unsigned8)	VT_UI1	Byte
CHAR	Byte (signed8)	VT_I1	Integer
W oder WORD	Wort (unsigned16)	VT_UI2	Long
INT	Wort (signed16)	VT_I2	Integer
D oder DWORD	Doppelwort (unsigned32)	VT_UI4	Double
DINT	Doppelwort (signed32)	VT_I4	Long
REAL	Fließkomma, IEEE Darstellung	VT_R4	Single
S5REAL	Fließkomma, S5 Darstellung	VT_R4	Single

#### <Adresse>

Byteadresse der Variablen im Bereich *Byte-Nummer*. Der Wertebereich ist von der Projektierung abhängig.

Für den Datentyp *Bit* wird das angeforderte Bit durch *Byte-Nummer.Bit-Nummer* adressiert.

<Byte-Nummer>{.<Bit-Nummer>}

Wenn Sie bei <Bit-Nummer> nichts eintragen, wird Bit mit 0 adressiert.

#### <Anzahl>

Anzahl der Variablen eines Typs, die ab der im Parameter *Adresse* angegebenen Adresse angesprochen werden sollen. Der Wertebereich ist von der Projektierung abhängig.

### Syntax der Prozessvariable für blockorientierte Dienste über SDA und SDN (FDL)

Es kann vorteilhaft sein, dass der verwendete Dienst nicht von der Kombination der Adresse der Partnerstation und den SAPs abhängt.

Durch Verwendung der Namen "SendSDA" und "SendSDN" können Sie festlegen, dass nur die jeweiligen Dienste "SDA" oder "SDN" zum Senden verwendet werden sollen. Diese besonderen Namen werden im OPC-Browsing nicht angezeigt.

## Syntax:

Es gibt zwei Möglichkeiten:

FDL: [<Verbindungsname>] SendSDA{<n>} {,<Typ><Adresse>{,<Anzahl>}}

FDL: [<Verbindungsname>] SendSDN{<n>} {,<Typ><Adresse>{,<Anzahl>}}

## Erklärungen

### FDL

FDL-Protokoll für den Zugriff auf die Prozessvariable.

### <Verbindungsname>

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

SendSDA

SendSDN

Es werden nur die Dienste SDA oder SDN zum Senden verwendet.

### <n>

Größe des Sendepuffers.

Mit blockorientierten Diensten können Sie Datenpuffer senden und empfangen. Die empfangenen und die zu sendenden Datenpuffer werden auf OPC-Variablen abgebildet. Ein entsprechend großes Feld kann beispielsweise den gesamten Empfangspuffer enthalten. Die Zuordnung von Teilabschnitten der Puffer zu einzelnen Variablen ist auch möglich.

### <Typ>

Datentyp.

Der Datentyp (Formatbezeichner) wird im OPC-Server in den entsprechenden OLE-Datentyp umgewandelt.

Formatbezeichner	Beschreibung	OLE-Datentyp	Visual Basic-Typ
X	Bit (Boolean)	VT_BOOL	Boolean
B oder BYTE	Byte (unsigned8)	VT_UI1	Byte
CHAR	Byte (signed8)	VT_I1	Integer
W oder WORD	Wort (unsigned16)	VT_UI2	Long
INT	Wort (signed16)	VT_I2	Integer
D oder DWORD	Doppelwort (unsigned32)	VT_UI4	Double
DINT	Doppelwort (signed32)	VT_I4	Long
REAL	Fließkomma, IEEE Darstellung	VT_R4	Single
S5REAL	Fließkomma, S5 Darstellung	VT_R4	Single

### <Adresse>

Byteadresse der Variablen im Bereich *Byte-Nummer*. Der Wertebereich ist von der Projektierung abhängig.

Für den Datentyp Bit wird das angeforderte Bit durch *Byte-Nummer.Bit-Nummer* adressiert.

**<Anzahl>**

Anzahl der Variablen eines Typs, die ab der im Parameter *Adresse* angegebenen Adresse angesprochen werden sollen. Der Wertebereich ist von der Projektierung abhängig.

**Hinweis**

Bitte beachten Sie folgende Hinweise:

- Durch die optionale Angabe von Typ, Adresse und Anzahl können Sie strukturiert auf Teilbereiche von Datenblöcken zugreifen.
- Variablen für Sendedaten oder Empfangsdaten mit unterschiedlicher Länge oder unterschiedlichem Verbindungsnamen verfügen über unabhängige Speicherbereiche.
- Der Sendedatenpuffer wird allokiert und mit Null initialisiert, wenn ein Item für einen unabhängigen Speicherbereich angelegt wird. Ein Schreibauftrag auf ein send-Item wird in einen internen Schreibpuffer geschrieben und übertragen.
- Die Übertragung der Datenblöcke erfolgt azyklisch. Parallelie Netzaufträge für die gleichen Daten sind möglich. Es wird immer der vollständige Sendedatenblock übertragen. Dies gilt auch bei Subelementzugriff oder wenn mehrere Clients gleichzeitig dieses Item beschreiben.  
Ein paralleles Schreiben des gleichen Sendedatenblocks oder von Teilbereichen des Sendedatenblocks durch mehrere Clients kann zu Inkonsistenzen führen. Es wird deshalb empfohlen:
  - immer den vollständigen Datenblock zu lesen oder zu schreiben.
  - die Maximale Anzahl paralleler Netzaufträge auf Eins setzen, was jedoch die Übertragungsleistung reduziert.

**Beispiele für Prozessvariablen für blockorientierte Dienste (FDL)**

Hier finden Sie Beispiele, die die Syntax von Variablennamen für FDL-Variablen verdeutlichen.

**Lesevariable**

*FDL:[MyConnection]receive,w0,6*  
6 Datenworte ab Byte 0 im Empfangspuffer.

*FDL:[MyConnection]receive,dword7*  
Ein Doppelwort ab Byte 7.

*FDL:[MyConnection]Receive,REAL0,2*  
Zwei Realwerte ab Byte 0 im Empfangspuffer

**Schreibvariablen**

*FDL:[MyConnection]send30,dword7*  
Ein Doppelwort ab Byte 7 in einem 30 Byte großen Sendepuffer.

*FDL:[MyConnection]SendSDN,B5,20*  
Ein Feld mit 20 Bytes ab Offset 5 in einem Sendepuffer mit Default-Größe. Die Default-Größe wird durch die Projektierung vorgegeben. Unabhängig von der Angabe der SAPs in der Projektierung wird der PROFIBUS-FDL-Dienst SDN verwendet.

*FDL:[MyConnection]Send8,DINT0*

Ein vorzeichenbehaftetes Doppelwort ab Adresse 0 in einem Sendepuffer der Größe 8 Bytes.

## 2.9.2.4 FDL-spezifische Informationsvariablen

Der OPC-Server für offene Kommunikationsdienste (SEND/RECEIVE) über PROFIBUS (FDL) stellt Variablen zur Verfügung, mit denen Informationen über Zustand und Ausbau des Kommunikationsnetzes ermittelt werden können.

### Syntax der FDL-spezifischen Informationsvariablen

```
FDL: [ | <CP-Name> ] &<Informationsparameter>()
```

### Erklärungen

#### FDL

FDL-Protokoll für den Zugriff auf die Prozessvariable.

#### <CP-Name>

Der CP-Name wird bei der Projektierung festgelegt. Es werden nur jene CP-Namen unterstützt, über die auch eine FDL-Verbindung projektiert ist.

#### <Informationsparameter>

Es gibt fünf Möglichkeiten:

- busparameter
- defaultsap
- identify
- ts
- lifelist

Alle fünf Möglichkeiten werden nachfolgend beschrieben.

#### busparameter

Liefert die Busparameter des an der angegebenen Verbindung betriebenen PROFIBUS-Netzes. Die Werte werden als Bytefeld zurückgeliefert und entsprechen den von dem FDL-Dienst FDL\_READ\_VALUE gelieferten Ergebnissen. Detaillierte Informationen können dem Handbuch zur FDL-Programmierschnittstelle entnommen werden.

Inhalt	Beschreibung	Datentyp	Länge (Bytes)
HSA	Höchste PROFIBUS-Adresse am Bus, 2...126	BYTE	1
TS	PROFIBUS-Adresse des lokalen Teilnehmers, 0...hsa bzw. 126.	BYTE	1
Station_Type	Typ des lokalen Teilnehmers	INTEGER	2
Baudrate	Übertragungsrate	INTEGER	2
Medium_red	Redundanz	INTEGER	2

Inhalt	Beschreibung	Datentyp	Länge (Bytes)
Retry_Ctr	Anzahl der Aufrufwiederholungen an einen nicht antwortenden Teilnehmer (remote), 0...7.	UWORD	2
Default_SAP	Nummer des Default-SAP der Station (local), 0...63	BYTE	1
network_con_SAP	reserviert	BYTE	1
TSL	SLOT Time	UWORD	2
TQUI	Modulatorausklingzeit / Repeater-Umschaltzeit	UWORD	2
TSET	Setup-Time	UWORD	2
MIN_TSDR	Minimum der station delay time	UWORD	2
MAX_TSDR	Maximum der station delay time	UWORD	2
TTR	Target rotation time	DWORD	4
GAP	GAP-Updatefaktor	BYTE	1
in_Ring_desired	Ringaufnahmewunsch	BOOLEAN	1
physical_layer	Einstellbare Busphysik	INTEGER	2
ident	Vendor-Name, Controllertype, Hard- und Softwarestände	STRING	211

OLE-Datentyp	Visual Basic Typ
VT_ARRAY   VT_UI1	Byte()

**defaultsap**

Liefert den Wert für den Default-SAP (SAP = Service Access Point). Immer wenn ein SAP nicht explizit angegeben wird, gilt für die Adressierung der sogenannte Default-SAP.

OLE-Datentyp	Visual Basic Typ
VT_UI1	Byte

**identify**

Liefert die Teilnehmeridentifikation der angegebenen Verbindung als Feld mit 4 Strings:

Elemente des Rückgabewerts:

*Hersteller*

*Controller*

*Hardware-Version*

*Software-Version*

OLE-Datentyp	Visual Basic Typ
VT_ARRAY   VT_BSTR	String()

**ts**

Gibt die lokale Stationsadresse der angegebenen Baugruppe zurück.

OLE-Datentyp	Visual Basic Typ
VT_UI1	Byte

**lifelist**

Informationen über die am Bus verfügbaren Teilnehmer.

Das 127 Elemente umfassende Feld enthält Informationen über jede mögliche Stationsadresse. Jeder Stationsadresse ist ein Index des Feldes zugeordnet.

Rückgabewerte der einzelnen Feldeinträge:

*FDL\_STATION\_NON\_EXISTENT*

Kein Teilnehmer vorhanden (Wert 0x10)

*FDL\_STATION\_PASSIVE*

Passiver Teilnehmer (Wert 0x00)

*FDL\_STATION\_READY\_FOR\_RING*

Teilnehmer bereit für die Aufnahme in den Token-Ring des PROFIBUS (Wert 0x30)

*FDL\_STATION\_ACTIVE*

Aktiver Teilnehmer (Wert 0x20)

OLE-Datentyp	Visual Basic Typ
VT_ARRAY of VT_UI1	Feld mit 127 Elementen vom Typ Byte()

**Hinweis**

Zur Ermittlung dieser Daten wird der FDL-Dienst *FDL\_LIFE\_LIST\_CREATE\_REMOTE* benutzt. Dieser Dienst erzeugt eine deutliche Busbelastung, deshalb sollte diese Variable möglichst nicht aktiv beobachtet werden.

**2.9.2.5****Syntax der systemspezifischen Informationsvariablen**

*FDL:[SYSTEM]&version()*

**&version()**

Liefert eine Versionskennung für den FDL-OPC-Server, hier z.B. die Zeichenfolge  
*SIMATIC NET Core Server FDL V 7.xxxx.yyyy.zzzz Copyright 2012*

Datentyp: VT\_BSTR

Zugriffsrecht: nur lesbar

## **2.10 Offene Kommunikationsdienste (SEND/RECEIVE) mit OPC UA über Industrial Ethernet**

### **2.10.1 Eigenschaften der SR-Kommunikation mit OPC UA**

Der SIMATIC NET OPC-Server ermöglicht die Nutzung der SR-Kommunikation über OPC UA.

Der SIMATIC NET OPC-UA-Server für die offenen Kommunikationsdienste (SEND / RECEIVE) über Industrial Ethernet ist für die Kommunikation mit S7-Geräten freigegeben. Zusätzlich ermöglicht er dem Anwender auch die Kommunikation zu Fremdgeräten.

Der SR-OPC-UA-Server von SIMATIC NET hat folgende Eigenschaften:

- Kommunikation über SIMATIC S5-Hantierungsbausteine und S7-Funktionsbausteine.
- Kopplung zweier PC-Stationen über Send und Receive möglich.
- Unterstützung der WRITE- und FETCH-Funktion zum Zugriff auf Objekte des Partnergeräts.
- Schneller Zugriff auf große Datenpakete über die Betriebsarten SEND und RECEIVE.
- Teilzugriff innerhalb eines Datenpakets.
- Anzeige und Überwachung des Verbindungszustands.

### **2.10.2 SIMATIC NET OPC-UA-Server für das SR-Protokoll**

#### **Einleitung**

Der folgende Abschnitt beschreibt eine Konfigurationsvariante für das SR-Protokoll, die auch OPC UA unterstützt. Hierfür werden die SR-COM-OPC-Data-Access-Server als Outproc-OPC-Server eingerichtet.

## Konfiguration

Die Aktivierung des SR-OPC-UA-Servers erfolgt durch die Auswahl von "SR" und "OPC UA" im Konfigurationsprogramm "Kommunikations-Einstellungen" im Katalog "OPC-Protokollauswahl":

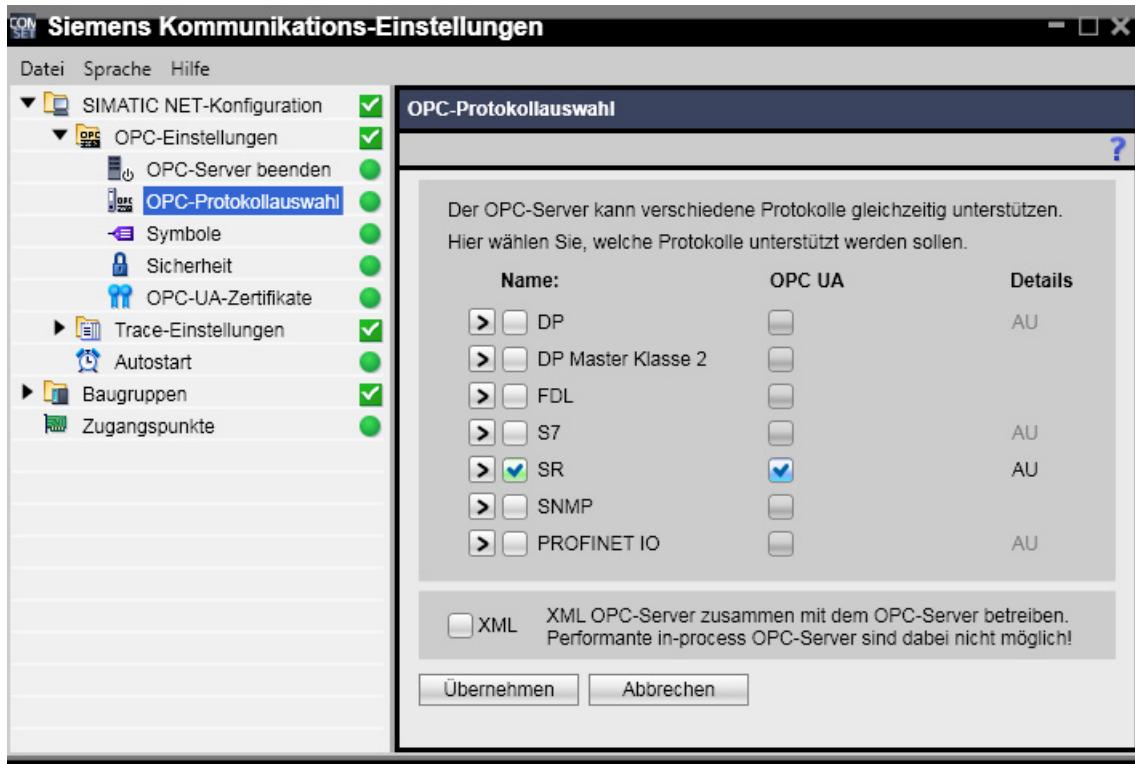


Bild 2-56 Fenster des Konfigurationsprogramms "Kommunikations-Einstellungen" zur Auswahl von OPC UA für das SR-Protokoll

## Vorteile / Nachteile

Bei Verwendung des SR-OPC-UA-Servers ist nur der Outproc-Betrieb von SR möglich. Der SR-OPC-UA-Server-Prozess muss zum Erhalt der Empfangsbereitsschaft in Betrieb sein. Ein Beenden des SR-OPC-UA-Servers - auch nach Abmeldung aller OPC-UA-Clients - wird nicht automatisch ausgeführt. Ein gleichzeitiger Betrieb des performanten Inproc-OPC-DA-SR-Servers ist nicht möglich.

Dem stehen folgende Vorteile gegenüber:

- Es ist keine COM/DCOM-Konfiguration mehr nötig.
- Performante, sichere Kommunikation
- Für Ereignisse und Datenzugriffe ist nur noch ein Server erforderlich.

### 2.10.3 Wie wird der SR-OPC-UA-Server adressiert?

#### Server-URL

Für das TCP-Protokoll gibt es für den OPC-Client zwei Möglichkeiten der Server-Adressierung:

- Direkte Adressierung:
  - opc.tcp://<hostname>:55102
  - oder
  - opc.tcp://<IP-Adresse>:55102
  - oder
  - opc.tcp://localhost:55102

Der SR-OPC-UA-Server hat den Port 55102.

- Die URL des SR-OPC-UA-Servers kann auch über den OPC-UA-Discovery-Dienst gefunden werden.

Die Eingabe zum Auffinden des Discovery-Servers lautet:

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

Der Discovery-Server hat den Port 4840 (für TCP-Verbindungen) und den Port 52601 (für HTTP-Verbindungen).

#### IPv6-Adresse

Für die IP-Adresse kann auch eine IPv6-Adresse verwendet werden. Die Adresse muss in Klammern angegeben werden, z.B. [fe80:e499:b710:5975:73d8:14]

#### Endpunkte und Sicherheitsmodi

Der SIMATIC NET SR-OPC-UA-Server unterstützt Endpunkte mit dem nativen binären TCP-Protokoll und ermöglicht Authentisierung über Zertifikate und eine verschlüsselte Übertragung.

Der Discovery-Dienst auf dem angesprochenen Host meldet die Endpunkte der Server, sowie deren Sicherheitsanforderungen und Protokollunterstützung.

Die Server-URL "opc.tcp://<hostname>:55102" des SR-OPC-UA-Servers bietet folgende Endpunkte:

- Endpunkt im Sicherheitsmodus "SignAndEncrypt":

Zur Kommunikation mit dem Server werden Signierung und Verschlüsselung gefordert.

Die Kommunikation ist durch Zertifikataustausch und Passworteingabe geschützt.

Zusätzlich zum Sicherheitsmodus wird die Sicherheitsrichtlinie Basic128Rsa15 angezeigt.

- Endpunkt im Sicherheitsmodus "keiner":

In diesem Modus werden keine Sicherheitsfunktionen vom Server gefordert (Sicherheitsrichtlinie "None").

Weitere Details zu den Sicherheitsfunktionen finden Sie im Kapitel "OPC-UA-Schnittstelle programmieren (Seite 508)".

Die Sicherheitsrichtlinien "Basic128Rsa18" und "None" finden Sie in der UA-Spezifikation der OPC Foundation unter folgender Internet-Adresse:

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

Weitere Informationen finden Sie auf folgender Internetseite:

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

## Die OPC-UA-Discovery des OPC Scout V10

Der OPC Scout V10 ermöglicht das Öffnen des OPC-UA-Discovery-Dialogs zur Übernahme von UA-Endpunkten in den Navigationsbereich des OPC Scout V10.

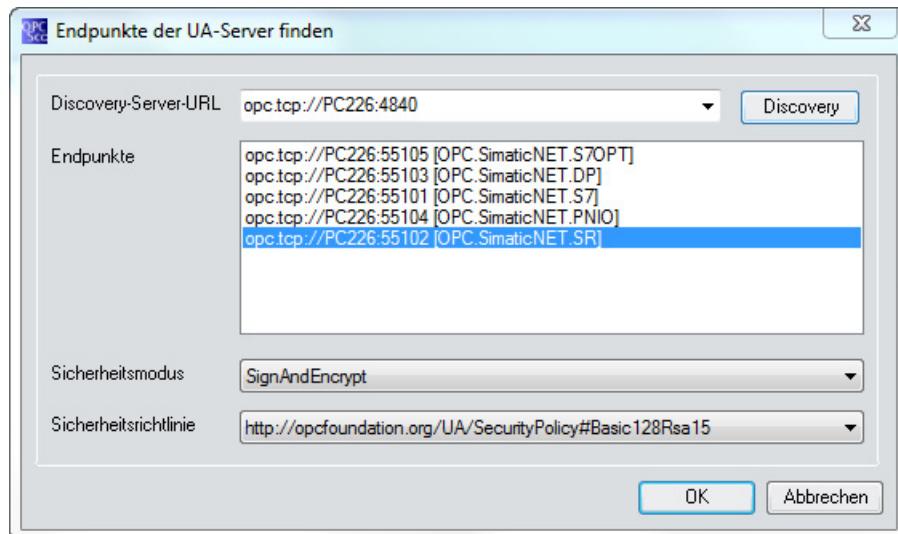


Bild 2-57 Das Dialogfeld "Endpunkte der UA-Server finden" des OPC Scout V10

Der SR-OPC-UA-Server kann über den OPC-UA-Discovery-Dienst gefunden werden. Zur Eingabe siehe oben unter "Server-URL".

Der OPC Scout V10 beinhaltet eine Liste der OPC-UA-Endpunkte. Der Discovery-Dienst auf dem angesprochenen Host meldet dann die registrierten OPC-UA-Server sowie deren Ports und Sicherheitsmodi.

Weitere Details finden Sie in der Online-Hilfe des OPC Scout V10.

#### 2.10.4 Protokoll-ID

Die Protokoll-ID für das SEND/RECEIVE-Protokoll unter OPC UA lautet "SR".

#### 2.10.5 Welche Namensräume bietet der SR-OPC-UA-Server an?

Der SR-OPC-UA-Server bietet folgende Namensräume an:

Namensraum-Index	"Bezeichner" (Namensraum-URI) / Kommentar
0	"http://opcfoundation.org/UA/" von der OPC Foundation spezifiziert
1	"urn:Siemens.Automation.SimaticNET.SR:(GUID)" Eindeutiger Bezeichner des lokalen performanten SR-OPC-UA-Servers.
2	"SRTYPES:" Definitionen für SR-spezifische Objekttypen.
3	"SR:" Bezeichner des lokalen performanten SR-OPC-UA-Servers mit neuer vereinfachter Syntax (durchsuchbar und verwendbar mit UA)
4	"SRCOM:" Bezeichner des Servers mit alter Syntax, SR-OPC-DA-kompatibel (mit UA verwendbar aber nicht durchsuchbar)

Die Namensraum-Indizes 0 und 1 sind reserviert und in Ihrer Bedeutung von der OPC Foundation spezifiziert.

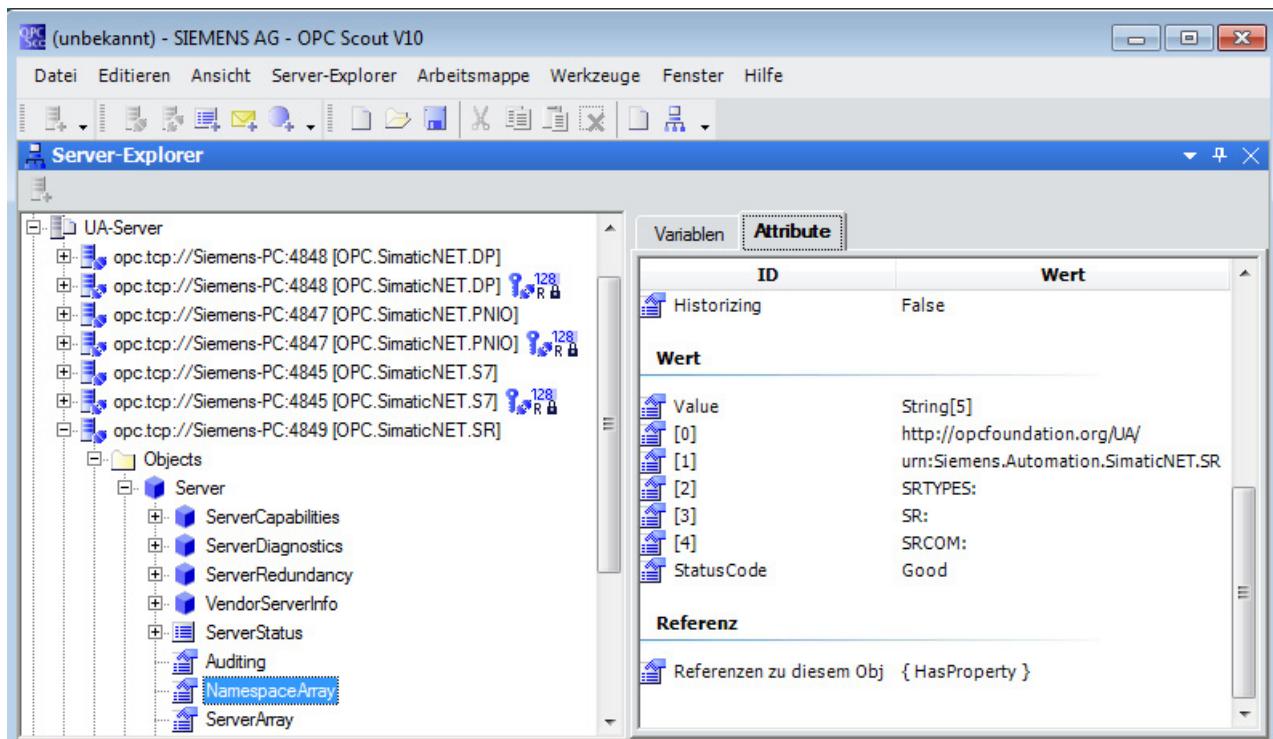


Bild 2-58 Anzeige der SR-OPC-UA-Namensräume mittels der Browse-Funktion des OPC Scout V10

## 2.10.6 Verbindungsnamen

Der Verbindungsname ist der in STEP 7 projektierte Name zur Identifikation der Verbindung.

Es werden folgende Verbindungstypen vom OPC-UA-Server unterstützt:

- ISO-Transportverbindung
- ISO-on-TCP-Verbindung
- TCP-Verbindung

### Art der Verbindung

Die Art der SR-Zugriffsmöglichkeiten, die über eine SR-Verbindung möglich ist, wird in STEP 7 eingestellt. Die Verbindung kann entweder:

- nur Fetch,
- nur Write oder
- nur Send/Receive

## Beispiele für Verbindungsnamen

Typische Beispiele sind:

- ISO-on-TCP-Verb-1
- ISO-on-TCP connection1

## 2.10.7 Die Nodeld

### Identifikation einer SR-Prozessvariable

Die Nodeld identifiziert mit Hilfe des folgenden Tupels eine SR-Prozessvariable eindeutig:

- Namensraum-Index
- Bezeichner (Zeichenfolge, numerischer Wert)

### Beispiele

- Nodeld:
  - Namensraum-URI:  
*SR*:  
(= Namensraum-Index 3) für Siemens.Automation.SimaticNET.SR
  - Bezeichner:  
*SRVerbindungsname.send,100.0,b,100*
- Nodeld:
  - Namensraum-URI:  
*SRCOM*:  
(= Namensraum-Index 4) für OPC.SimaticNET; die Syntax ist SR-OPC-DA-kompatibel
  - Bezeichner:  
*SR:[SRVerbindungsname]send,b0,100*

### Wie verhält sich der neue auf OPC UA angepasste Namensraum?

Die OPC-UA-Sicht ist auf Automatisierungsobjekte und auch auf verschiedene Eigenschaften der Objekte bezogen. OPC UA greift nicht mehr alleine auf Items zu, sondern auf Objekte und deren Unterobjekte.

- Datenvariablen sind beispielsweise Unterobjekte eines SR-Verbindungsobjekts. Attribute und Properties definieren die Objekte näher.
- Ein OPC-Data-Access-Item für den Bausteinzugriff entspricht dabei am ehesten einer OPC-UA-Datenvariablen.

Den qualifizierten Bezeichnern der Nodelds kommt unter OPC UA eine größere Bedeutung als unter OPC Data Access zu. Jeder einzelne Zugriff auf ein Objekt, Unterobjekt, Property und Attribut erfolgt über dessen Nodeld.

Unter anderem für die Unterstützung durch lokale Sprachen sieht OPC UA den Anzeigenamen vor. So kann ein und dasselbe Objekt beispielsweise in unterschiedlichen Sprachumgebungen, die der OPC-UA-Client vorgibt, unterschiedlich durchsucht werden, wobei aber jedes Mal die selbe Nodeld präsentiert wird. Der Anzeigename wird analog zur jeweiligen Nodeld gewählt. Die Texte des gesamten Namensraums sind in Englisch.

### Syntax der SR-OPC-UA-Datenobjekte

In der Regel sind die Nodelds der OPC-UA-Objekte im SR-UA-Server nach folgender Struktur aufgebaut:

<verbindungsobjekt>."<unterobjekt>".<property>

Ein Unterobjekt kann weitere Unterobjekte beinhalten.

Eine nicht interpretierbare Nodeld wird mit einem Fehler zurückgewiesen. Die Groß- oder Kleinschreibung der Buchstaben "A-Z" wird bei allen Items ignoriert.

### Symbolische Objektdarstellung

Die OPC-UA-Spezifikation empfiehlt zur hierarchischen Beschreibung des Adressraums eine einheitliche Symboldarstellung. Folgende Symbole werden in diesem Dokument verwendet:

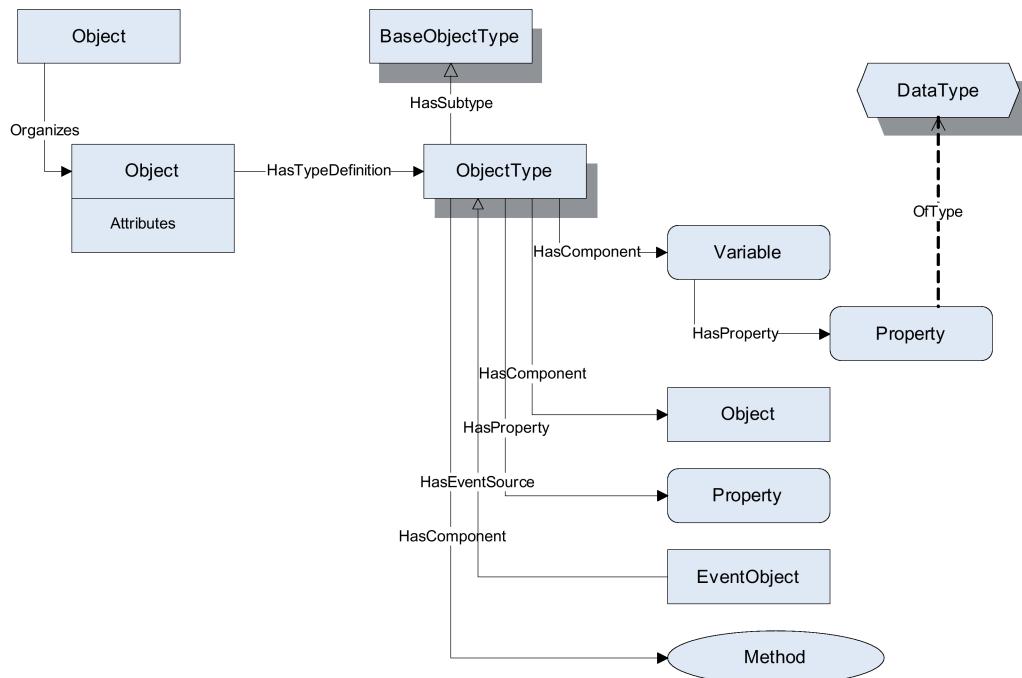


Bild 2-59 Symbole des OPC-UA-Adressraums

## 2.10.8 Datenvariablen für S5-Datenbausteine und Bereiche (S5-kompatible Kommunikation)

Das Lesen und Schreiben von Datenvariablen für S5-Datenbausteine und Bereiche (S5-kompatible Kommunikation) setzt die Projektierung einer Fetch- bzw. Write Verbindung voraus.

Die Datenvariablen auf einer Fetch-Verbindung werden nur gelesen. Auf eine Write-Verbindung werden die Datenvariablen nur geschrieben. Wenn Datenbausteine eines Kommunikationspartners sowohl gelesen als auch geschrieben werden sollen, dann müssen Sie zwei entsprechende Verbindungen projektieren, die vom OPC-UA-Server jedoch völlig unabhängig verwaltet werden.

### Syntax der Datenvariablen für S5-Datenbausteine und Bereiche

Vereinfachte Syntax der Prozessvariablen der SR-OPC-UA-Node-ID für das Lesen und Schreiben von Datenvariablen:

Namensraum-URI: SR: (Namensraum-Index: 3)

### Syntax

Es gibt zwei Möglichkeiten:

```
<FETCHVerbindungsname>.<Datenvariable>.{<Offset>{,<SRTyp>{,<Anzahl>} }}
```

```
<WRITEVerbindungsname>.<Datenvariable>.{<Offset>{,<SRTyp>{,<Anzahl>} }}
```

### Erklärungen

#### <FETCHVerbindungsname> und <WRITEVerbindungsname>

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

#### <Datenvariable>

Datenvariable für S5-kompatible Kommunikation.

Datenvariable		Hinweis
DB<db>	unsigned8	Datenbaustein
DX<dx>	unsigned8	Erweiterter Datenbaustein
DE<de>	unsigned8	Datenbaustein im Externspeicher
I		Eingang
Q		Ausgang
P		Ein- und Ausgang für Peripherie
QB		Erweiterte Peripherie
M		Merker
C		Zähler
T		Timer

- **Informationen zu den Bereichen der Datenvariablen "db", "dx" und "de"**

Nach den Kennungen "DB" bzw. "DX" und "DE" ist zusätzlich noch die Bausteinnummer anzugeben, z.B. DB10, die protokollspezifische Adressierung unterstützt jedoch nur Bausteinnummern von 0-255 (8 Bit). Das Protokoll erlaubt in diesen Bereichen nur die Adressierung von ganzen Worten (16 Bits). Der Defaultdatentyp dieser Bereiche ist daher "w".

- Bei Fetch-Verbindungen können Sie auch Bereiche adressieren, die nicht ganzen Worten entsprechen. Daher können Sie z.B. Bytes bzw. Bits (inkl. Feldern) adressieren, außerdem 16- und 32 Bit Datentypen mit ungeradem Byteoffset. Der OPC-UA-Server fordert allerdings vom Kommunikationspartner immer ganze Worte an und extrahiert aus diesen die entsprechenden Daten.
- Bei Write-Verbindungen können Sie aus Konsistenzgründen grundsätzlich nur Bereiche adressieren, die ganzen Worten entsprechen. Daher müssen Sie gerade und ganzzahlige Blockoffsets angeben und nur Bereiche, die einem ganzzahligen Vielfachen von 16 Bits entsprechen. Bei S7-Kommunikationspartnern können Sie auch ungerade Blockoffsets angeben.

Beispiele:

Beispiel-Syntax	FETCH	WRITE
SRVerbindungsname.db10.1,b	nur lesbar	nicht verfügbar
SRVerbindungsname.db10.4, b,8	nur lesbar	nur schreibbar
SRVerbindungsname.db10.3,x1	nur lesbar	nicht verfügbar
SRVerbindungsname.db10.5,x4,6	nur lesbar	nicht verfügbar
SRVerbindungsname.db10.6,x0,32	nur lesbar	nur schreibbar
SRVerbindungsname.db10.7,w	nur lesbar	nur schreibbar
SRVerbindungsname.db10.8,w	nur lesbar	nur schreibbar

- **Informationen zu den Bereichen der Datenvariablen "m", "i", "q", "p" und "qb"**

Das Protokoll erlaubt in diesen Bereichen nur die Adressierung von ganzen Bytes (8 Bits). Bei den meisten Datentypen gibt es daher keine Einschränkungen, Ausnahme: Bits.

- Bei Fetch-Verbindungen können Sie auch Bereiche adressieren, die nicht ganzen Bytes entsprechen. Daher können Sie insbesondere Bits (inkl. Feldern) adressieren. Der OPC-UA-Server fordert allerdings vom Kommunikationspartner immer ganze Bytes an und extrahiert aus diesen die entsprechenden Daten.
- Bei Write-Verbindungen können Sie aus Konsistenzgründen grundsätzlich nur Bereiche adressieren, die ganzen Bytes entsprechen. Daher müssen Sie bei Bits ganzzahlige Blockoffsets angeben und nur Bereiche, die einem ganzzahligen Vielfachen von 8 Bits entsprechen.

Beispiele:

Beispiel-Syntax	FETCH	WRITE
SRVerbindungsname.m.1,b	nur lesbar	nur schreibbar
SRVerbindungsname.m.4,b,7	nur lesbar	nur schreibbar
SRVerbindungsname.m.3,x1	nur lesbar	nicht verfügbar

Beispiel-Syntax	FETCH	WRITE
SRVerbindungsname.m.5,x4,6	nur lesbar	nicht verfügbar
SRVerbindungsname.m.6,x0,32	nur lesbar	nur schreibbar
SRVerbindungsname.m.7,w	nur lesbar	nur schreibbar
SRVerbindungsname.m.8,w	nur lesbar	nur schreibbar

- **Informationen zu den Bereichen der Datenvariablen "c"**  
Die Adressangabe besteht aus der Zählernummer.
- **Informationen zu den Bereichen der Datenvariablen "t"**  
Die Adressangabe besteht aus der Timernummer.

**<Offset>**

Byteadresse im Datensatz für das Element, das angesprochen werden soll. Bei OPC UA ist dieses Offset immer ein Byte-Offset.

**<SRTyp>**

Datentyp.

Der Datentyp wird im OPC-UA-Server in den entsprechenden OPC-UA-Datentyp umgewandelt.

Tabelle 2- 7 Datentypbeschreibung

Datentyp	OPC-UA-Datentyp	Hinweis S7-Datentyp
x<Bitadresse>	Boolean	Bit (bool) Zusätzlich zum Byte-Offset im Bereich ist die <Bitadresse> im jeweiligen Byte anzugeben. Wertebereich 0...7
b	Byte ByteString	Byte (unsigned) Wird als Defaultwert verwendet, falls kein <SRTyp> angegeben ist. OPC UA kennt kein "Byte[]", sondern verwendet hierzu den skalaren Datentyp "ByteString".
w	UInt16	Wort (unsigned)
dw	UInt32	Doppelwort (unsigned)
c	SByte	Byte (signed)
i	Int16	Wort (signed)
di	Int32	Doppelwort (signed)
r	Float	Fließkomma (4 Byte)
s5r	Float	S5-codierter Real
c	UInt16	Nur im Zählerbereich
t	UInt16	Nur im Timerbereich

**<Anzahl>**

Anzahl der Elemente. Der Datentyp der Variable ist ein Feld mit Elementen des angegebenen Formats. Die Angabe einer Anzahl von Feldelementen führt immer zur Bildung

eines Feldes vom entsprechenden Typ, auch wenn nur ein einziges Feldelement adressiert wird.

### Beispiele für Datenvariablen für S5-Datenbausteine und Bereiche

- **FETCHVerbindungsname.db10.10,w**  
bezeichnet ein Datenwort des Datenbausteins 10 ab Byteadresse 10.
- **WRITEVerbindungsname.q.3**  
bezeichnet ein Ausgangsbyte ab Byteadresse 3 (BYTE-Default).
- **FETCHVerbindungsname.i.0,x0**  
bezeichnet ein Eingangsbit ab Byteadresse 0, Bit 0.
- **WRITEVerbindungsname.m.3,x4,16**  
bezeichnet ein Feld von Merkerbits ab Byteadresse 3, Bit 4 (nur lesbar)
- **FETCH/WRITEVerbindungsname.t.22**  
bezeichnet den Timer 22 (Default)

### 2.10.9 Blockorientierte Dienste

Die blockorientierten Dienste ermöglichen eine programmgesteuerte Übertragung größerer Datenblöcke. Diese Dienste werden auch als SEND/RECEIVE-Dienste bezeichnet. Die Übertragung mit dem OPC-UA-Server wird durch Variablen realisiert:

- Variablen, die Datenblöcke empfangen
- Variablen, die Datenblöcke senden

Eine Standardgröße der Datenblöcke wird in der Projektierung festgelegt, beim Senden von Variablen kann die Größe eingeschränkt werden. Ein Teilzugriff innerhalb der Datenblöcke ist möglich.

### Fest definierte Variablennamen

Folgende Variablennamen sind für jede Verbindung fest definiert:

- *receive*
- *send*

### Syntax der Prozessvariablen für blockorientierte Dienste

Vereinfachte Syntax der Prozessvariablen der SR-OPC-UA-Node-ID, für das Lesen und Schreiben von Datenvariablen:

Namensraum-URI: SR: (Namensraum-Index: 3)

### Syntax

Es gibt folgende Möglichkeiten:

```
<SRVerbindungsname>.send{,<Block>} { .<Offset>{,<SRTyp>{,<Anzahl>}} }
```

---

```
<SRVerbindungsname>.receive{.<Offset>{,<SRTyp>{,<Anzahl>}}}
```

## Erklärungen

### **<SRVerbindungsname>**

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

### **send**

Ein Sendepuffer, der an den Verbindungspartner übertragen werden kann.

Für die Größe des Sendepuffers ist durch die Projektierung ein Defaultwert festgelegt. Der Puffer wird immer als Feld von Bytes geliefert (OPC-UA-Datentyp "Bytestring").

Schreibzugriffe auf diese Variable bewirken, dass der Sendepuffer an den Partner übertragen wird.

---

### Hinweis

Diese Variable und daraus abgeleitete Variablen dürfen nicht gelesen und nicht aktiviert werden. Lesezugriffe auf diese Variable werden unter Umständen vom Partnergerät mit einem Verbindungsabbau quittiert.

### **receive**

Zuletzt vom Partner empfangener Datenpuffer.

Die Struktur des Datenpuffers ist nicht vorgegeben. Daher wird der Puffer immer als Feld von Bytes geliefert (OPC-UA-Datentyp "Bytestring").

---

### Hinweis

Wenn das Empfangen von Datenblöcken über den OPC-UA-Beobachtungsdienst gelesen wird (receive als MonitoredItem), dann kann über die Konfiguration des Beobachtungsdienstes das Verhalten beim Empfang von Datenblöcken eingestellt werden. Vorausgesetzt der DataChangeTrigger ist im Beobachtungsdienst auf OpcUa\_DataChangeTrigger\_StatusValueTimestamp gesetzt, werden auch neu empfangene Datenblöcke mit gleichen Daten an den OPC-UA-Client gemeldet. Dadurch wird es einem OPC-UA-Client ermöglicht, auch unveränderte gesendete Datenpuffer vom Partner zu erhalten. Die DataChange Notification erfolgt nicht schneller als die ausgehandelte Aktualisierungszeit (Update rate). Stellen Sie also hierfür immer Aktualisierungszeiten schneller als die Senderate der send/receive-Daten ein.

---

### Hinweis

Die RECEIVE-Variable entspricht einem Empfangspuffer. Deshalb kann die Variable nur gelesen werden. Bei einem Lesezugriff auf diese Nodeld wird explizit ein Empfangspuffer im Kommunikationssystem bereitgestellt. Wird nicht innerhalb einer Fehlerwartezeit für diesen Puffer ein Datenblock empfangen, so wird ein Fehler erzeugt. Deshalb sollten Sie diese Variablen nur beobachten (Zugriff über UA-Subscription).

---

### **<Block>**

Größe des Sendepuffers in Byte.

Sie können <bl> verwenden, wenn auf einer Verbindung Sendepuffer verschiedener Größe

verwendet werden sollen. Wenn **<bl>** weggelassen wird, wird die in der Projektierung eingetragene Puffergröße verwendet.

Bei TCP/IP native mit abgeschaltetem Miniprotokoll (siehe Projektierungswerzeug SIMATIC STEP 7 oder SIMATIC NCM PC) ist die Angabe der Größe des Sendepuffers nicht möglich. In diesem Fall wird die in der Projektierung eingetragene Puffergröße verwendet.

Die Größe des Sendepuffers ist von der Projektierung abhängig. Beachten Sie, dass die **<Anzahl>** multipliziert mit der Größe des **<SRTyp>** in Byte der Variablen nicht größer als der Sendepuffer **<bl>** gewählt werden darf.

#### **<SRTyp>**

Ein DP-Datentyp wird im OPC-UA-Server in den entsprechenden OPC-UA-Datentyp umgewandelt. Die folgende Tabelle listet den Typ-Bezeichner und den entsprechenden OPC-Datentyp auf, in dem der Variablenwert dargestellt werden kann.

Datentyp	OPC-UA-Datentyp	Hinweis
x<Bitadresse>	Boolean	Bit (bool) Zusätzlich zum Byte-Offset im Bereich ist die <Bitadresse> im jeweiligen Byte anzugeben. Wertebereich 0...7
b	Byte	Byte (unsigned)
	ByteString	Wird als Defaultwert verwendet, falls kein <DPTyp> angegeben ist. OPC UA kennt kein "Byte[]", sondern verwendet hierzu den skalaren Datentyp "ByteString".
char	Byte	Integer
w	UInt16	Wort (unsigned)
int	Integer	
dw	UInt32	Doppelwort (unsigned)
di	Int32	Doppelwort (signed)
r	Float	Fließkomma (4 Byte)
s5r	Fließkomma, S5-Darstellung	Single

#### **<Offset>**

Bytaddress im Datensatz für das Element, das angesprochen werden soll.

**<Anzahl>**

Anzahl der Elemente. Der Datentyp der Variable ist ein Feld mit Elementen des angegebenen Formats. Die Angabe einer Anzahl von Feldelementen führt immer zur Bildung eines Feldes vom entsprechenden Typ, auch wenn nur ein einziges Feldelement adressiert wird.

**Hinweis**

Bitte beachten Sie folgende Hinweise:

- Durch die optionale Angabe von Typ, Adresse und Anzahl können Sie strukturiert auf Teilbereiche von Datenblöcken zugreifen.
- Variablen für Sendedaten oder Empfangsdaten mit unterschiedlicher Länge oder unterschiedlichem Verbindungsnamen verfügen über unabhängige Speicherbereiche.
- Der Sendedatenpuffer wird allokiert und mit Null initialisiert, wenn ein Item für einen unabhängigen Speicherbereich angelegt wird. Ein Schreibauftrag auf ein send-Item wird in einen internen Schreibpuffer geschrieben und übertragen.
- Die Übertragung der Datenblöcke erfolgt azyklisch. Es wird immer der vollständige Sendedatenblock übertragen. Dies gilt auch bei Subelementzugriff oder wenn mehrere Clients gleichzeitig dieses Item beschreiben.

**Beispiele für Prozessvariablen für blockorientierte Dienste**

Hier finden Sie Beispiele, die die Syntax von Variablennamen für blockorientierte Dienste verdeutlichen.

**Empfangsvariablen**

- **SRVerbindungsname.receive.4,w,6**  
bezeichnet 6 Datenworte ab Offset 4 im Empfangspuffer.
- **SRVerbindungsname.receive.7,dw**  
bezeichnet ein Doppelwort ab Offset 7 im Empfangspuffer
- **SRVerbindungsname.receive.0,r,2**  
bezeichnet ein Feld mit 2 Fließkommawerten ab Offset 0 im Empfangspuffer.

**Sendvariablen**

- **SRVerbindungsname.send,30,7,dw**  
bezeichnet ein Doppelwort ab Byte 7 in einem 30 Byte großen Sendepuffer. Wenn der Standardwert der Größe des Sendpuffers nicht gleich 30 ist, greift diese Variable auf einen separaten Puffer zu.
- **SRVerbindungsname.send,256,6,b,20**  
bezeichnet ein Feld mit 20 Bytes ab Offset 6 in einem Sendepuffer mit Standardgröße. Die Standardgröße des Sendepuffers wird in der Projektierung festgelegt.
- **SRVerbindungsname.send,8,0,di**  
bezeichnet ein vorzeichenbehaftetes Doppelwort ab Adresse 0 in einem Sendepuffer mit der Größe 8.

## 2.10.10 SR-spezifische Informationsvariablen

Mit den SEND/RECEIVE-spezifischen Informationsvariablen können Sie Informationen über den Status der Verbindung abfragen.

### Syntax der Informationsvariablen für offene Kommunikationsdienste (SEND/RECEIVE)

Vereinfachte Syntax der Prozessvariablen der SR-OPC-UA-Node-ID, für das Lesen und Schreiben von SR-spezifischen Informationsvariablen:

Namensraum-URI: SR: (Namensraum-Index: 3)

#### Klassische Syntax

<SRVerbindungsname>.statepath.statepath

#### Erklärungen

**<SRVerbindungsname>**

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

**statepath.statepath**

statepath	Zustand einer Kommunikationsverbindung zum Partnergerät Der Wert der Variable wird als Zahl ausgelesen und kann durch zusätzliches Auslesen des zugehörigen Enumstring {UNKNOWN, DOWN, UP, RECOVERY, ESTABLISH} einem Text zugeordnet werden. Variable vom UA-Typ MultistateDiscreteType, nur lesbar		
	1	DOWN	Verbindung ist nicht aufgebaut
	2	UP	Verbindung ist aufgebaut
	3	RECOVERY	Verbindung ist nicht aufgebaut. Es wird versucht, die Verbindung aufzubauen.
	4	ESTABLISH	Für zukünftige Erweiterungen reserviert
	0	UNKNOWN	Für zukünftige Erweiterungen reserviert

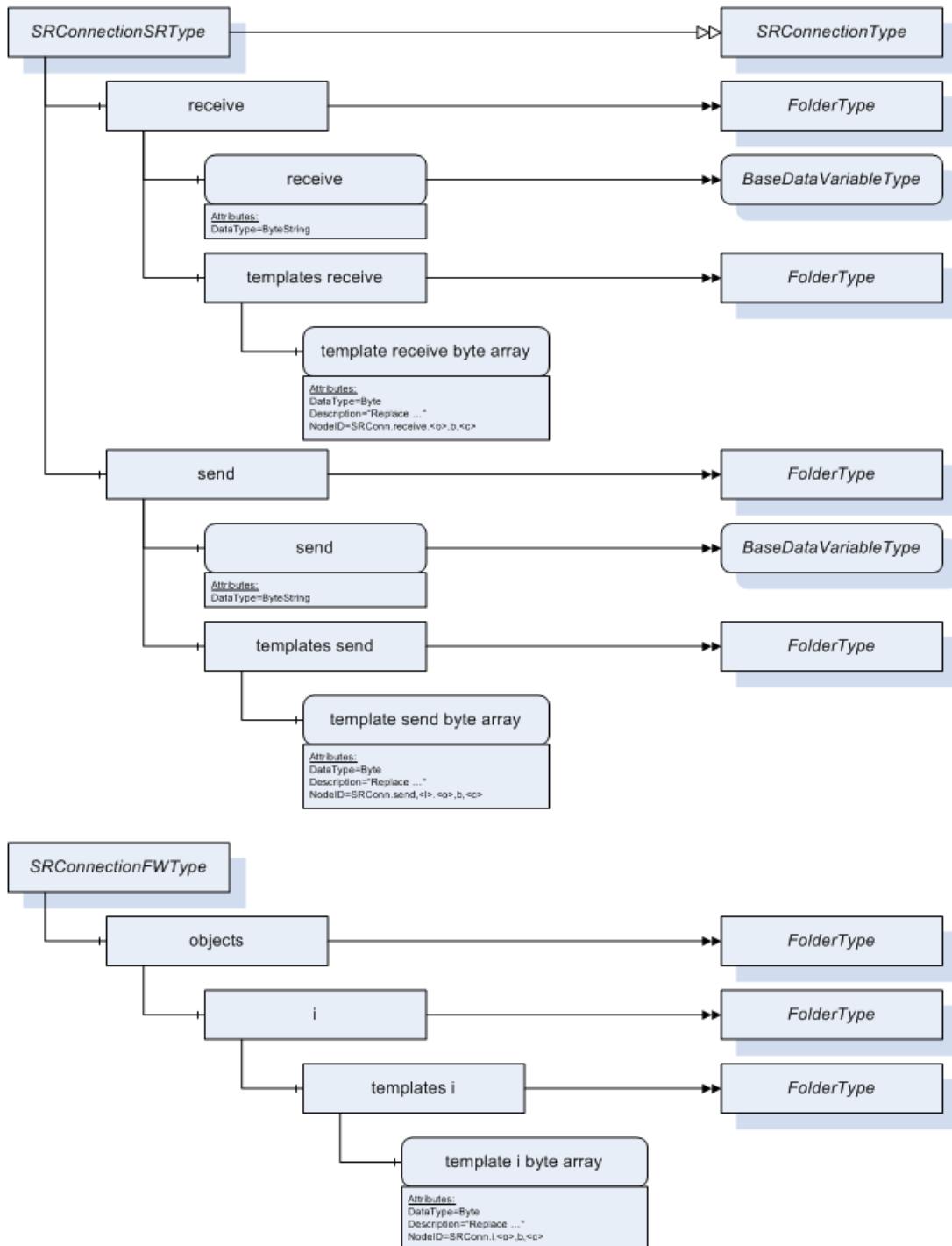
## 2.10.11 SR-OPC-UA-Template-Datenvariablen

Sie haben mit den Prozessvariablen für das OPC-UA-SR-Protokoll flexible Einstellungsmöglichkeiten, um die Prozessdaten Ihrer Anlage in den gewünschten Datenformaten zu erhalten.

Die Vielfalt der Adressierungsmöglichkeiten lässt sich allerdings nicht in einen vollständig durchsuchbaren Namensraum fassen. Bereits ein Datenbaustein mit der Länge eines einzelnen Bytes besitzt etwa 40 verschiedene Datenformatoptionen – angefangen vom Byte, SByte, Felder mit einem Element davon, einzelne Bits, Felder von Bits mit bis zu 8 Feldelementen an unterschiedlichen Bitoffsets beginnend.

## 2.10 Offene Kommunikationsdienste (SEND/RECEIVE) mit OPC UA über Industrial Ethernet

Der OPC-UA-Server unterstützt den Anwender deshalb mit den Template-Datenvariablen im SR-Namensraum. In einem für einen OPC-UA-Client typischen Texteingabefeld können diese Templates durch Ändern einiger weniger Zeichen in gültige ItemIDs verwandelt werden.



**Hinweis**

Die Verwendbarkeit von OPC-UA-SR-Template-Datenvariablen kann im Konfigurationsprogramm "Kommunikations-Einstellungen" unter "OPC-Protokollauswahl" > Klicken des Pfeilsymbols bei "SR" aktiviert und deaktiviert werden.

**Template-Datenvariablen innerhalb der Browse-Hierarchie**

Die Template-Datenvariablen sind neben den ihnen entsprechenden Ordnern in der Namensraum-Darstellung eingesortiert, so dass sie bei Bedarf leicht genutzt werden können.

**Syntax der Template-Datenvariablen**

Es gibt folgende Möglichkeiten:

- <SRVerbindungsname>send,<bl>.<o>,<SRTypTemplate>,<c>
- <SRVerbindungsname>receive.<o>,<SRTypTemplate>,<c>
- <FETCHVerbindungsname>.<DatenvARIABLE>.<o>,<SRTypTemplate>,<c>
- <WRITEVerbindungsname>.<DatenvARIABLE>.<o>,<SRTypTemplate>,<c>

**Erklärungen**

**<SRVerbindungsname>, <FETCHVerbindungsname> und <WRITEVerbindungsname>**  
Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt.

**<bl>**

Der Platzhalter für die Länge des Sendepuffers.

**<o>**

Der Platzhalter für den Byteoffset.

**<SRTypTemplate>**

Ein SR-Template-Datentyp wird im OPC-UA-Server in den entsprechenden OPC-UA-Datentyp umgewandelt. Die folgende Tabelle listet den Datentyp und den entsprechenden OPC-UA-Datentyp auf, in dem der Variablenwert dargestellt werden kann.

Datentyp	OPC-UA-Datentyp	Hinweis
x<Bitadresse>	Bit (Boolean)	VT_BOOL
b	Byte (unsigned8)	VT_UI1
char	Byte (signed8)	VT_I1
w	Wort (unsigned16)	VT_UI2
int	Wort (signed16)	VT_I2
dw	Doppelwort (unsigned32)	VT_UI4
di	Doppelwort (signed32)	VT_I4

Datentyp	OPC-UA-Datentyp	Hinweis
r	Fließkomma, IEEE-Darstellung	VT_R4
s5r	Fließkomma, S5-Darstellung	VT_R4

**<Datenvariable>**

Datenvariable für S5-kompatible Kommunikation.

Datenvariable		Hinweis
db<db>	unsigned8	Datenbaustein
dx<dx>	unsigned8	erweiterter Datenbaustein
de<de>	unsigned8	erweiterter Datenbaustein
i		Eingang
q		Ausgang
p		Ein- und Ausgang für Peripherie
qb		erweiterte Peripherie
m		Merker
c		Zähler
t		Timer

**<c>**

Platzhalter für Anzahl der Elemente.

**Beispiele:**

Nodeld	BrowseName	Beschreibung
SRVerbindungsname.send,<bl>.<o>,b	template send byte	<bl> Länge des Sendepuffers <o> Offset
SRVerbindungsname.receive.<o>,b	template receive byte	<o> Offset
FETCHVerbindungsname.i.<o>,x<bit>,<c>	template i bit array	<o> Offset <bit> Bitoffset (0..7) <c> Größe des Feldes
WRITEVerbindungsname.db<db>.<o>,x<bit>,<c>	template db bit array	<db> Nummer des Datenbausteins <o> Offset <bit> Bitoffset (0..7) <c> Größe des Feldes

## 2.11 SNMP-Kommunikation über Industrial Ethernet

### Prozessvariablen für SNMP

SNMP-Daten sind in erster Linie Diagnosevariablen (bei Schreibzugriff auch Geräteparameter).

Der SNMP-OPC-Server von SIMATIC NET bietet folgende Variablen an:

- Prozessvariablen
- Informationsvariablen
- Trap-Variablen

Weitere Informationen zum SNMP OPC-Server:

(<http://www.automation.siemens.com/mcms/industrial-communication/de/ie/software/netzwerkmanagement/snmp-opc-server/Seiten/snmp-opc-server.aspx>)

### 2.11.1 Protokoll-ID

#### Protokoll-ID

Die Protokoll-ID für das SNMP-Protokoll lautet "SNMP".

### 2.11.2 Datentypen des SNMP-Protokolls

#### Abbildung auf verfügbare Datentypen

Das SNMP-Protokoll verwendet bei der Deklaration der MIB Objekte eigene Datentypen. Der SNMP-OPC-Server bildet die Datentypen des SNMP-Protokolls auf die an der OPC-Schnittstelle verfügbaren Datentypen wie folgt ab:

SNMP-Datentyp Code	Kanonischer OLE-Datentyp	Visual Basic-Typ	Bedeutung
ASN_INTEGER 02h	VT_I4	Long	signed long (4 Byte)
ASN_INTEGER32 02h	VT_I4	Long	signed long (4 Byte)
ASN_UNSIGNED32 47h	VT_UI4	Double	unsigned long (4 Byte)
ASN_OCTETSTRING 04h	VT_BSTR	String	Falls keine druckbaren Zeichen, Notation: <i>ostring:xx.xx.xx.xx</i>

SNMP-Datentyp Code	Kanonischer OLE-Datentyp	Visual Basic-Typ	Bedeutung
ASN_OBJECTIDENTIFIER 06h	VT_BSTR	String	In der MIB-Notation: .a.b.c.d.e.
ASN_IPADDRESS 40h	VT_BSTR	String	In dezimaler Notation: 129.168.0.4
ASN_COUNTER32 41h	VT_UI4	Double	unsigned long (4 Byte)
ASN_GAUGE32 42h	VT_UI4	Double	unsigned long (4 Byte)
ASN_TIMETICKS 43h	VT_UI4	Double	unsigned long (4 Byte)
ASN_OPAQUE 44h	VT_ARRAY of VT_UI1	VT_ARRAY of VT_UI1	unsigned Byte

## 2.11.3 Prozessvariablen für SNMP-Variabldienste

### Variablen und Geräteprofile

SNMP-Variablen bezeichnen die Variablen im Partnergerät. Diese Variablen werden dem OPC-Server anhand der Geräteprofile bekannt gemacht. Diese Geräteprofile können mit Hilfe des MIB-Compilers und den entsprechenden MIB-Dateien des Gerätes erstellt werden. Standardgeräteprofile sind im Lieferumfang des Projektierungswerkzeugs enthalten. Anschließend wird die Projektierung auf den PC mit dem OPC-Server geladen, damit die Variablen dem OPC-Server zur Verfügung stehen.

### Syntax

---

#### Hinweis

Abweichend von der Syntax der anderen Protokolle wird bei SNMP folgende Syntax verwendet:

---

SNMP : [<Teilnehmername>]<Objektname>

---

### Erklärungen

#### SNMP

SNMP-Protokoll für den Zugriff auf die Prozessvariablen (MIB-Objekte) und Trap-Variablen.

#### <Teilnehmername>

Der Teilnehmername wird bei der Projektierung (Anlagenkonfiguration) festgelegt und ist eindeutig.

#### <Objektname>

Symbolischer Name für das MIB-Objekt des Partnergeräts.

## Beispiel

*SNMP:[OSM]sysName*

Abfrage des MIB-Objektes sysName. Der Teilnehmername wurde in diesem Beispiel in der Anlagenkonfiguration als OSM projektiert.

## 2.11.4 SNMP-spezifische Informationsvariablen

### Einleitung

Die folgenden Abschnitte beschreiben, welche Variablen der SNMP-OPC-Server zur Verfügung stellt:

- Variablen für das Kommunikationssystem und die Verbindungsstzände
- Variablen für den SNMP-OPC-Server

### Informationen zum SNMP-Gerät

Der OPC-Server stellt Variablen zur Verfügung, mit denen Sie Informationen über das SNMP-Gerät (z.B. Port; Switch) abfragen können.

Sie können folgende Informationen abfragen:

- Den eingegebenen Kommentar bei der Projektierung
- Die projektierte IP Adresse des Partnergeräts
- Den Status der Verbindung zum Partnergerät
- Zusatzinformationen bei nicht SNMP-fähigen Geräten

### Syntax

`SNMP: [<Teilnehmername>]<Informationsparameter>`

### Erklärungen

#### SNMP

SNMP-Protokoll für den Zugriff auf die Prozessvariablen (MIB-Objekte) und Trap-Variablen.

#### <Teilnehmername>

Der Teilnehmername wird bei der Projektierung (Anlagenkonfiguration) festgelegt.

#### <Informationsparameter>

Es sind folgende Informationsparameter definiert:

<code>&amp;description()</code>	Der bei der Projektierung im Kommentarfeld eingegebene Text. VT_BSTR, Read-Only
<code>&amp;ipaddress()</code>	Die IP Adresse des Partnergerätes. VT_BSTR, Read-Only.

&statepath()	Verbindungszustand zum Partnergerät. Wert in Textform (VT_BSTR, Read-Only):	
	DOWN	Das Gerät kann nicht mit einem Ping erreicht werden bzw. es bestehen keine aktiven Lese- oder Schreibaufträge.
	UP	Es konnten erfolgreich Aufträge zu diesem Gerät abgewickelt werden.
	RECOVERY	Das Gerät kann nicht mit einem Ping erreicht werden bzw. der letzte Auftrag wurde mit einem Fehler abgeschlossen, der auf einen Kommunikationsabbruch hindeutet, und es bestehen weitere aktive Aufträge zum Gerät.
&statepathval()	Verbindungszustand zum Partnergerät. Wert als Integer. VT_UI1, Read-Only.	
	1	Das Gerät kann nicht mit einem Ping erreicht werden bzw. es bestehen keine aktiven Lese- oder Schreibaufträge.
	2	Es konnten erfolgreich Aufträge zu diesem Gerät abgewickelt werden.
	3	Das Gerät kann nicht mit einem Ping erreicht werden bzw. der letzte Auftrag wurde mit einem Fehler abgeschlossen, der auf einen Kommunikationsabbruch hindeutet und es bestehen weitere aktive Aufträge zum Gerät.
&ping()	Verbindungszustand zum Partnergerät. VT_UI1, Read-Only.	
	0	Das Gerät kann nicht mit einem Ping erreicht werden.
	1	Das Gerät kann mit einem Ping erreicht werden.
&alarmagentmib2()	Verbindungszustand zum SNMP-Agent-Partnergerät (Item sysUpTime aus dem MIB2-Profil kann erreicht werden). VT_UI1, Read-Only.	
	0	"sysUpTime" kann nicht gelesen werden.
	1	"sysUpTime" kann gelesen werden.

Die folgenden Informationsparameter sind nur sichtbar, wenn bei der Projektierung "Kein SNMP" ausgewählt wurde, weil das betreffende Gerät kein SNMP sondern nur den Ping unterstützt.

&syscontact()	Der bei der Projektierung im Feld sysContact eingegebene Text. VT_BSTR, Read-Only.
&syslocation()	Der bei der Projektierung im Feld sysLocation eingegebene Text. VT_BSTR, Read-Only.
&sysname()	Der bei der Projektierung im Feld sysName eingegebene Text. VT_BSTR, Read-Only.

## Beispiel

*SNMP:[OSM]&ipaddress()*

Als Rückgabewert wird die projektierte IP Adresse des Teilnehmernamens mit dem Namen OSM zurückgeliefert.

## Informationen über den SNMP-OPC-Server

Der OPC-Server stellt Variablen zur Verfügung, mit denen Informationen über den SIMATIC NET SNMP-OPC-Server abgefragt werden können.

Sie können folgende Informationen abfragen:

- Version des SIMATIC NET SNMP-OPC-Servers
- Version zur Winsocket
- Information zum Trap-Empfang

## Syntax

```
SNMP:[SYSTEM]<Informationsparameter>()
```

## Erklärungen

### SNMP

SNMP-Protokoll für den Zugriff auf die Informationsvariablen zum lokalen System.

### SYSTEM

Kennung des lokalen Systems, dieser Name ist immer fest.

### <Informationsparameter>

Es sind folgende Informationsparameter definiert:

&version()	Versionskennung des SIMATIC NET SNMP-OPC-Servers. VT_BSTR, Read-Only.
&winsockversion()	Versionskennung von Winsocket. VT_BSTR, Read-Only.
&traplisten()	Indikator, ob sich der SIMATIC NET SNMP-OPC-Server für den Empfang von Traps anmelden konnte. VT_BOOL; Read-Only.  <i>FALSE</i> Der SNMP-OPC-Server konnte sich nicht zum Empfang anmelden.  <i>TRUE</i> Der SNMP-OPC-Server hat sich erfolgreich zum Empfang angemeldet.

## Beispiel

```
SNMP:[SYSTEM]&version()
```

Als Rückgabewert wird die Version des SIMATIC NET SNMP-OPC-Servers zurückgeliefert, zum Beispiel *SIMATIC NET Core Server SNMP V6.1.1000.2815 Copyright ©SIEMENS AG*

## 2.11.5 SNMP-spezifische Traps

### Verarbeitung der Traps durch den OPC-Server

Traps sind Ereignisse, die vom Gerät ohne Aufforderung an den OPC-SNMP-Server geschickt werden. Der OPC-Server verarbeitet diese Ereignisse auf die folgende Art und Weise:

- Der Trap wird als Simple Event an der Alarms & Events-Schnittstelle abgebildet.
- Für jeden projektierten Trap werden an der Data Access-Schnittstelle 2 Variablen gebildet: Eine Variable wird bei jedem Auftreten der betreffenden Trap inkrementiert, die andere Variable speichert eine Beschreibung der Trap.

### Syntax

Erste Variable für die Anzahl der aufgetretenen Ereignisse:

`SNMP:[<Teilnehmername>]<Trapname>`

Zweite Variable für die Beschreibung der Trap:

`SNMP:[<Teilnehmername>]<Trapname>_description`

### Erklärungen

#### **SNMP**

SNMP-Protokoll für den Zugriff auf die Prozessvariablen (MIB-Objekte) und Trap-Variablen.

#### **<Teilnehmername>**

Der Teilnehmername wird bei der Projektierung (Anlagenkonfiguration) festgelegt.

#### **<Trapname>**

Name des Traps.

### Beispiel

Die erste Variable liefert die Anzahl der Coldstart-Traps, die vom Partnergerät ausgelöst wurden:

`SNMP:[OSM]coldStart`

Die zweite Variable liefert eine Beschreibung der Trap:

`SNMP:[OSM]coldStart_description`

## **2.12**

## **PROFINET-IO-Kommunikation über Industrial Ethernet**

Das Systemmodell eines dem PROFINET-IO-Controller zugeordneten PROFINET-IO-Device ist modulorientiert aufgebaut. Ein PROFINET-IO-Device kann mehrere Module enthalten, jedes Modul kann mehrere Submodule enthalten. Ein Submodul enthält i.d.R. die physikalischen Klemmen oder Treiberbausteine (Kanäle) für die anzuschließende Anlagen-Hardware, z. B. Förderbänder oder Sensoren, deren Eingangsdaten oder Stellwerte werden mit den IO-Daten angesprochen. Das Lesen und Schreiben der IO-Daten ist der wichtigste Anwendungsfall für PROFINET IO.

Sowohl Devices als auch Module und Submodule können Datensätze zur Verfügung stellen und Alarne generieren. Die Anwendung von Datensätzen und Alarmen ist geräteabhängig.

### **2.12.1**

### **Perfomanter SIMATIC NET OPC-Server für das PROFINET-IO-Protokoll**

#### **Einleitung**

Diese Konfigurationsvariante für das PROFINET-IO-Protokoll erfüllt höhere Performance-Anforderungen. Es wird der unterlagerte PROFINET IO COM-Server als Inproc-Server in den Outproc-OPC-Server geladen. Die Protokollbearbeitung läuft im Prozess des OPC-Servers ab; weitere Laufzeiten für Prozesswechsel und Multiprotokollbetrieb fallen weg. Der Prozesswechsel zwischen OPC-Client und OPC-Server ist noch vorhanden.

## Konfiguration

Die Aktivierung dieser performanten Variante erfolgt implizit dadurch, dass im Konfigurationsprogramm "Kommunikations-Einstellungen" das Protokoll "PROFINET IO" als einziges Protokoll ausgewählt wird (bei Auswahl weiterer Protokolle oder der OPC-UA-Schnittstelle entfällt der beschriebene Performance-Vorteil):

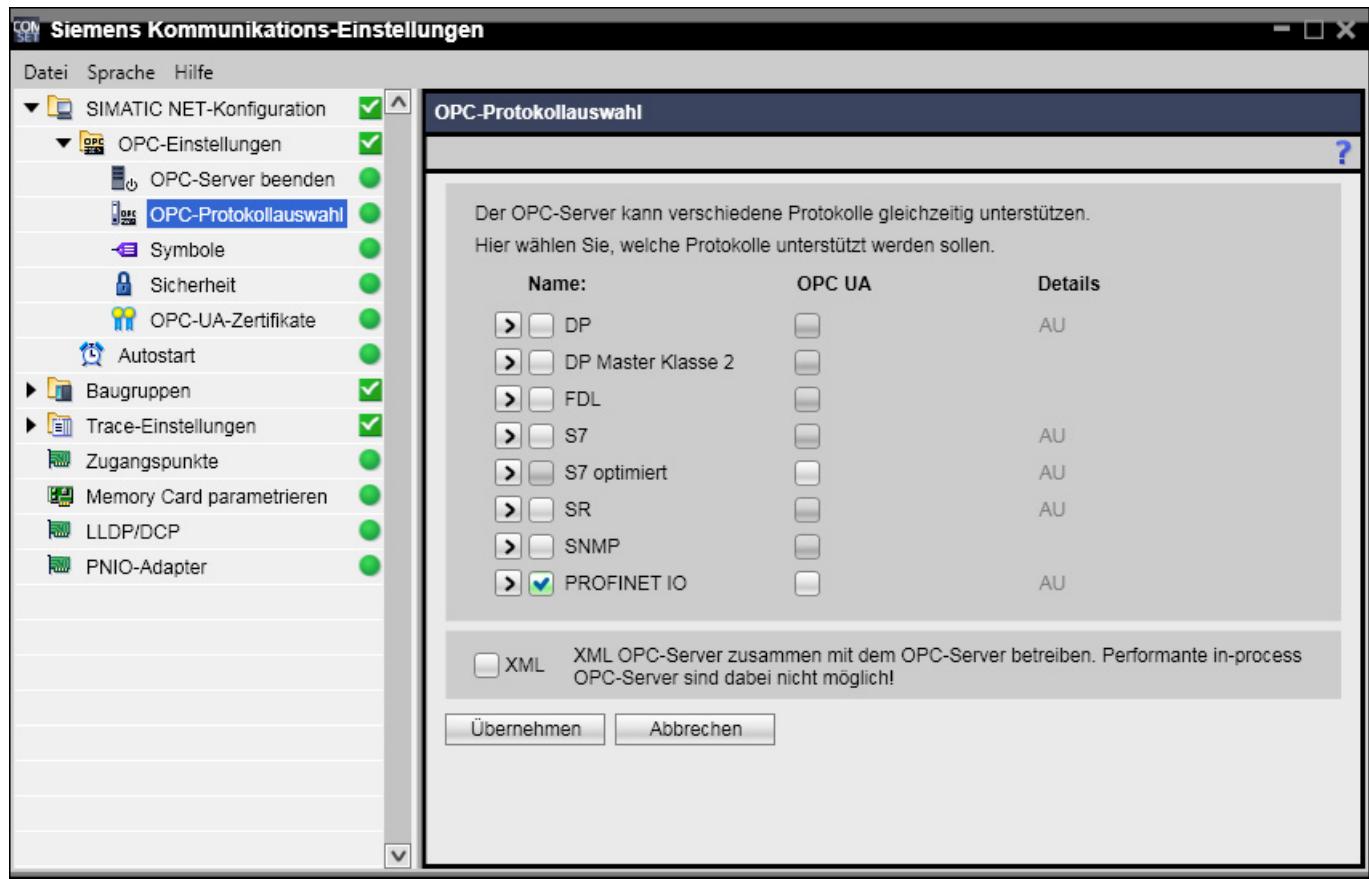


Bild 2-60 Fenster des Konfigurationsprogramms "Kommunikations-Einstellungen" zur Auswahl des PROFINET-IO-Protokolls

"Symbolik" darf zusätzlich ausgewählt werden.

### Hinweis

Für die Erstellung von Symbolen mit STEP 7 oder dem Symbol-Editor ist die Verwendung folgender Zeichen erlaubt: A-Z, a-z, 0-9, \_, -, ^, !, #, \$, %, &, ', /, (, ), <, >, =, ?, ~, +, \*, '!', :, |, @, [ ], { }, ". Zusätzlich sollten Sie bei der Erstellung von Symbolen mit STEP 7 darauf achten, dass es bei der Array-Auflösung zu Problemen kommen kann, wenn Ihre Symboldatei gleichzeitig Symbole der Form <Symbolname> und <Symbolname>[<Index>] enthält.

## Vor-/Nachteile

Die Verwendung des performanten SIMATIC NET OPC-Servers hat jedoch den Nachteil, dass nur der Einzelprotokollbetrieb von PROFINET IO möglich ist.

Dem stehen folgende Vorteile gegenüber:

- Höhere Performance als beim Multiprotokollbetrieb
- Einfache Konfiguration
- Zugang über die ProgID "OPC.SimaticNET"
- Mehrere Clients können den Server zur gleichen Zeit nutzen
- Die Stabilität des OPC-Servers ist nicht von den Clients abhängig.

---

### Hinweis

Die projektierte Zykluszeit des Servers ist nicht automatisch die Zeit, mit der Aufrufe an die unterlagerte PROFINET IO-Protokoll-Schnittstelle erfolgen. Je nach gewünschter Aktualisierungszeit eines OPC-Client kann die Aufrufrate auch langsamer sein.

---

### Hinweis

Wenn Sie im Programm "Kommunikations-Einstellungen" im Dialog "Sicherheit" die Schaltfläche "Sperren..." aktiviert haben, müssen Sie zur erneuten remoten Kommunikation über den PROFINET-IO-OPC-Server vorher die Schaltfläche "Freischalten" (remote Basis- und OPC-Kommunikation) klicken.

---

## 2.12.2 Wie können IO-Daten adressiert werden?

Im Allgemeinen werden für die IO-Daten keine Adressierungen unterstützt, die mehrere Submodule, Module oder Devices umfassen. Es sind keine internen Mengenaufrufe möglich, d.h. jede PROFINET-IO-Adresse muss einzeln gelesen oder geschrieben werden.

### Adressierung

Es wird eine sogenannte logische Adressierung verwendet.

Die logische Adresse verweist auf ein Abbild der E/A-Bereiche aller Devices in vom Controller verwalteten Prozessabbildern. Diese Prozessabbilder sind für den E- und den A-Bereich jeweils 4 Gbyte groß ( $2^{32}-1$ ). Die Zuordnung der Prozessabbilder erfolgt mit den PROFINET-IO-Projektierwerkzeugen. Die logische Adresse gibt den Byte-Offset des E/A-Bereichs eines Device im Prozessabbild an.

Die logische Adressierung ist für das Lesen und Schreiben von IO-Daten ideal. Bei geschickter Projektierung des Anwenders liegen die E/A Abbilder der Devices dicht gepackt. Es können zusammenhängende Bereiche mehrerer Devices auf einmal gelesen und geschrieben werden.

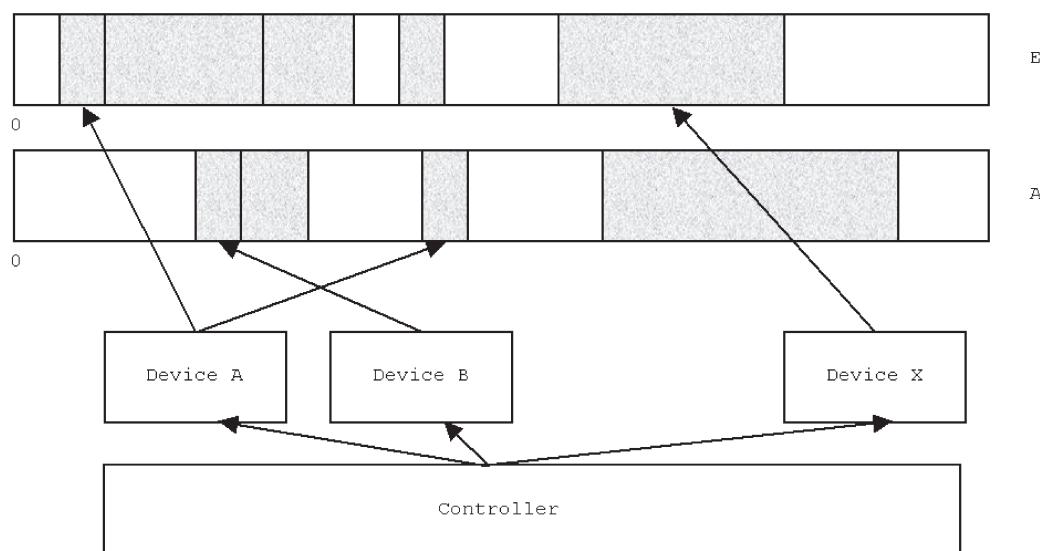


Bild 2-61 Lesen und Schreiben der IO-Daten von Devices in getrennt liegenden bzw. zusammenhängenden Bereichen durch den Controller

## Beispiel

Adressierungsbeispiel eines Controllers CP 1612-PROFINET-IO mit einer OPC-Server-Applikation und einem IM 151.

Die Adressen sind:

- Eingang 0, Länge 4 Bit
- Ausgang 0 und 1, Länge 2 Bit

Die Diagnoseadresse der IM151 ist 16382.

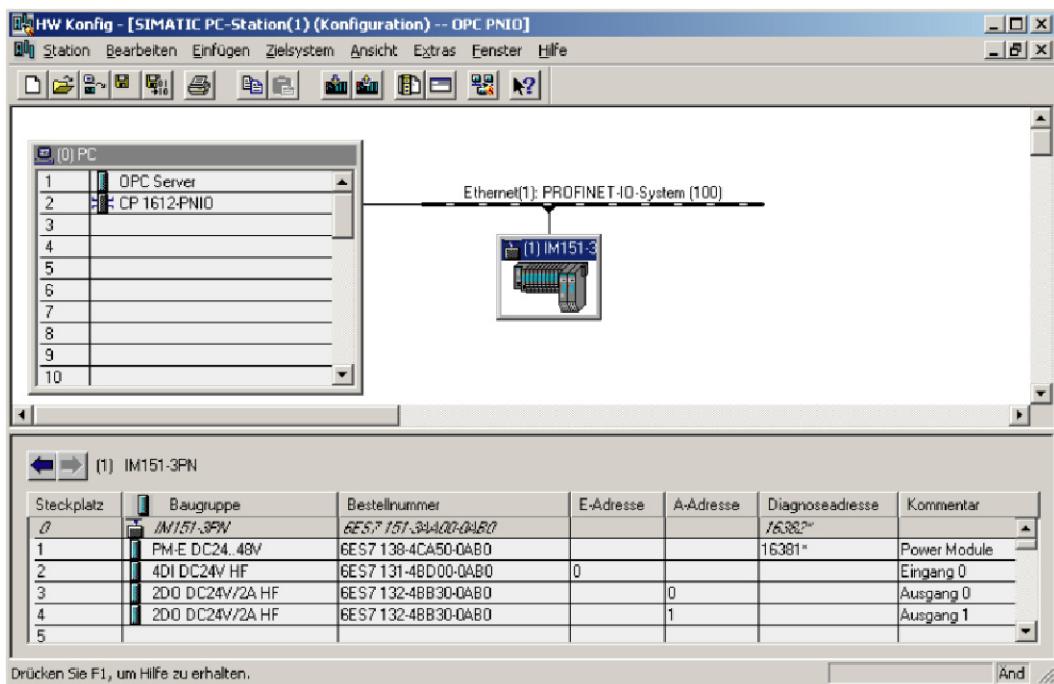


Bild 2-62 Anzeige der Adressen der Baugruppen einer PC-Station im Stationsfenster von "HW Konfig"

### Wie sollte die Adressierung sinnvoll erweitert werden?

Wenn nur ein einzelnes Eingangs-Bit eines Submoduls mit z. B. 64 Bit Breite gelesen werden soll, müssen alle 8 Bytes vollständig gelesen werden. Gleiches gilt für das Schreiben eines Ausgangs-Bit, es müssen alle 8 Bytes vollständig geschrieben werden.

Der Aufbau der Anlage, also Informationen über die tatsächlich vorhandenen Devices und deren Module und Submodule, wird projektiert und ist damit dem Anwender bekannt. Es müssen jedoch Module und Submodule auch noch im laufenden Betrieb entfernt oder hinzugefügt werden können.

Um dem OPC-Anwender trotzdem einen gewissen Komfort bei der Konvertierung der E/A Daten zu geben, wird die erweiterte Adressierung eingeführt. Der Anwender, der seine Anlage ja kennt, kann mit Hilfe der Item-Syntax den PROFINET-IO-OPC-Server lokal mit den notwendigen Projektierungsinformationen versorgen

#### Beispiel:

Angenommen, ein PROFINET-IO-Device steuert über ein IO-Submodul 64 elektrische Schalter. Das Submodul wird mit der Projektierungs-Software auf die logische Basisadresse 400 abgebildet mit einem Wertebereich von 64 Bit gleich 8 Byte. Diese Information wird vorgegeben. Mit dem folgenden Item kann der Anwender Bit 5 des Submoduls ab Offset 4 Lesen oder Schreiben.

*PNIO:[ctrl1]AB400,8,X4.5*

---

### Hinweis

Beachten Sie, dass bei einfacher Adressierung das Lesen und Schreiben von PROFINET IO-Daten exakt die projektierte Längenangabe erfordert (bei erweiterter Adressierung ist auch eine Teiladressierung möglich).

Beispiel:

Ein IO-Modul mit einer Länge von 4 Byte lässt sich über ein OPC-Item

PNIO:[CTRL1]EWORD0 (2 Byte) nicht teilweise lesen.

Erlaubt sind hier z. B. "PNIO:[CTRL1]EB0,4" oder "PNIO:[CTRL1]EDWORD0".

---

### 2.12.3 Wie kann der projektierte PROFINET-IO-Namensraum durchsucht werden?

Die projektierten PROFINET-IO-Geräte des OPC-Servers können mit den Suchfunktionen (Browsing) von OPC DA angezeigt werden.

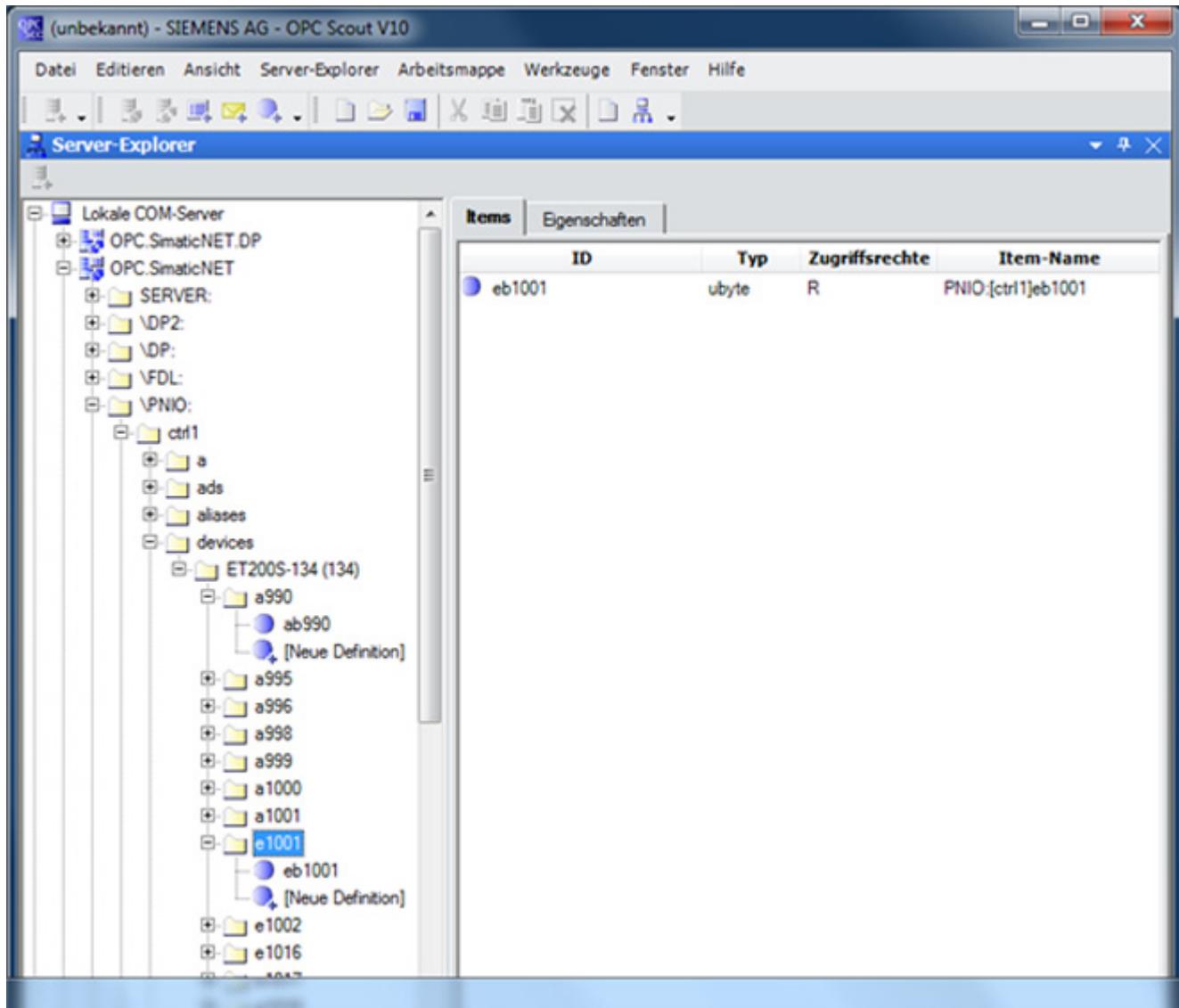


Bild 2-63 Durchsuchen des projektierten Namensraums im OPC Scout

Die vorhandenen Eingänge und Ausgänge werden mit Adressen angezeigt. Da auch die projektierte Datenlänge bekannt ist, wird jeweils ein vorgefertigtes Item angeboten.

In der technologischen Sicht werden unter dem Zweig "devices" alle projektierten Geräte mit Namen (darunter deren Module) aufgelistet. Zu jedem Modul werden alle vorhanden Ein- und Ausgänge mit Adressen angezeigt. Vorgefertigte Items sind bereits vorhanden.

## 2.12.4 Welche OPC-Variablen für PROFINET IO stehen zur Verfügung?

Der PROFINET-IO-OPC-Server von SIMATIC NET bietet folgende Variabtentypen an:

- Prozessvariablen - Daten Lesen/Schreiben/Beobachten
- Informationsvariablen - Zustandsabfragen
- Steuervariablen - Geräte aktivieren/deaktivieren

## 2.12.5 Protokoll-ID

Die Protokoll-ID für das PROFINET-IO-Protokoll lautet "PNIO".

## 2.12.6 Controller-Name

Der Verbindungsname spezifiziert den Kommunikationszugang durch Angabe des Kommunikationsprozessors, der mit dem PROFINET-Ethernet verbunden ist.

Der Controller-Name ist beim PROFINET-IO-Protokoll der Name der Kommunikationsbaugruppe mit Angabe des Steckplatzes (Index) in der PC-Station ("Komponenten Konfigurator").

Controller-Name = *CTRL<Index>*

Beispiele für Controller-Name:

- CTRL3
- CTRL5

## 2.12.7 PROFINET-IO-Prozessvariablen

Das Lesen und Schreiben von Items für PROFINET-IO-Prozessvariablen ist der häufigste Anwendungsfall. Ein einfacher Zugriff wird ermöglicht. Der Anwender muss den Controller-Namen und die Adresse aus der Projektierung vorgeben.

Der projektierte Adressraum kann im OPC-Namensraum durchsucht werden.

Eine Adressüberprüfung wird beim Hinzufügen von Items durchgeführt.

### Einfache Adressierung

#### Syntax

**Eingänge:**

```
PNIO:[<Controller-Namen>]E<Format><Adresse>
{.<Bit|Stringlänge>}{,<Anzahl>}
```

**Ausgänge:**

```
PNIO:[<Controller-Namen>]A<Format><Adresse>
{.<Bit|Stringlänge>}{,<Anzahl>}
```

## Erklärungen

### PNIO

PROFINET-IO-Protokoll für den Zugriff auf die Prozessvariable.

### <Gerätename> = CTRL<Index>

Protokollspezifischer Gerätename. Der Index des Gerätenamens wird bei der Konfiguration der PC-Station festgelegt. Wertebereich 1...32

Beispiel: CTRL1

### E

Kennzeichen für einen Eingang. Eingänge sind nur lesbar.

### A

Kennzeichen für einen Ausgang. Ausgänge sind les- und schreibbar.

### <Format>

Format der gelieferten Daten.

Durch die Formatangabe wird der Datentyp festgelegt.

Über die Automation- und Custom Schnittstelle von OPC können prinzipiell alle angegebenen OLE-Datentypen gelesen werden. Einige Entwicklungswerkzeuge, wie z.B. Visual Basic, bieten jedoch nur eine eingeschränkte Menge von Datentypen an.

Die folgende Tabelle listet deshalb den entsprechenden Visual Basic-Typ auf, in dem der Variablenwert dargestellt werden kann.

Datentypen, die größer als ein Byte sind, werden im Motorola-Format interpretiert.

Format-bezeichner	OLE-Datentyp	Visual Basic V6.0-Typ Automation Datatype	Beschreibung
X	VT_BOOL	Boolean	Zusätzlich ist noch die Bit-Adresse im jeweiligen Byte anzugeben. Das Schreiben einzelner Bits ist aus Konsistenzgründen nur bei der erweiterten Adressierung möglich, der OPC-Client sollte dabei jedoch immer alle Bits der logischen Adresse gleichzeitig beschreiben. Zusätzlich kann eine Array-Länge in Bits angegeben werden. Das Lesen von Bitarrays ist immer möglich, das Schreiben aus Konsistenzgründen aber nur, wenn die Bit-Adresse den Wert 0 hat und die Bit-Länge ein Vielfaches von 8 ist. Aus Gründen der Einheitlichkeit wird jedoch das Lesen von Bitarrays ebenfalls nur zugelassen, wenn die Bit-Adresse den Wert 0 hat und die Bit-Länge ein Vielfaches von 8 ist.
B oder BYTE	VT_UI1	Byte	Byte (unsigned)
W oder WORD	VT_UI2	Long	Wort (unsigned)
D oder DWORD	VT_UI4	Currency	Doppelwort (unsigned) Abbildung auf den umfassenderen Visual Basic- Typ 6.0 Currency 8 Byte, da kein entsprechender Datentyp vorhanden.
LWORD	VT_UI8	VT_UI8	Automation Typ Integer 8 Byte (unsigned), wird von Visual Basic 6.0 nicht unterstützt.

Formatbezeichner	OLE-Datentyp	Visual Basic V6.0-Typ Automation Datatype	Beschreibung
CHAR	VT_I1	Integer	Byte (signed) Abbildung auf den umfassenderen Visual Basic- Typ Integer 2 Byte.
INT	VT_I2	Integer (%)	Wort (signed)
DINT	VT_I4	Long (&)	Doppelwort (signed)
LINT	VT_I8	VT_I8	Automation Typ Integer 8 Byte (signed), wird von Visual Basic 6.0 nicht unterstützt.
REAL	VT_R4	Single	Fließkomma 4 Byte
LREAL	VT_R8	Double	Fließkomma 8 Byte
STRING	VT_BSTR	String	Zeichenkette Zusätzlich ist noch die für den String reservierte Länge anzugeben. Beim Schreiben können auch kürzere Strings geschrieben werden, wobei die übertragene Datenlänge immer die reservierte String-Länge in Bytes ist. Die nicht benötigten Bytes werden mit dem Wert 0x00 gefüllt.

**<Bit|Stringlänge>**

Spezifizierung für den Datentyp X: Bit-Nummer im adressierten Byte.

Der Bereich liegt zwischen 0 und 7.

Spezifizierung für den Datentyp STRING: Reservierte Stringlänge.

Für die Formatbezeichner X und STRING muss zusätzlich .<Bit|Stringlänge> angegeben werden.

**<Anzahl>**

Anzahl der Elemente

Das Lesen und Schreiben von Bitarrays ist nur möglich, wenn,

- die Bit-Adresse den Wert 0 hat und
- die Länge ein Vielfaches von 8 ist und
- die exakte Modullänge erreicht ist.

**<Adresse>=<logische Adresse>**

Einfache logische Adressierung

Logische Basisadresse des Device entsprechend der Projektierung.

Zahl Wertebereich 0...4 294 967 295.

---

**Hinweis**

Beachten Sie, dass das Schreiben und Lesen von PROFINET-IO-Daten exakt die projektierte Längenangabe erfordert. Z. B. ein IO-Modul mit einer Länge von 4 Byte lässt sich nicht mit nur 2 Byte über ein OPC-Item `PNIO:[CTRL1]AWORD0` teilweise beschreiben. Erlaubt sind dann z. B. `PNIO:[CTRL1]AB0,4` oder `PNIO:[CTRL1]ADWORD0`.

---

## Erweiterte Adressierung

In der erweiterten Adresse wird der Datenbereich des gesamten zusammengehörigen Submoduls angegeben. Diese Information ist der Projektierung zu entnehmen.

### Syntax

Eingänge:

```
PNIO:[<Controller-Name>]E<erweiterte Adresse>,
        <Format><Offset>{.<Bit|Stringlänge>}[,<Anzahl>]
```

Ausgänge:

```
PNIO:[<Controller-Name>]A<erweiterte Adresse>,
        <Format><Offset>{.<Bit|Stringlänge>}[,<Anzahl>]
```

### Zusätzliche Erklärungen

**<erweiterte Adresse> =**

B oder BYTE<logische Adresse>,<Bereichanzahl>

Die erweiterte Adresse beschreibt den gesamten Adressbereich eines Submoduls, innerhalb dessen beliebige Unterbereiche bis zu einzelnen Bits relativ adressiert werden können.

Die erweiterte Adresse wird durch die Byte-Anzahl ab der logischen Basisadresse beschrieben.

**<Offset>**

Byteoffset innerhalb des adressierten Submodul, an der das Element liegt, das angesprochen werden soll.

---

### Hinweis

Das Lesen und Schreiben einzelner Bits ist mit der erweiterten Adressierung möglich.

Weiterhin ist das Lesen und Schreiben von Feldern einzelner Bits ohne Einschränkung der Beginn-Bit-Adresse und beliebiger Feldlänge innerhalb der erweiterten Adresse möglich.

---

## Beispiele für Prozessvariablen für PROFINET IO

Hier finden Sie Beispiele, die die Syntax von Variablennamen für PROFINET-IO-Variablen verdeutlichen.

- **Eingänge**

*PNIO:[CTRL3]EB10*

Controller Index 3, logische Adresse 10, erstes Eingangsbyte.

*PNIO:[CTRL3]EB4,3*

Controller Index 3, logische Adresse 4, Feld der ersten 3 Bytes.

*PNIO:[CTRL 1]ED2*

Controller Index 1, logische Adresse 2, erstes Doppelwort.

*PNIO:[CTRL 1]EX10.0,64*

Controller Index 1, logische Adresse 10.0, Feld der ersten 64 Ausgangsbits. Nur Vielfache von 8 vom Beginn des ersten Byte.

*PNIO:[CTRL 1]EReal4*

Controller Index 1, logische Adresse 4, erste Fließkommazahl im Eingangsbereich.

**Erweiterte Adressierung:**

*PNIO:[CTRL 1]EB2,1,X1.0,2*

Controller Index 1, erweiterte Adresse 2, Bereichsanzahl 1 Byte, Offset 1 Byte, Feld der ersten 2 Bits ab Eingangsbit 0.

*PNIO:[CTRL 1]EB10,4,X1.3,7*

Controller Index 1, erweiterte Adresse 10, Bereichsanzahl 4 Bytes, Offset 1 Byte, Feld der ersten 7 Bits ab Eingangsbit 3.

- **Ausgänge**

*PNIO:[CTRL3]AB7*

Controller Index 3, logische Adresse 7, erstes Ausgangsbyte.

*PNIO:[CTRL 1]AW0,8*

Controller Index 1, logische Adresse 0, Feld der ersten 8 Worte.

*PNIO:[CTRL3]AX2.0,64*

Controller Index 3, logische Adresse 2.0, Feld der ersten 64 Ausgangsbits, lesbar und schreibbar. Nur Vielfache von 8 beginnend ab Bytes.

*PNIO:[CTRL 1]ASTRING100.32,3*

Controller Index 1, logische Adresse 100, Feld der ersten 3 Zeichenketten von 32 Zeichen.

**Erweiterte Adressierung:**

*PNIO:[CTRL 1]AB1,1,X0.0*

Controller Index 1, erweiterten Adresse 1, Bereichsanzahl 1 Byte, Offset 0 Byte erstes Ausgangsbit 0.

*PNIO:[CTRL 1]AB2,4,X1.0,2*

Controller Index 1, erweiterten Adresse 2, Bereichsanzahl 4 Bytes, Offset 1 Byte, Feld der ersten 2 Bits ab Ausgangsbit 0.

## 2.12.8 PROFINET-IO-Datenstatus

Mit den Items für den Status (IOPS und IOCS) kann ein OPC-Client die jeweiligen Status-Bytes einer logischen Adresse abfragen bzw. setzen.

Die folgende Grafik zeigt die Datenstatus und den Wertefluss im Prozessabbild.

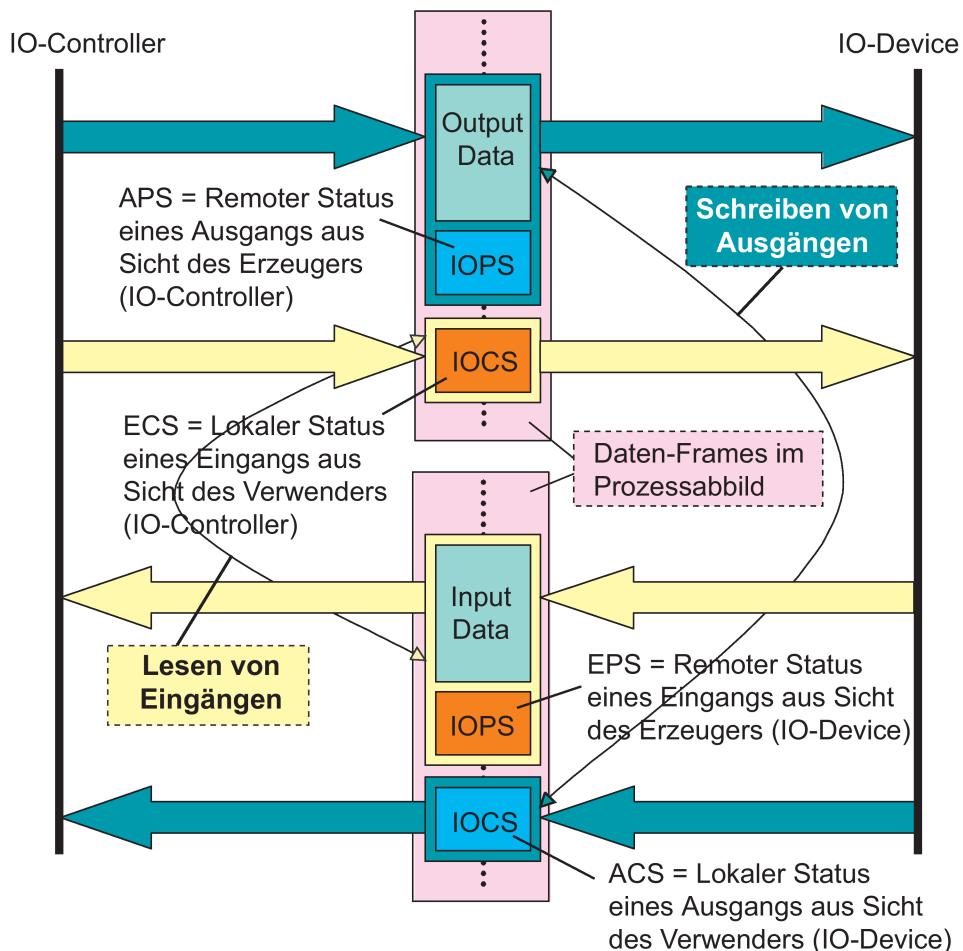


Bild 2-64 Übersicht der Datenstatus im Prozessabbild bei der PROFINET-IO-Kommunikation zwischen IO-Controller und einem IO-Device

Beim Schreiben von Ausgangsdaten in das Prozessabbild durch den OPC-Client gibt dieser den IO-Daten den remoten Status des Ausgangs aus seiner Sicht (Erzeuger) mit "APS". Dabei teilt das Device den eigenen, lokalen Status des Ausgangs aus seiner Sicht (Verwender) mit "ACS".

Beim Lesen von Eingangsdaten aus dem Prozessabbild durch den OPC-Client gibt dieser den eigenen, lokalen Status des Eingangs aus seiner Sicht (Verwender) mit "ECS". Dabei teilt das Device den remoten Status des Eingangs aus seiner Sicht (Erzeuger) mit "EPS".

---

### Hinweis

#### Lesender Zugriff auf zyklische Ausgangsdaten

Das Lesen, Beobachten oder Aktivieren von OPC-Items/nodes, die zyklische Ausgangsdaten abbilden, bewirkt eine einmalige bzw. ständige Aktualisierung der unterlagerten PROFINET IO-Protokoll-Schnittstelle mit den im OPC-Server-Cache enthaltenen Daten incl. Provider-Status (PNIO\_data\_write()). Bitte beachten Sie, dass nach Wiederkehr eines PROFINET IO-Devices oder Moduls ab dem Zeitpunkt der Aktualisierung diese Ausgangsdaten wieder zum Device übertragen werden.

---

## Syntax

```
PNIO: [<Controller-Name>] <E | A> <PS | CS> <Adresse>
```

## Erklärungen

### PNIO

PROFINET-IO-Protokoll für den Zugriff auf Datensätze.

### <Controller-Name> = CTRL<Index>

Protokollspezifischer Controller-Name. Der Index des Controller-Namens wird bei der Konfiguration der PC-Station festgelegt. Wertebereich 1 ... 32

### E

Kennzeichen für einen Eingang. Eingänge sind nur lesbar.

### A

Kennzeichen für einen Ausgang. Ausgänge sind les- und schreibbar.

### PS

Kennzeichen des remoten Status (Provider-Status).

### CS

Kennzeichen des lokalen Status (Consumer-Status).

oder

**EPS** Remoter Status Eingang

**APS** Remoter Status Ausgang

**ECS** Lokaler Status Eingang

**ACS** Lokaler Status Ausgang

**<Adresse>=<logische Adresse>**

Einfache logische Adressierung

Logische Basisadresse des Device entsprechend der Projektierung.

Zahl Wertebereich 0 ... 4 294 967 295.

Bezeichnung	Erläuterung
EPS	<p>Remoter Status einer Eingangs-Adresse Datentyp: VT_I4, ReadOnly. Der Datenerzeuger eines Eingangs ist aus Sicht des OPC-Servers das Device. Das Device teilt hierbei den Zustand der Eingangsdaten mit. Das Beobachten des EPS wird pollend durchgeführt. Die Verwendung des Items ist für eine OPC-Applikation nicht unbedingt erforderlich, da sich der Inhalt des Items in der OPC-Quality der mit derselben Adresse verbundenen Eingangs-Items wiederfindet.</p>
ECS	<p>Lokaler Status einer Eingangs-Adresse Datentyp: VT_I4, ReadWrite. Default: <i>GOOD</i> = 0 Der Datenverwender eines Eingangs ist aus Sicht des OPC-Servers der OPC-Client. Der OPC-Client teilt hierbei dem Device den Zustand der Eingangsdaten aus seiner Sicht mit. Üblicherweise ist dieser Zustand aber immer <i>GOOD</i>. Beim Lesen von IO Daten wird immer der zuletzt von der OPC-Applikation geschriebene lokale Status einer Eingangssadresse schreibend dem Partnergerät mitgegeben. Hat die OPC-Applikation das Item nicht angelegt oder noch nicht mit einem Wert beschrieben, so wird bei einem Lesen der zugehörigen Eingangsdaten der Wert <i>GOOD</i> übergeben. Das Lesen des Status durch den OPC-Client wird über den Cache abgewickelt.</p>
ACS	<p>Lokaler Status einer Ausgangs-Adresse Datentyp: VT_I4, ReadOnly. Der Datenverwender eines Ausgangs ist aus Sicht des OPC-Servers das Device. Das Device teilt hierbei den Zustand der Ausgangsdaten aus seiner Sicht mit. Das Lesen des Status erfolgt dadurch, dass der OPC-Client zuvor oder gleichzeitig ein mit derselben Adresse verbundenes Ausgangs-Item erfolgreich beschreiben konnte. Das Beobachten des ACS wird pollend durchgeführt, dabei werden der Funktion immer gleichzeitig die zuletzt vom OPC-Client geschriebenen Ausgangsdaten übergeben. Konnte der OPC-Client noch keinen Wert erfolgreich schreiben, so liefert das Lesen von Device einen Fehler bzw. die OPC-Quality <i>bad</i>.</p>
APS	<p>Remoter Status einer Ausgangs-Adresse Datentyp: VT_I4, ReadWrite. Default: <i>GOOD</i> = 0 Der Datenerzeuger eines Ausgangs ist aus Sicht des OPC-Servers der OPC-Client. Der OPC-Client teilt hierbei dem Device den Zustand der Ausgangsdaten aus seiner Sicht mit. Normalerweise ist dieser Zustand aber immer <i>GOOD</i>. Beim Schreiben von IO Daten werden immer der zuletzt von der OPC-Applikation geschriebene lokale Status der Ausgangsdaten und die Ausgangsdaten einer Ausgangssadresse schreibend mitgegeben. Hat die OPC-Applikation das IO-Item nicht angelegt oder noch keinen Ausgangswert erfolgreich geschrieben, so wird bei einem Schreiben der zugehörigen Ausgangsdaten der Wert <i>GOOD</i> übergeben. Beim Schreiben eines neuen Status durch den OPC-Client wird der Datenwert verwendet, der vom OPC-Client zuvor (oder gleichzeitig) mit der Adresse des verbundenen Ausgangs-Items erfolgreich beschrieben wurde. Konnte der OPC-Client noch keinen Wert erfolgreich schreiben, so liefert das Schreiben des Items den speziellen Rückgabewert (S_OPC_COREPNIOSTATUS). Der geschriebene Status wird dann beim nächsten erfolgreichen Schreiben von dem zugehörigen Ausgangs-Item geschrieben.</p>

### Aufbau des Statuswerts

Der Statuswert Typ *VT\_UI4* hat immer folgenden Aufbau:

- 0 = GOOD
- 1 = BAD

## 2.12.9 PROFINET-IO-Datensätze

Mit den Datensatzfunktionen können gerätespezifische Informationen ausgelesen oder Parameter geschrieben werden.

Die Bedeutung der Datensätze ist durch den Hersteller festgelegt. Sie können beispielsweise für Konfigurationsdaten eines Antriebs genutzt werden. Der OPC-Server kann beim Anmelden einer Datensatz-Variablen nur die Syntax auf Korrektheit überprüfen, aber nicht, ob auf Grund der Projektierung des Device die Variable im Partnergerät gültig und die Größe des Datensatzes ausreichend ist.

### Syntax

```
PNIO:[<Controller-Name>]<E|A>DS<Adresse>,DATA<DatensatzIndex>,
{<DatensatzLänge>}[,<Subelement>]
```

### Erklärungen

#### PNIO

PROFINET-IO-Protokoll für den Zugriff auf Datensätze.

#### <Controller-Name> = CTRL<Index>

Protokollspezifischer Controller-Name. Der Index des Controller-Namens wird bei der Konfiguration der PC-Station festgelegt.

Wertebereich: 1 ... 32

#### E

Kennzeichen für einen Eingang.

#### A

Kennzeichen für einen Ausgang.

#### DS

Kennzeichen für einen Datensatz.

#### <Adresse>=<logische Adresse>

Einfache logische Adressierung

Logische Basisadresse des Device entsprechend der Projektierung.

Zahl Wertebereich 0 ... 4 294 967 295.

## DATA

Kennzeichen für Daten.

- **<DatensatzIndex>**

Datensatzindex, gerätespezifischer Parameter.  
Wertebereich: 0 ... 65535).

Einige in der PROFINET IO Spezifikation standardisierte Datensatzindices sind:

0x8030 = 32816 Lesen Input Data Object

0x8031 = 32817 Lesen Output Data Object

0x8020 = 32800 Schreiben von Ersatzwerten für Output Data Object

60k+10 = 0xF00A = 61450 Lesen Diagnoseinformationen IO Device-Ebene

56k+10 = 0xE00A = 57354 Lesen Diagnoseinformationen AR Ebene

48k+10 = 0xC00A = 49162 Lesen Diagnoseinformationen Slot Ebene

32k+10 = 0x800A = 32788 Lesen Diagnoseinformationen Subslot Ebene

- **<DatensatzLänge>**

Datensatzlänge. Beim Lesen von Datensätzen kann dem Device die exakte Länge des zu lesenden Datensatzes in Byte als Vorgabe mitgeteilt werden. Der OPC-Client muss dann die Länge spezifizieren. Bei einer Längenvorgabe kann das Item auch geschrieben werden. Ohne Längenvorgabe kann das Item nicht geschrieben werden. Beim Lesen kann die Datensatzlänge optional weggelassen werden. Dann wird die Länge des Feldes der (Rückgabe-)Werte variabel zurückgeliefert. Der Anwender muss nicht wissen wie groß der Datensatz ist (Wertebereich: 0 ... 65535).

- **<Subelement>**

Um dem Anwender bei der Strukturierung der Datensätze zu unterstützen, kann er daraus einen oder mehrere Teilbereiche selektieren. Dazu verwendet er die ebenfalls für die IO-Prozessvariable gültigen Datentypen, die weiter oben beschrieben sind.

Lesen: Da die Struktur und Länge des vom Partner gelesenen Datensatzes nicht fest sein muss, ist es nicht verboten, Daten außerhalb des Bereichs zu definieren und anzufordern. Kann der entsprechende Bereich bei einem Empfang von Daten nicht mehr ausgefüllt werden, wechselt die Quality der Variablen entsprechend.

Schreiben: Beim Schreibzugriff auf einen Teilbereich wird immer der ganze Datensatz abgeschickt. Werden innerhalb eines OPC-Schreibauftrags mehrere Teilbereiche des Datensatzes geschrieben, wird er erst abgeschickt, nachdem alle seine Teilbereiche aktualisiert wurden.

Im Datensatz werden die Datentypen im Motorola-Format interpretiert und entsprechend in das Intel-Format konvertiert.

*<Subelement>=<Format><Offset>{.<Bit/Stringlänge>}{,<Anzahl>}*

- **<Offset>**

Byteoffset des Subelements relativ zum adressierten Datensatz.

Die Bedeutung der Werte *<Format>*, *<Bit/Stringlänge>* und *<Anzahl>* entspricht derjenigen im Kapitel "PROFINET-IO-Prozessvariablen (Seite 335)".

---

### Hinweis

Das Lesen und Schreiben einzelner Bits ist auch für Datensätze möglich.

Weiterhin ist das Lesen und Schreiben von Feldern einzelner Bits ohne Einschränkung der Beginnadresse und beliebiger Feldlänge innerhalb des Datensatzes möglich.

---

**Hinweis**

Durch die optionale Angabe von Typ, Adresse und Anzahl können Sie strukturiert auf Teilbereiche von Datensätzen zugreifen.

Variablen für Eingangs- oder Ausgangsdatensätze mit unterschiedlichem Controller-Namens oder unterschiedlicher Adresse oder unterschiedlichem Datensatzindex oder unterschiedlicher Länge verfügen über unabhängige Speicherbereiche.

Der Ausgangsdatenpuffer wird allokiert und mit Null initialisiert, wenn ein Item für einen unabhängigen Speicherbereich angelegt wird. Ein Schreibauftrag auf ein ADS-Item wird in einen internen Schreibpuffer geschrieben und übertragen.

Der OPC-Lese-Cache ist ein separater Puffer, welcher nur über Leseaufträge über Device aktualisiert wird.

Die Übertragung der Datensätze erfolgt azyklisch. Parallele Netzaufträge für die gleichen Daten sind möglich. Es wird immer der vollständige Sendedatensatz übertragen. Dies gilt auch bei Subelementzugriff oder wenn mehrere Clients gleichzeitig dieses Item beschreiben.

Ein paralleles Schreiben des gleichen Sendedatensatzes oder von Teilbereichen des Sendedatensatzes durch mehrere Clients kann zu Inkonsistenzen führen. Es wird deshalb empfohlen, immer den vollständigen Datensatz zu lesen oder zu schreiben.

**Wert des Datensatzes**

Datentyp: *VT\_ARRAY / VT\_UI1*. Lesbar; falls die Datensatzlänge angegeben wird auch schreibbar.

Enthält den zuletzt vom Device gelesenen Datensatz bzw. den zuletzt an das Device geschriebenen Datensatz. Struktur und Länge des Datensatzes sind nicht fest vorgegeben. Die Länge des gelesenen Datensatzes ist implizit in der Feldlänge des Werts enthalten.

Es ist zu beachten, dass, abhängig vom angesprochenen Device und dem Datensatzindex, geschriebene Datensätze möglicherweise nicht oder nur einmalig verändert zurückgelesen werden können. Dementsprechend sollte ein Pollen von Datensätzen (aktives Item) nur bei Datensätzen vorgenommen werden, die mehrmaliges Lesen unterstützen.

Wird ein Datensatz mit Leseeinschränkungen- oder Schreibeinschränkungen angesprochen, wird an der OPC-Schnittstelle ein Fehler (Kommunikationsfehler oder Zugriffsfehler) zurückgegeben.

**Beispiele für PROFINET-IO-Datensätze**

*PNIO:[CTRL 1]EDS16382,DATA61450,22*

Controller Index 1, Diagnoseadresse 16382, Datensatzindex 61450, Länge 22 Byte, Lesen der Diagnoseinformationen IO Device-Ebene.

*PNIO:[CTRL 1]EDS32,DATA32788,100,W8*

Controller Index 1, logische Adresse 32, Datensatzindex 32788, Länge 100 Byte. Lesen der Diagnoseinformationen des Subslots. Das Wort 8 enthält, falls vorhanden, den Fehlertyp des ersten Kanalfehlers.

*PNIO:[CTRL 1]JEDS10,DATA32768*

Controller Index 1, Modul Adresse 10, Datensatzindex 32768, Länge nicht vorgegeben, Item nur lesbar. Lesen der Diagnoseinformationen eines Slot.

*PNIO:[CTRL 1]JEDS16382,DATA61452*

Controller Index 1, Diagnose-(Basis-) Adresse 16832, Datensatzindex 61452, Länge nicht vorgegeben, Item nur lesbar. Lesen der Diagnoseinformationen herstellerspezifisch, Block, Slot, Kanal und Daten.

*PNIO:[CTRL 1]ADS10,DATA32768,22,B0,10*

Controller Index 1, logische Adresse 10, Datensatzindex 32768, Länge 22 Byte vorgegeben, Item ist schreib- und lesbar, Teilbereich der ersten 10 Bytes. Diagnoseinformationen eines Slot.

*PNIO:[CTRL 1]ADS10,DATA32768,22,X0.0,13*

Controller Index 1, logische Adresse 10, Datensatzindex 32768, Länge 22 Byte vorgegeben, Item ist schreib- und lesbar, Teilbereich der ersten 13 Bits ab Bit 0. Diagnoseinformationen eines Slot.

*PNIO:[Ctrl2]EDS11000,DATA2147483648,80,X0.0,1*

Dieses Item ist in eine OPC-Gruppe einfügbar, wenngleich auch keine gute Qualität erreichbar ist, da dieser Index nicht projektierbar ist und außerhalb des Wertebereichs liegt.

## 2.12.10 Syntax der systemspezifischen Informationsvariablen

### Syntax

PNIO: [SYSTEM] &version()

### Erklärung

Liefert eine Versionskennung für den PROFINET-IO-OPC-Server, hier die Zeichenfolge, z.B. SIMATIC NET Core Server PNIO V 7.xxxx.yyyy.zzzz Copyright 2012

Datentyp: VT\_BSTR

Zugriffsrecht: nur lesbar

## 2.12.11 PROFINET-IO-spezifische Informationsvariablen

### Syntax

PNIO: [<Controller-Name>] &<Informationsparameter>

### Erklärungen

#### PNIO

PROFINET-IO-Protokoll für den Zugriff auf die Prozessvariable.

**<Controller-Name>**

Protokollspezifischer Controller-Name. Der Index des Controller-Namens wird bei der Konfiguration der PC-Station festgelegt.

**&<Informationsparameter>**

Mögliche Werte sind:

- deviceactivate()
- deviceactivateval()
- devicedeactivate()
- devicedeactivateval()
- mode()
- modeval()

**&deviceactivate()**

Aktivieren eines Device.

Diese Betriebsart kann nur geschrieben werden. Das Einstellen der Betriebsart durch Schreiben eines der unten genannten Werte ist nur in Abhängigkeit von der PROFINET-IO-Anwendungsumgebung möglich.

Es müssen die erlaubten Betriebszustandsänderungen beachtet werden. Diese sind:

INACTIVE → ACTIVE

Eingabewerte als String:

Logische Adresse des Device entsprechend der Projektierung und das Kennzeichen für einen Eingang **E** oder Ausgang **A** : <A|E><logische Diagnoseadresse>.

Beispiel: E16373.

**&deviceactivateval()**

Aktivieren eines Device mit Zahlenwerten.

Datentyp VT\_ARRAY | VT\_I4, WriteOnly. Arraylänge 2 Elemente (EA-Flag und logische Adresse).

Eingabewerte als Feld von 2 Zahlenwerten:

Das erste Element des Feldes ist immer ein Flag, welches angibt, ob es sich bei der Adresse um eine Eingangs- oder Ausgangsadresse handelt.

0 = EingangAdresse

1 = AusgangAdresse

Das zweite Element beschreibt eine beliebige logische Adresse des Device.

Beispiel: Wert{0|16375} entspricht E16375.

**Hinweis**

Auf Grund des Aktivierens des Device kann protokollseitig der Alarm "Stationswiederkehr" ausgelöst werden.

**&devicedeactivate()**

Deaktivieren eines Device.

Diese Betriebsart kann nur geschrieben werden. Das Einstellen der Betriebsart durch Schreiben eines der unten genannten Werte ist nur in Abhängigkeit von der PROFINET-IO-Anwendungsumgebung möglich.

Es müssen die erlaubten Betriebszustandsänderungen beachtet werden.

Diese sind:

*ACTIVE → INACTIVE*

Eingabewerte als String:

Logische Adresse des Device entsprechend der Projektierung und das Kennzeichen für einen Eingang **E** oder Ausgang **A** : <**A|E**><logische Diagnoseadresse>.

Beispiel: *E16373*.

Es kann auch die die Diagnoseadresse eines Gerätes gewählt werden.

**&device deactivateval()**

Deaktivieren eines Devices mit Zahlenwerten.

Datentyp: VT\_ARRAY | VT\_I4, WriteOnly.

Feldlänge: 2 Elemente (EA-Flag und logische Adresse).

Das erste Element des Arrays ist immer ein Flag, welches angibt, ob es sich bei der Adresse um eine Eingangs- oder Ausgangsadresse handelt.

0 = EingangAdresse

1 = AusgangAdresse

Das zweite Element beschreibt eine beliebige logische Adresse des Device.

Beispiel: Wert{0|16373} entspricht E16373.

### Hinweis

Auf Grund des Deaktivierens des Device kann protokollseitig der Alarm "Stationsausfall" ausgelöst werden.

**&mode()**

Betriebszustand des Controllers ermitteln oder setzen.

Beim Schreiben eines neuen Betriebszustands sind alle Betriebszustandsänderungen erlaubt:

OFFLINE ↔ CLEAR ↔ OPERATE

oder

OFFLINE ↔ OPERATE

Damit ist beispielsweise auch ein direkter Wechsel von OFFLINE nach OPERATE und umgekehrt möglich.

Der Zustand OFFLINE darf gesetzt werden, z. B. um den Controller zu stoppen.

Ausschließlich im Anlauf, wenn sich der erste Client mit dem OPC-Server verbindet, wird nicht automatisch in den Zustand CLEAR, sondern in den Zustand OPERATE gewechselt, falls mit NCM "PNIO Controller automatisch auf OPERATE setzen" projektiert wurde.

Beim Beenden des OPC-Servers wird selbstständig der Zustand OFFLINE gesetzt.

Datentyp VT\_BSTR, Schreibbar, Lesbar.

- OFFLINE
- CLEAR
- OPERATE

**&modeval()**

Betriebszustand des Controllers ermitteln oder setzen als Zahl.

Datentyp: VT\_UI4, Schreibbar, Lesbar

Wertebereich	Bedeutung
0	OFFLINE
1	CLEAR
2	OPERATE

## 2.13 PROFINET-IO-Kommunikation mit OPC UA über Industrial Ethernet

Das Systemmodell eines dem PROFINET-IO-Controller zugeordneten PROFINET-IO-Device ist modulorientiert aufgebaut. Ein PROFINET-IO-Device kann mehrere Module enthalten, jedes Modul kann mehrere Submodule enthalten. Ein Submodul enthält i.d.R. die physikalischen Klemmen oder Treiberbausteine (Kanäle) für die anzuschließende Anlagen-Hardware, z.B. Förderbänder oder Sensoren. Deren Eingangsdaten oder Stellwerte werden mit den IO-Daten angesprochen. Das Lesen und Schreiben der IO-Daten ist der wichtigste Anwendungsfall für PROFINET IO.

Sowohl Devices als auch Module und Submodule können Datensätze zur Verfügung stellen und Alarne generieren. Die Anwendung von Datensätzen und Alarmen ist geräteabhängig.

Der PROFINET-IO-OPC-UA-Server von SIMATIC NET hat folgende Eigenschaften:

- Prozessvariablen - Daten Lesen/Schreiben/Beobachten
- Informationsvariablen - Zustandsabfragen
- Steuervariablen - Geräte aktivieren/deaktivieren

### 2.13.1 SIMATIC NET OPC-UA-Server für das PROFINET-IO-Protokoll

#### Einleitung

Der folgende Abschnitt beschreibt eine Konfigurationsvariante für das PROFINET-IO-Protokoll, die auch OPC UA unterstützt. Hierfür werden die PROFINET-IO-COM-OPC-Data-Access-Server als Outproc-OPC-Server eingerichtet.

## Konfiguration

Die Aktivierung des PROFINET-IO-UA-Servers erfolgt durch die Auswahl von "PROFINET IO" und "OPC UA" im Konfigurationsprogramm ""Kommunikations-Einstellungen" im Katalog "OPC-Protokollauswahl":

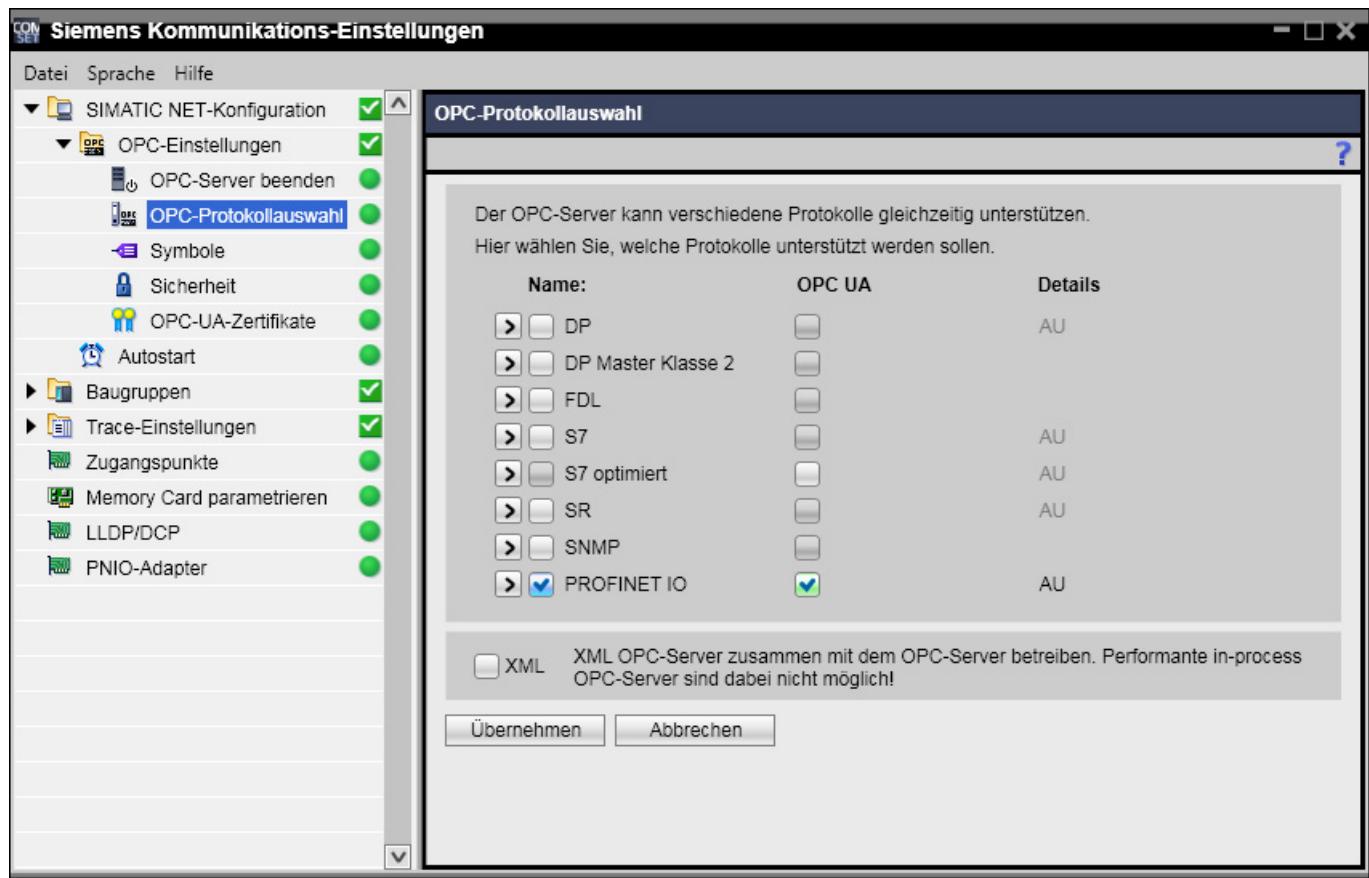


Bild 2-65 Fenster des Konfigurationsprogramms "Kommunikations-Einstellungen" zur Auswahl von OPC UA für das PROFINET-IO-Protokoll

## Vorteile / Nachteile

Bei Verwendung des PROFINET-IO-OPC-UA-Servers ist nur der Outproc-Betrieb von PROFINET IO möglich. Der PROFINET-IO-OPC-UA-Server-Prozess muss zum Erhalt der Empfangsbereitsschaft in Betrieb sein. Ein Beenden des PROFINET-IO-OPC-UA-Servers - auch nach Abmeldung aller OPC-UA-Clients - wird nicht ausgeführt und darf nicht ausgeführt werden.

Dem stehen folgende Vorteile gegenüber:

- Es ist keine COM/DCOM-Konfiguration mehr nötig.
- Performante, sichere Kommunikation
- Für Ereignisse und Datenzugriffe ist nur noch ein Server erforderlich.
- Mehrere Clients können den Server zur gleichen Zeit nutzen.

Wenn Sie die Optionskästchen "OPC UA" und "PROFINET IO" aktiviert haben, wird der PROFINET-IO-UA-Server immer gestartet. Das hat bei einer entsprechenden PROFINET-IO-Projektierung zur Folge, dass der PROFINET-IO-Controller im OFFLINE-Betrieb ist. Erst wenn mindestens ein OPC-UA-Client sich auf den Server verbunden hat, wechselt der PROFINET-IO-Controller in den OPERATE-Betrieb.

---

#### **Hinweis**

Die projektierte Zykluszeit des Servers ist nicht automatisch die Zeit, mit der Aufrufe an die unterlagerte PROFINET IO-Protokoll-Schnittstelle erfolgen. Je nach gewünschter Aktualisierungszeit eines OPC-Client kann die Aufrufrate auch langsamer sein.

---

---

#### **Hinweis**

Wenn OPC UA für das PROFINET-IO-Protokoll aktiv ist, geht ein projektierter PROFINET-IO-Controller für den OPC-Server in den OPERATE-Betrieb, sobald der erste OPC-UA-Client am OPC-UA-PROFINET-IO-Server angemeldet ist. Wenn sich kein OPC-UA-Client verbunden hat, z.B. nach Hochfahren des Rechners, bleibt ein projektierter PROFINET-IO-Controller für den OPC-Server im OFFLINE-Betrieb.

---

## 2.13.2 Wie wird der PROFINET-IO-OPC-UA-Server adressiert?

### Server-URL

Für das native binäre TCP-Protokoll gibt es für den OPC-Client zwei Möglichkeiten der Server-Adressierung:

- Direkte Adressierung:
  - opc.tcp://<hostname>:55104
    - oder
  - opc.tcp://<IP-Adresse>:55104
    - oder
  - opc.tcp://localhost:55104

Der PNIO-OPC-UA-Server hat den Port 55104.

- Die URL des PROFINET-IO-OPC-UA-Servers kann auch über den OPC-UA-Discovery-Dienst gefunden werden.

Die Eingabe zum Auffinden des Discovery-Servers lautet:

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

Der Discovery-Server hat den Port 4840 (für TCP-Verbindungen) und den Port 52601 (für HTTP-Verbindungen).

### IPv6-Adresse

Für die IP-Adresse kann auch eine IPv6-Adresse verwendet werden. Die Adresse muss in Klammern angegeben werden, z.B. [fe80:e499:b710:5975:73d8:14].

### Endpunkte und Sicherheitsmodi

Der SIMATIC NET PROFINET-IO-OPC-UA-Server unterstützt Endpunkte mit dem nativen binären TCP-Protokoll und ermöglicht Authentisierung über Zertifikate und eine verschlüsselte Übertragung.

Der Discovery-Dienst auf dem angesprochenen Host meldet die Endpunkte der Server, sowie deren Sicherheitsanforderungen und Protokollunterstützung.

Die Server-URL "opc.tcp://<hostname>:55104" des PROFINET-IO-OPC-UA-Servers bietet folgende Endpunkte:

- Endpunkt im Sicherheitsmodus "SignAndEncrypt":

Zur Kommunikation mit dem Server werden Signierung und Verschlüsselung gefordert. Die Kommunikation ist durch Zertifikataustausch und Passworteingabe geschützt.

Zusätzlich zum Sicherheitsmodus wird die Sicherheitsrichtlinie Basic128Rsa15 angezeigt.

- Endpunkt im Sicherheitsmodus "keiner":

In diesem Modus werden keine Sicherheitsfunktionen vom Server gefordert (Sicherheitsrichtlinie "None").

Weitere Details zu den Sicherheitsfunktionen finden Sie im Kapitel "OPC-UA-Schnittstelle programmieren (Seite 508)".

Die Sicherheitsrichtlinien "Basic128Rsa18" und "None" finden Sie in der UA-Spezifikation der OPC Foundation unter folgender Internet-Adresse:

"<http://opcfoundation.org/UA>" > "Specifications" > "Part 7"

Weitere Informationen finden Sie auf folgender Internetseite:

OPC Foundation (<http://www.opcfoundation.org/profilereporting/index.htm>) > "Security Category" > "Facets" > "Security Policy"

## Die OPC-UA-Discovery des OPC Scout V10

Der OPC Scout V10 ermöglicht das Öffnen des OPC-UA-Discovery-Dialogs zur Übernahme von UA-Endpunkten in den Navigationsbereich des OPC Scout V10.

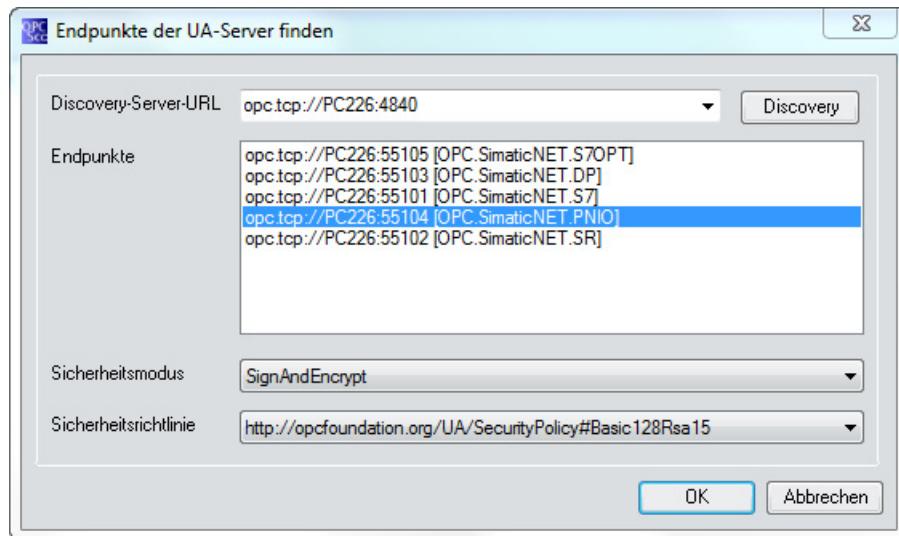


Bild 2-66 Das Dialogfeld "Endpunkte der UA-Server finden" des OPC Scout V10

Der PROFINET-IO-OPC-UA-Server kann über den OPC-UA-Discovery-Dienst gefunden werden. Zur Eingabe siehe oben unter "Server-URL".

Der OPC Scout V10 beinhaltet eine Liste der OPC-UA-Endpunkte. Diese sind persistent. Der Discovery-Dienst auf dem angesprochenen Host meldet dann die registrierten OPC-UA-Server sowie deren Ports und Sicherheitsmodi.

Weitere Details finden Sie in der Online-Hilfe des OPC Scout V10.

### 2.13.3 Welche Namensräume bietet die PROFINET-IO-Kommunikation mit OPC UA?

Der PROFINET-IO-OPC-UA-Server bietet folgende Namensräume an:

Tabelle 2- 8 Namensräume von OPC UA

Namensraum-Index	"Bezeichner" (Namensraum-URI) / Kommentar
0	"http://opcfoundation.org/UA/" von der OPC Foundation spezifiziert
1	"urn:Siemens.Automation.SimaticNET.PNIO:(GUID)" Eindeutiger Bezeichner des lokalen performanten PROFINET-IO-OPC-UA-Servers.
2	"PNIOTYPES:" Bezeichner für den Namensraum der PROFINET-IO-Typen.
3	"PNIO:" Bezeichner des lokalen performanten PROFINET-IO-OPC-UA-Servers mit neuer vereinfachter Syntax (durchsuchbar und verwendbar mit UA)
4	"PNIOCOM:" Bezeichner des Servers mit alter Syntax. (mit UA verwendbar aber nicht durchsuchbar) In diesem Namensraum kann die PROFINET-IO-OPC-DA-kompatible Syntax im OPC-UA-Server angewendet werden, jedoch sind hier keine Bezeichner möglich, die im OPC-DA-Server Alias-Items sowie Symbolik entsprechen. Beispiel: Im Namensraum PNIOCOM: gibt es den Bezeichner PNIO:[cltr1]ab0 ebenso wie als Item im OPC-DA-Server.

Abhängig von der Konfiguration kann sich in Ihrem OPC-UA-Client bei den Indizes 2 bis 4 die Zuordnung zwischen Namensraum-Index und Namensraum-Bezeichner ändern. Der Bezeichner muss daher immer eindeutig sein.

Die Namensraum-Indizes 0 und 1 sind reserviert und in Ihrer Bedeutung von der OPC Foundation spezifiziert.

Die Zuordnung der restlichen Namensraum-Indizes zu den Bezeichnern (Namensraum-URI) muss zu Beginn einer OPC-UA-Session vom Client unter Angabe des Bezeichners über die Datenvariable "NamespaceArray" ermittelt werden. Die Bezeichner PNIOTYPES: PNIO: PNIOCOM: sind beim PROFINET-IO-OPC-UA-Server immer vorhanden.

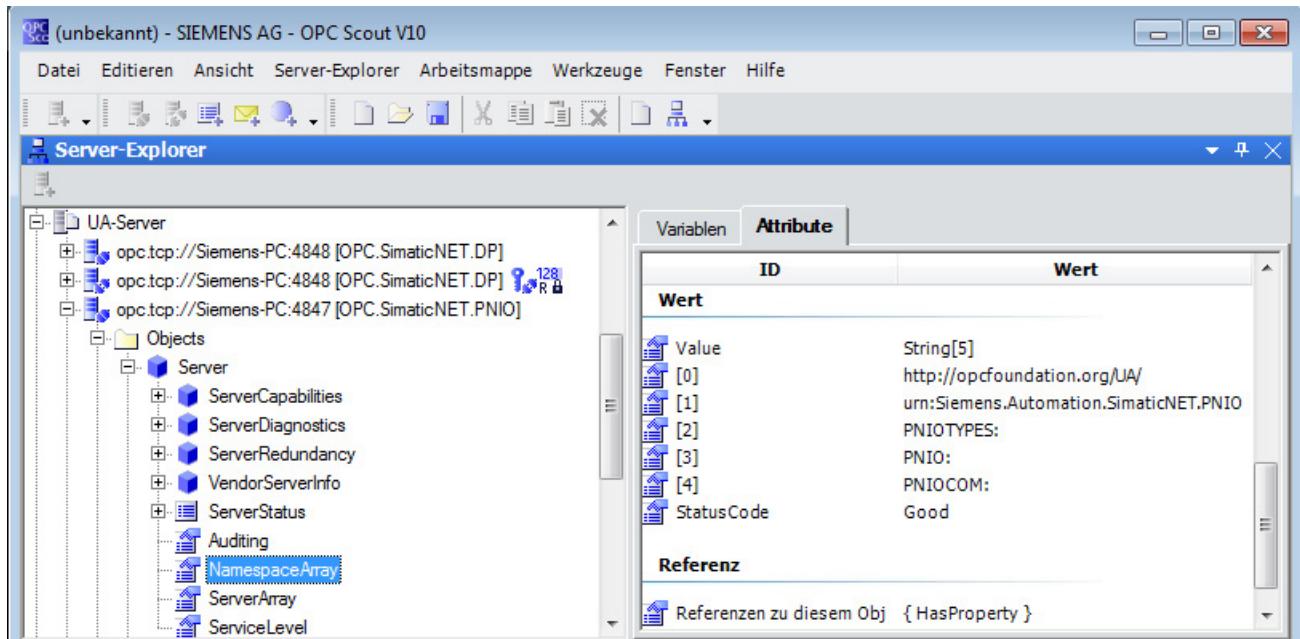


Bild 2-67 Anzeige der PROFINET-IO-Namensräume mittels der Browse-Funktion des OPC Scout V10

## Wie kann der projektierte PROFINET-IO-Namensraum durchsucht werden?

Die projektierten PROFINET-IO-Geräte des OPC-Servers können mit den Suchfunktionen (Browsing) von OPC Data Access angezeigt werden.

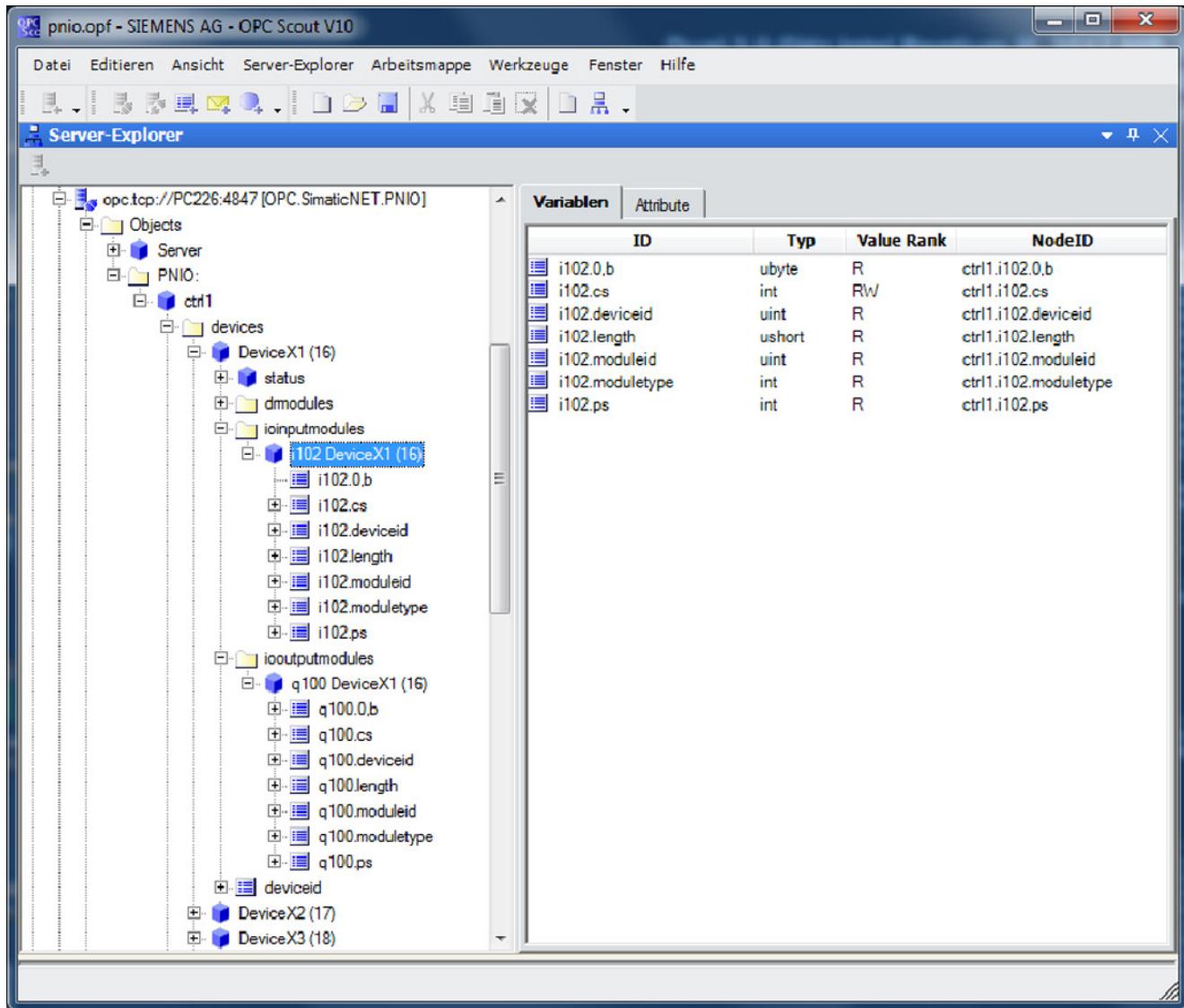


Bild 2-68 Durchsuchen des projektierten Namensraums im OPC-Scout

Die vorhandenen Eingänge und Ausgänge werden mit Adressen angezeigt. Da auch die projektierte Datenlänge bekannt ist, wird jeweils eine vorgefertigte Variable angeboten.

In der technologischen Sicht werden unter dem Zweig "devices" alle projektierten Geräte mit Namen (darunter deren Module) aufgelistet. Zu jedem Modul werden alle vorhandenen Ein- und Ausgänge mit Adressen angezeigt. Vorgefertigte Items sind bereits vorhanden.

## 2.13.4 Die Nodeld

### Identifikation einer PROFINET-IO-Prozessvariable

Die Nodeld identifiziert mit Hilfe des folgenden Tupels eine PROFINET-IO-Prozessvariable eindeutig:

- Namensraum-Index
- Bezeichner (Zeichenfolge, numerischer Wert)

### Beispiele

- Nodeld:
  - Namensraum-URI:  
*PNIO:*  
(= Namensraum-Index 3) für Siemens.Automation.SimaticNET.PNIO
  - Bezeichner  
*ctrl1.q0.0,b*
- Nodeld:
  - Namensraum-URI:  
*PNIocom:*  
(= Namensraum-Index 4)
  - Bezeichner  
*PNIO:[ctrl1]AB0*

### Wie verhält sich der neue auf OPC UA angepasste Namensraum?

Während die Welt der OPC-Data-Access-Items eher auf das Lesen und Schreiben von Prozessvariablen beschränkt, in sich abgeschlossen ist und daneben davon völlig losgelöst beispielsweise die Alarmwelt existiert, ist die OPC-UA-Sicht auf Automatisierungsobjekte mit verschiedenen Eigenschaften bezogen.

Dementsprechend greift OPC UA nicht mehr alleine auf Items zu, sondern auf Objekte und deren Unterobjekte. Datenvariablen und Methoden sind beispielsweise Unterobjekte eines PROFINET-IO-Device-Objekts. Properties definieren die Objekte näher, alle Informationen werden von Attributen der Objekte bereitgestellt.

Ein OPC-Data-Access-Item für Zugriff auf IO-Bereiche eines PROFINET-IO-Device entspricht dabei einer OPC-UA-Datenvariablen. Ein OPC-Data-Access-Item zur Aktivierung/Deaktivierung eines Device entspricht einer OPC-UA-Methode.

Den qualifizierten Bezeichnern (Nodeld) unter OPC UA kommt eine größere Bedeutung als unter OPC Data Access zu. Jeder einzelne Zugriff auf Objekt, Unterobjekt, Property und Attribut erfolgt über dessen Nodeld.

Der OPC-Namensraum für PROFINET IO ist in einer Baumstruktur dargestellt und kann nach Nodelds durchsucht werden. Alle Texte des Namensraums sind in Englisch.

## Syntax

Die Nodelds aller OPC-UA-Objekte haben folgenden Aufbau:

<controllerobjekt>"."<objekt>"."<unterobjekt>"."<property>

Ein Objekt kann weitere Unterobjekte beinhalten.

Eine nicht interpretierbare Nodeld wird mit einem Fehler zurückgewiesen. Die Groß- oder Kleinschreibung der Buchstaben "A-Z" wird bei allen Objekten ignoriert.

## Symbolische Objektdarstellung

Die OPC-UA-Spezifikation empfiehlt zur hierarchischen Beschreibung des Adressraums eine einheitliche Symboldarstellung. Die in diesem Dokument verwendeten Symbole sind:

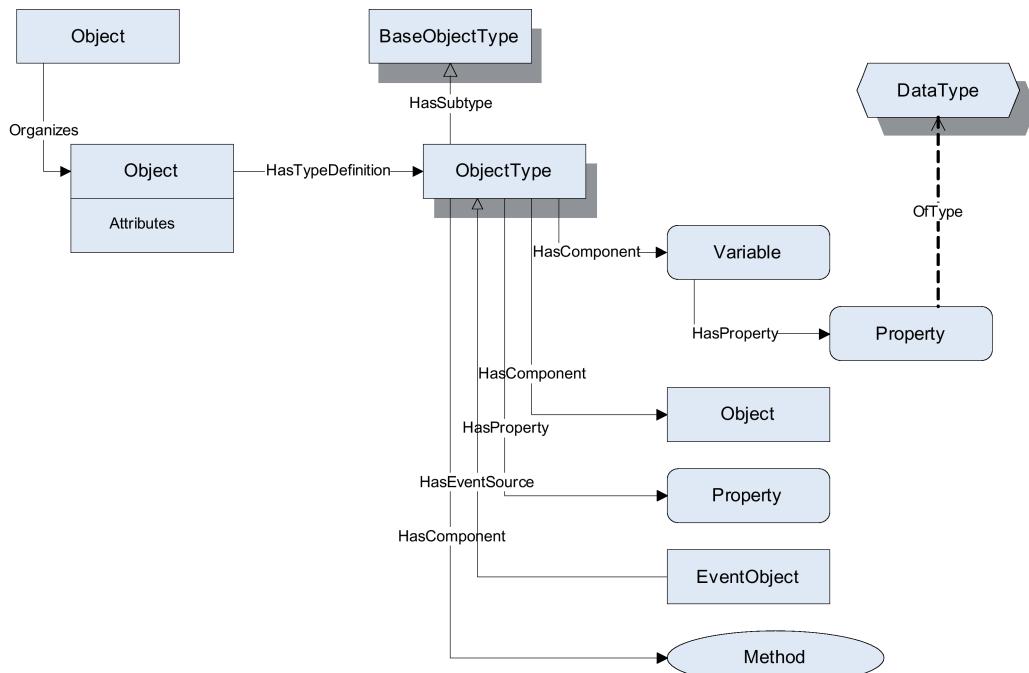


Bild 2-69     Symbole des OPC-UA-Adressraums

## 2.13.5 Board-Objekt und PROFINET-IO-Controller

### Der Board-Name eines PROFINET-IO-Controllers

Alle protokollspezifischen Objekte sind immer einem Board zugeordnet, bei PROFINET IO ist dies der PROFINET-IO-Controller. Der Board-Name ist der in STEP 7 oder dem TIA Portal projektierte PNIO-Controller zur Identifikation der Baugruppe.

## Syntax

`ctrl<Steckplatz>`

## Erklärungen

**ctrl**

Kennzeichen für den PNIO-Controller

**<Steckplatz>**

Steckplatz im "Komponenten Konfigurator"

**Beispiel:**

`ctrl12`

bezeichnet den PNIO-Controller im Steckplatz 12 des "Komponenten Konfigurators"

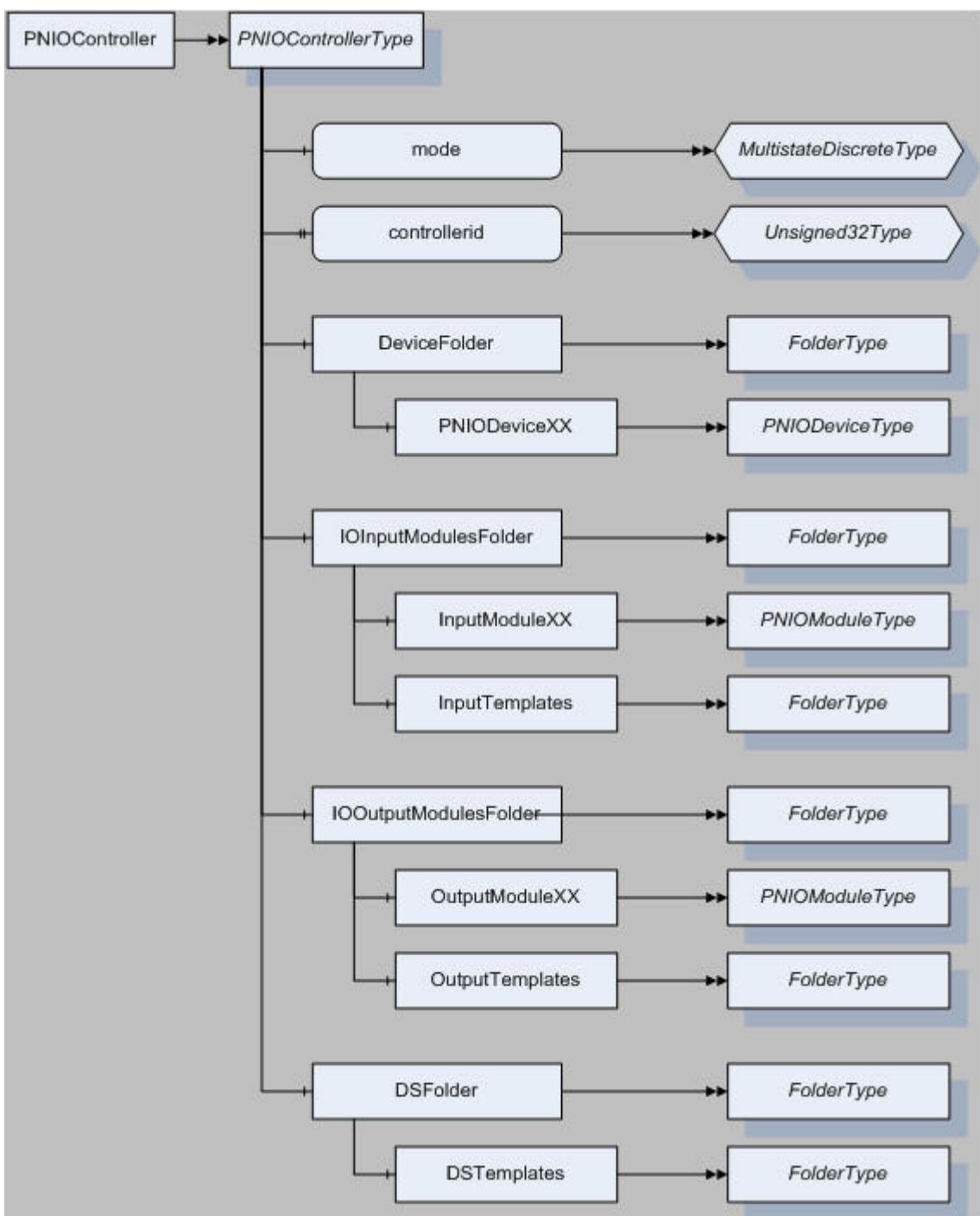


Bild 2-70 PROFINET-IO-Controller Klassenbaum

## 2.13.6 Datenvariablen des PROFINET-IO-Controllers

### Syntax für die Datenvariablen des PROFINET-IO-Controllers

<Controller>.<Datenvariable>

#### Erklärungen

##### <Controller>

Protokollspezifischer Verbindungsname. Der Verbindungsname wird bei der Projektierung festgelegt. Beim PNIO-Protokoll ist der Verbindungsname der projektierte Name der Kommunikationsbaugruppe.

##### <Datenvariable>

Datenvariable	Beschreibung							
mode	<p>Betriebszustand des Controllers ermitteln oder setzen. Datenvariable vom UA-Typ "MultistateDiscreteType", Lesen und Schreiben.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <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>Beim Schreiben eines neuen Betriebszustandes müssen die erlaubten Betriebszustandsänderungen beachtet werden. Diese sind: OFFLINE ↔ CLEAR ↔ OPERATE Im Fehlerfall wird ein Kommunikationsfehler zurückgegeben. Der Zustand CLEAR wird in der Regel nur von DP-Slaves umgesetzt, die über einen IE-PB Link (Verbinder zwischen Ethernet und PROFIBUS) als PROFINET-IO-Devices emuliert werden. Abhängig von Konfigurationswert AutoOnline wird der Controller vom OPC-Server beim ersten Anmelden eines Client selbstständig in den Modus OPERATE gefahren. Beim Abmelden des letzten Client wird der Controller unabhängig vom Konfigurationswert AutoOnline automatisch in den Modus OFFLINE gefahren. Beim unplanmäßigen Beenden eines Client (Absturz) wird dies vom UA-Protokollstack entsprechend der zwischen Client und Server ausgehandelten KeepAlive-Mechanismen erkannt, nach der entsprechenden Wartezeit wird der Client automatisch vom Protokollstack abgemeldet. Beim Herunterfahren des OPC-Servers wird der Controller beendet.</p>		0	OFFLINE	1	CLEAR	2	OPERATE
0	OFFLINE							
1	CLEAR							
2	OPERATE							
controllerid	<p>Steckplatz im "Komponenten Konfigurator" Property vom Typ "UInt32", nur lesbar</p>							

#### Beispiele:

ctrl4.mode  
ctrl4.controllerid

## 2.13.7 Device-Objekte

Kommunikationspartner des PROFINET-IO-Controllers sind die PROFINET-IO-Devices, welche die Ein- und Ausgangsmodule besitzen.

## Syntax

ctrl<Steckplatz>.dev<DeviceNummer>

## Erklärungen

**ctrl**

Kennzeichen für den PNIO-Controller

**<Steckplatz>**

Steckplatz im "Komponenten Konfigurator"

**dev**

Kennzeichen für ein PNIO-Device

**<DeviceNummer>**

Device-Nummer entsprechend der STEP 7- bzw. TIA-Portal-Projektierung

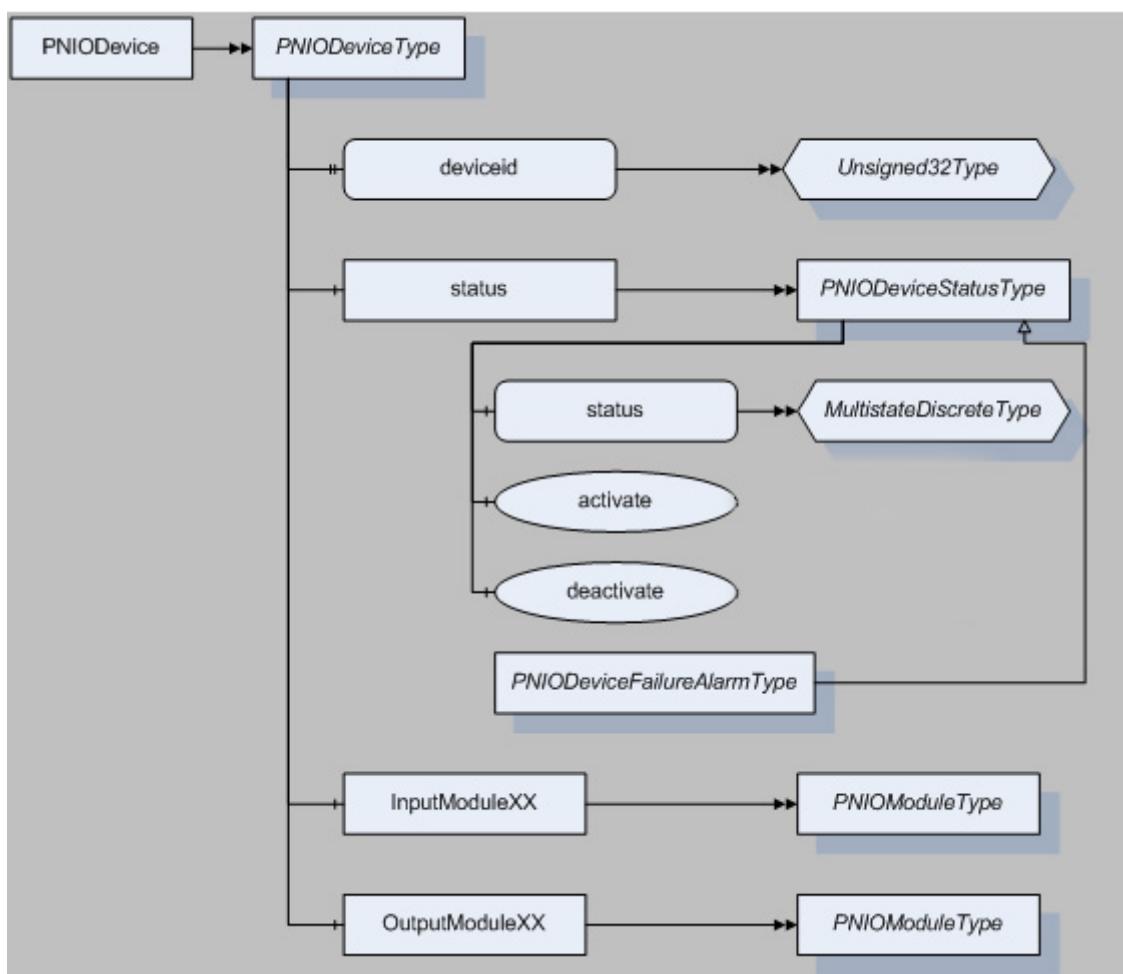


Bild 2-71 Device-Objekt

Das Device-Objekt des PROFINET-IO-Controller-Klassenbaums befindet sich in Bild 2-48. Die PROFINET-IO-Devices haben im Rahmen der Adressierung nur eine untergeordnete Bedeutung und werden vom Protokollstack nur beim Verbindungsaufbau genutzt. An der

OPC-Server-Client-Schnittstelle werden die PROFINET-IO-Devices ausschließlich über deren Module angesprochen. Auch wenn ein Kommando (z. B. Device activate/deactivate) Rückwirkungen auf alle Module eines PNIO-Device hat, so genügt es für das Kommando, ein beliebiges Modul dieses PNIO-Device anzusprechen.

Aus der Projektierung erhält der OPC-Server die diesem PNIO-Device zugeordneten E/A- und Diagnoseadressen, welche eindeutig sind und für die OPC-UA-Nodelds und die Kommunikation verwendet werden.

Alle Datenvariablen eines Device-Objekts erhalten jedoch im ergänzenden UA-Attribut "DisplayName" für den jeweiligen Device-Anteil des Names den mit STEP 7 projektierten Gerätenamen.

Das PROFINET-IO-Device-Objekt hat die Aufgabe, eine globale Statusinformation über das PNIO-Device bereitzustellen. Der PNIO-Device-Status wird in einem untergeordneten Objekt verwaltet und bietet dazu auch zwei Methoden an, um das PNIO-Device mit allen seinen Modulen zu aktivieren bzw. zu deaktivieren.

## 2.13.8 Datenvariablen und Methoden des Device-Objekts

### Syntax der Datenvariablen und Methoden des Device-Objekts

Es gibt vier Möglichkeiten:

- `ctrl<Steckplatz>.dev<DeviceNummer>.<Datenvariable>`
- `ctrl<Steckplatz>.dev<DeviceNummer>.<Methode>`
- `ctrl<Steckplatz>.dev<DeviceNummer>.status.<Datenvariable>`
- `ctrl<Steckplatz>.dev<DeviceNummer>.status.<Methode>`

### Erklärungen

**ctrl**

Kennzeichen für den PNIO-Controller

**<Steckplatz>**

Steckplatz im "Komponenten Konfigurator"

**dev**

Kennzeichen für ein PNIO-Device

**<DeviceNummer>**

PNIO-Device-Nummer entsprechend der STEP 7- bzw. TIA-Portal-Projektierung

**status**

Statusobjekt des PNIO-Device

## &lt;Datenvatiable&gt; / &lt;Methode&gt;

Datenvatiable / Methode	Beschreibung	
deviceid	PNIO-Device-Nummer aus STEP 7 bzw. TIA-Portal Property vom Typ "UInt32", nur lesbar	
status	Status des PNIO-Device Datenvatiable vom UA-Typ "MultistateDiscreteType", nur lesbar	
	0	FAILURE
activate()	PNIO-Device aktivieren Die OPC-UA-Methode "activate()" kann nur geschrieben werden. Sie besitzt keine Argumente. Ein erfolgreicher Abschluss des Methodenaufrufs zeigt an, dass die Aktivierung erfolgreich zum PNIO-Device transportiert wurde. Ob das PNIO-Device aus Sicht des PNIO-Controllers wirklich aktiv ist, muss über die Datenvatiable "status" überprüft werden.	
deactivate()	PNIO-Device deaktivieren Die OPC-UA-Methode "deactivate()" kann nur geschrieben werden. Sie besitzt keine Argumente. Ein erfolgreicher Abschluss des Methodenaufrufs zeigt an, dass die Deaktivierung erfolgreich zum PNIO-Device transportiert wurde. Ob das PNIO-Device aus Sicht des PNIO-Controllers wirklich inaktiv ist, muss über die Datenvatiable "status" überprüft werden.	

**Beispiele:**

```

ctrl4.dev32.deviceid
ctrl4.dev32.status.status
ctrl4.dev32.status.activate()
ctrl4.dev32.status.deactivate()

```

**2.13.9 PROFINET-IO-Modulobjekte****Allgemein**

Die eigentliche PNIO-Kommunikation findet über IO-Module statt. Ein IO-Modul ist immer einem PNIO-Device zugeordnet. Typ und Länge eines IO-Moduls erhält der OPC-Server aus der STEP 7- bzw. TIA-Portal-Projektierung.

**Syntax der PNIO-Modulobjekte**

Es gibt zwei Möglichkeiten:

- `ctrl<Steckplatz>.i<Adresse>`
- `ctrl<Steckplatz>.q<Adresse>`

## Erklärungen

**ctrl**

Kennzeichen für den PNIO-Controller.

**<Steckplatz>**

Steckplatz im "Komponenten Konfigurator".

**q**

Kennzeichen für einen Ausgang. Ausgänge sind les- und schreibbar.

**i**

Kennzeichen für einen Eingang. Eingänge sind nur lesbar.

**<Adresse>**

Logische Adresse des IO-Moduls.

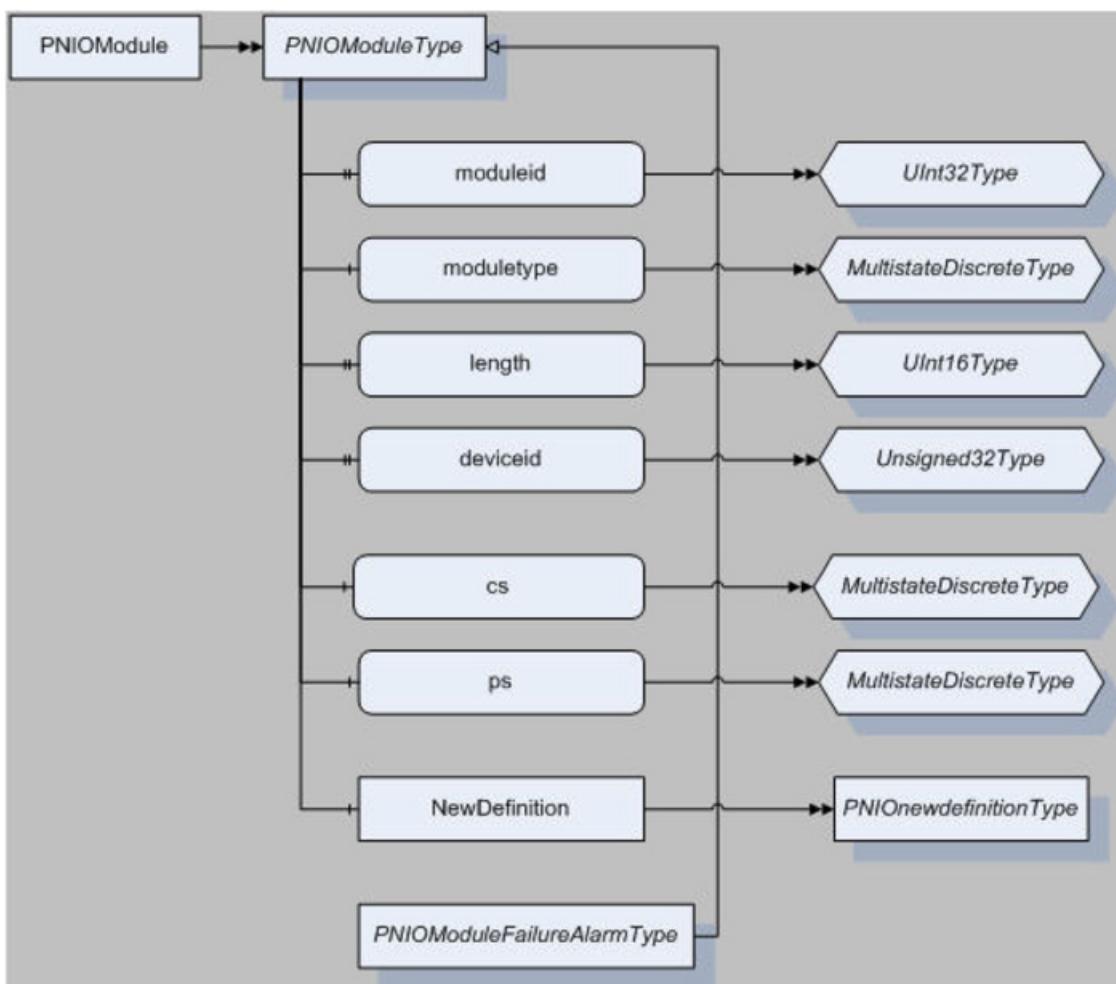


Bild 2-72 PROFINET-IO-Modulobjekte

### 2.13.9.1 Datenvariablen der PROFINET-IO-Module

#### Syntax für Datenvariablen der PNIO-Modulobjekte

Es gibt zwei Möglichkeiten:

- `ctrl<Steckplatz>.i<Adresse>.<IOModulVariable>`
- `ctrl<Steckplatz>.q<Adresse>.<IOModulVariable>`

#### Erklärungen

**ctrl**

Kennzeichen für den PNIO-Controller.

**<Steckplatz>**

Steckplatz im "Komponenten Konfigurator".

**q**

Kennzeichen für ein IO-Ausgangsmodul. Ausgänge sind les- und schreibbar.

**i**

Kennzeichen für ein IO-Eingangsmodul. Eingänge sind nur lesbar.

**<Adresse>**

Logische Adresse des IO-Moduls.

**<IOModulVariable>**

moduleid	Moduladresse aus STEP 7 bzw. TIA-Portal Property vom Typ "UInt32", nur lesbar	
modulotype	IO-Modultyp aus STEP 7 bzw. TIA-Portal Datenvariable vom Typ "MultistateDiscreteType", nur lesbar	
0	UNKNOWN	
1	I (Input) Der aktuell vom PROFINET-IO-UA-Server gelieferte Wert ist I.	
2	Q (Output) Der aktuell vom PROFINET-IO-UA-Server gelieferte Wert ist O.	
3	I (Diag) Der aktuell vom PROFINET-IO-UA-Server gelieferte Wert ist I. Keine zyklische Daten, nur Datensätze.	
Variable zu Informationszwecken		
length	Modullänge in Bytes aus STEP 7 bzw. TIA-Portal Property vom Typ "UInt16", nur lesbar Besondere Modullängen:	
	0	kein Modul aus dem Peripherieadressbereich (modulotype = 3)
deviceid	Zum IO-Modul gehörige PNIO-Device-Nummer aus STEP 7 bzw. TIA-Portal Property vom Typ "UInt32", nur lesbar	

**Beispiele:**

```
ctrl14.q27.moduleid
ctrl14.q28.modulename
ctrl14.q28.length
ctrl14.q28.deviceid
```

**2.13.9.2 IO-Datenvariablen der PROFINET-IO-Module****Allgemein**

Das Lesen und Schreiben von Nodelds zyklischer IO-Daten ist der häufigste Anwendungsfall. Hierbei wird ein einfacher Zugriff auf den PNIO-Adressraum ermöglicht. Der Anwender muss den PNIO-Controller-Namen und die logische Adresse des IO-Moduls aus der Projektierung vorgeben.

**Syntax für Datenvariablen der PNIO-Modulobjekte**

Es gibt zwei Möglichkeiten:

- `ctrl<Steckplatz>.i<Adresse>.<Offset>{,<PNIOTyp>{,<Anzahl>}}`
- `ctrl<Steckplatz>.q<Adresse>.<Offset>{,<PNIOTyp>{,<Anzahl>}}`

**Erklärungen****ctrl**

Kennzeichen für den PNIO-Controller.

**<Steckplatz>**

Steckplatz im "Komponenten Konfigurator".

**q**

Kennzeichen für ein IO-Ausgangsmodul. Ausgänge sind les- und schreibbar.

**i**

Kennzeichen für ein IO-Eingangsmodul. Eingänge sind nur lesbar.

**<Adresse>**

Logische Adresse des IO-Moduls.

**<Offset>**

Byteadresse im Bereich der Daten des IO-Moduls, das angesprochen werden soll.

## &lt;PNIOTyp&gt;

Datentyp	OPC-UA-Datatype	Hinweis
x<Bitadresse>	Boolean	Bit (bool) Zusätzlich zum Byte-Offset im Bereich ist noch die <Bitadresse> im jeweiligen Byte anzugeben. Wertebereich 0...7
b	Byte ByteString	Byte (unsigned) Wird als Defaultwert verwendet, falls kein <PNIOType> angegeben ist.
w	UInt16	Wort (unsigned)
dw	UInt32	Doppelwort (unsigned)
lw	UInt64	Langwort (unsigned)
c	SByte	Byte (signed)
i	Int16	Wort (signed)
di	Int32	Doppelwort (signed)
li	Int64	Langwort (signed)
r	Float	Fließkomma (4 Byte)
lr	Double	Fließkomma (8 Byte)
s<Stringlänge>	String	Es ist noch die für den String reservierte <Stringlänge> anzugeben. Beim Schreiben können auch kürzere Strings geschrieben werden, wobei die übertragene Datenlänge immer die reservierte Stringlänge in Byte ist. Die nicht benötigten Bytes werden mit dem Wert 0 gefüllt.

## &lt;Anzahl&gt;

Anzahl der Elemente. Der Datentyp der Variable ist ein Feld mit Elementen des angegebenen Formats. Die Angabe einer Anzahl von Feldelementen führt immer zur Bildung eines Feldes vom entsprechenden Typ, auch wenn nur ein einziges Feldelement adressiert wird.

**Beispiele:**

- ctrl4.i10.0,w                         Eingangswort ab Byteadresse 10
- ctrl4.q17.3                             Ausgangsbyte ab Byteadresse 3 (BYTE-Default)
- ctrl4.i10.0,x0                         Eingangsbit ab Byteadresse 0, Bit 0
- ctrl4.q17.1,x4,16                     Feld von 16 Ausgangsbits ab Byteadresse 1, Bit 4
- ctrl4.i27355.100,s32,3                 Feld mit 3 Zeichenketten zu je 32 Zeichen

**2.13.9.3 Datenvariablen für IOPS und IOCS****Allgemein**

Mit den Items für den Status (IOPS und IOCS) kann ein OPC-Client die jeweiligen Status-Bytes einer logischen Adresse abfragen bzw. setzen.

Die folgende Grafik zeigt den Datenstatus und den Wertefluss im Prozessabbild.

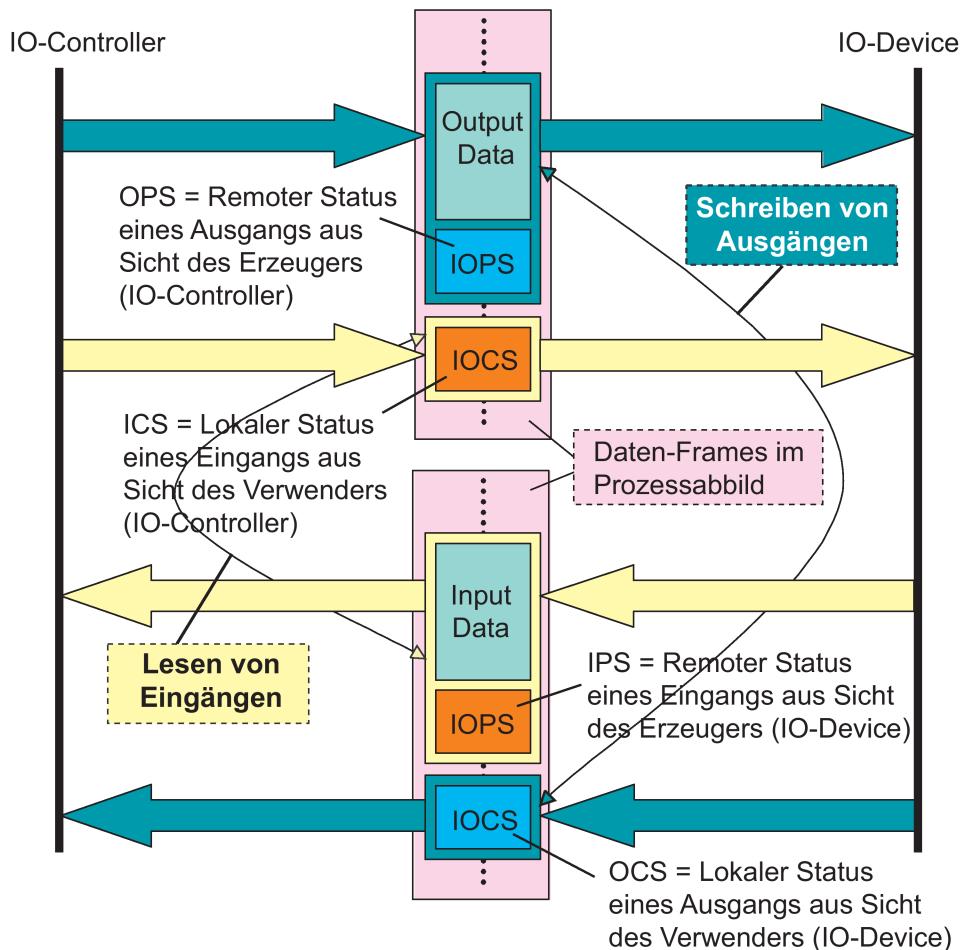


Bild 2-73 Übersicht des Datenstatus im Prozessabbild bei der PROFINET-IO-Kommunikation zwischen IO-Controller und einem IO-Device.

Beim Schreiben von Ausgangsdaten in das Prozessabbild durch den OPC-Client gibt dieser den IO-Daten den remoten Status des Ausgangs aus seiner Sicht (Erzeuger) mit "OPS". Dabei teilt das Device den eigenen, lokalen Status des Ausgangs aus seiner Sicht (Verwender) mit "OCS".

Beim Lesen von Eingangsdaten aus dem Prozessabbild durch den OPC-Client gibt dieser den eigenen, lokalen Status des Eingangs aus seiner Sicht (Verwender) mit "ICS". Dabei teilt das Device den remoten Status des Eingangs aus seiner Sicht (Erzeuger) mit "IPS".

### Hinweis

#### Lesender Zugriff auf zyklische Ausgangsdaten

Das Lesen, Beobachten oder Aktivieren von OPC-Items/nodes, die zyklische Ausgangsdaten abbilden, bewirkt eine einmalige bzw. ständige Aktualisierung der unterlagerten PROFINET IO-Protokoll-Schnittstelle mit den im OPC-Server-Cache enthaltenen Daten incl. Provider-Status (PNIO\_data\_write()). Bitte beachten Sie, dass nach Wiederkehr eines PROFINET IO-Devices oder Moduls ab dem Zeitpunkt der Aktualisierung diese Ausgangsdaten wieder zum Device übertragen werden.

## Syntax der Datenvariablen für IOPS und IOCS

Es gibt folgende vier Möglichkeiten:

**IPS-Variable:**

- `ctrl<Steckplatz>.i<Adresse>.ps`

**OPS-Variable**

- `ctrl<Steckplatz>.q<Adresse>.ps`

**ICS-Variable:**

- `ctrl<Steckplatz>.i<Adresse>.cs`

**OCS-Variable:**

- `ctrl<Steckplatz>.q<Adresse>.cs`

## Erklärungen

**ctrl**

Kennzeichen für den PNIO-Controller.

**<Steckplatz>**

Steckplatz im "Komponenten Konfigurator".

**q**

Kennzeichen für ein IO-Ausgangsmodul. Ausgänge sind les- und schreibbar.

**i**

Kennzeichen für ein IO-Eingangsmodul. Eingänge sind nur lesbar.

**<Adresse>**

Logische Adresse des IO-Moduls.

**ps**

Providerstatus eines IO-Moduls.

**cs**

Consumerstatus eines IO-Moduls.

Bezeichnung	Erläuterung
IPSVariable	<p>Remoter Status einer Eingangs-Adresse</p> <p>Datenvariable vom UA-Typ "MultistateDiscreteType", nur lesbar</p> <p>Der Provider eines Eingangs ist das Device. Das Device teilt hierbei den Zustand der Eingangsdaten mit.</p> <p>Das Beobachten des IPS wird pollend durchgeführt.</p> <p>Die Verwendung des Items ist für eine OPC-Applikation nicht unbedingt erforderlich, da sich der Inhalt des Items in der OPC-Quality der mit derselben Adresse verbundenen Eingangs-Items wiederfindet.</p>
ICSVariable	<p>Lokaler Status einer Eingangs-Adresse</p> <p>Datenvariable vom UA-Typ "MultistateDiscreteType", Schreib- und lesbar</p> <p>Default: <i>GOOD</i> = 0</p> <p>Der Consumer eines Eingangs ist der OPC-Client. Der OPC-Client teilt hierbei dem Device den Zustand der Eingangsdaten aus seiner Sicht mit. Üblicherweise ist dieser Zustand aber immer <i>GOOD</i>.</p> <p>Beim Lesen von IO Daten wird immer der zuletzt von der OPC-Applikation geschriebene lokale Status einer Eingangsadresse schreibend dem Partnergerät mitgegeben. Hat die OPC-Applikation das Item nicht angelegt oder noch nicht mit einem Wert beschrieben, so wird bei einem Lesen der zugehörigen Eingangsdaten der Wert <i>GOOD</i> übergeben.</p> <p>Das Lesen des Status durch den OPC-Client wird über den Cache abgewickelt.</p>
OCSVariable	<p>Lokaler Status einer Ausgangs-Adresse</p> <p>Datenvariable vom UA-Typ "MultistateDiscreteType", nur lesbar</p> <p>Der Consumer eines Ausgangs ist das Device. Das Device teilt hierbei den Zustand der Ausgangsdaten aus seiner Sicht mit.</p> <p>Das Lesen des Status erfolgt dadurch, dass der OPC-Client zuvor oder gleichzeitig ein mit derselben Adresse verbundenes Ausgangs-Item erfolgreich beschreiben konnte.</p> <p>Das Beobachten des OCS wird pollend durchgeführt, dabei werden der Funktion immer gleichzeitig die zuletzt vom OPC-Client geschriebenen Ausgangsdaten übergeben. Konnte der OPC-Client noch keinen Wert erfolgreich schreiben, so liefert das Lesen von Device einen Fehler bzw. die OPC-Quality <i>bad</i>.</p>
OPSVariable	<p>Remoter Status einer Ausgangs-Adresse</p> <p>Datenvariable vom UA-Typ "MultistateDiscreteType", Schreib- und lesbar</p> <p>Default: <i>GOOD</i> = 0</p> <p>Der Provider eines Ausgangs ist der OPC-Client. Der OPC-Client teilt hierbei dem Device den Zustand der Ausgangsdaten aus seiner Sicht mit. Normalerweise ist dieser Zustand aber immer <i>GOOD</i>.</p> <p>Beim Schreiben von IO Daten werden immer der zuletzt von der OPC-Applikation geschriebene lokale Status der Ausgangsdaten und die Ausgangsdaten einer Ausgangsadresse schreibend mitgegeben. Hat die OPC-Applikation das IO-Item nicht angelegt oder noch keinen Ausgangswert erfolgreich geschrieben, so wird bei einem Schreiben der zugehörigen Ausgangsdaten der Wert <i>GOOD</i> übergeben.</p> <p>Beim Schreiben eines neuen Status durch den OPC-Client wird der Datenwert verwendet, der vom OPC-Client zuvor (oder gleichzeitig) mit der Adresse des verbundenen Ausgangs-Items erfolgreich beschrieben wurde.</p> <p>Konnte der OPC-Client noch keinen Wert erfolgreich schreiben, so liefert das Schreiben des Items den speziellen Statuscode OPCUA_GoodClamped. Der geschriebene Status wird dann beim nächsten erfolgreichen Schreiben von dem zugehörigen Ausgangs-Item geschrieben.</p>

### Aufbau des Statuswerts

Die Datenvariable vom UA-Typ "MultistateDiscreteType" hat immer folgenden Aufbau:

0 = GOOD  
1 = BAD

Im Normalfall müssen die ctrl.xxx.cs und ctrl.xxx.ps an der UA-Schnittstelle nicht geschrieben werden, da diese Werte durch das Schreiben/Lesen der Variablen selbst gesetzt werden. Nur wenn dem Device die fehlende Bereitstellungs- bzw. Verarbeitungsmöglichkeit mitgeteilt werden soll, wird der Status explizit gesetzt.

#### Beispiele:

ctrl1.i10.ps	Remoter Status der logischen Eingangsadresse 10
ctrl1.q20.cs	Lokaler Status der logischen Ausgangsadresse 20

## 2.13.9.4 Datensatz-Datenvariablen der PROFINET-IO-Module

### Allgemein

Mit den Datensatz-Datenvariablen ("data record" = "dr") können gerätespezifische Informationen ausgelesen oder geschrieben werden. Die möglichen Adressen sind an projektierte Modul- oder Diagnoseadressen gekoppelt.

### Datensätze und Längenangabe

Die Syntax unterscheidet Datensätze mit und ohne Längenangabe.

Die Längenangabe wird immer beim Schreiben von Datensätzen benötigt, wenn mit der Variablen nicht der gesamte Datensatz als Bytestring adressiert wird und damit die Länge über die zu schreibenden Daten bekannt sind. Dies ist der Fall, wenn die Subelementadressierung verwendet wird.

Wird ohne Längenangabe und ohne Subelementadressierung geschrieben, ermittelt der OPC-UA-Server die Länge des Datensatzes anhand der Anzahl zu schreibender Bytes.

Ohne Längenangabe ist das Lesen und Schreiben von Datensätzen auf 480 Bytes beschränkt.

### Subelementadressierung

Die Client-seitige Strukturierung von Anwenderdatensätzen wird durch die Subelementadressierung unterstützt. Dabei werden aus dem Datensatz Teilbereiche adressiert.

Nur lesbare Datenvariablen: Ohne Längenangabe des Datensatzes ist die Adressierung eines Subelements möglich. Da die gerätespezifische Länge des Datensatzes vorher nicht bekannt ist oder sogar wechselt, können Subelemente auch außerhalb der tatsächlich erhaltenen Länge verwendet werden. Der OPC-UA-Statuscode zeigt dann allerdings einen Fehler.

Les- und schreibbare DatenvARIABLEN: Das Schreiben einzelner oder gleichzeitig mehrerer Subelemente über den gesamten Datensatz erfolgt konsistent über den mit der Längenangabe spezifizierten Bereich. Etwaige Lücken werden mit den zuletzt geschriebenen Werten (nicht gelesenen) ergänzt, bzw. mit "0"-Werten, falls dieser Datensatz noch nie geschrieben wurde.

## Syntax der Datensatz-Datenvariablen für PROFINET IO-Module

Es gibt vier Möglichkeiten:

- `ctrl<Steckplatz>.i<Adresse>.dr<Nummer>{.<DRLänge>,<Offset>,<PNIOTyp>{,<Anzahl>}}`
- `ctrl<Steckplatz>.q<Adresse>.dr<Nummer>{.<DRLänge>,<Offset>,<PNIOTyp>{,<Anzahl>}}`
- `ctrl<Steckplatz>.i<Adresse>.dr<Nummer>.<Offset>,<PNIOTyp>{,<Anzahl>}`
- `ctrl<Steckplatz>.q<Adresse>.dr<Nummer>.<Offset>,<PNIOTyp>{,<Anzahl>}`

## Erklärungen

**ctrl**

Kennzeichen für den PNIO-Controller.

**<Steckplatz>**

Steckplatz im "Komponenten Konfigurator".

**q**

Kennzeichen für ein IO-Ausgangsmodul. Ausgänge sind les- und schreibbar.

**i**

Kennzeichen für ein IO-Eingangsmodul. Eingänge sind nur lesbar.

**<Adresse>**

Logische Adresse des IO-Moduls.

**dr**

Kennzeichen für Datensatz-Datenvariablen.

**<Nummer>**

Datensatznummer des IO-Moduls.

**<DRLänge>**

Datensatzlänge des IO-Moduls.

**<Offset>**

Byteadresse im Datensatz für das IO-Modul, das angesprochen werden soll.

## &lt;PNIOTyp&gt;

Datentyp	OPC-UA-Datatype	Hinweis
x<Bitadresse>	Boolean	Bit (bool) Zusätzlich zum Byte-Offset im Bereich ist noch die <Bitadresse> im jeweiligen Byte anzugeben. Wertebereich 0...7
b	Byte ByteString	Byte (unsigned) Wird als Defaultwert verwendet, falls kein <PNIOType> angegeben ist.
w	UInt16	Wort (unsigned)
dw	UInt32	Doppelwort (unsigned)
lw	UInt64	Langwort (unsigned)
c	SByte	Byte (signed)
i	Int16	Wort (signed)
di	Int32	Doppelwort (signed)
li	Int64	Langwort (signed)
r	Float	Fließkomma (4 Byte)
lr	Double	Fließkomma (8 Byte)
s<Stringlänge>	String	Es ist noch die für den String reservierte <Stringlänge> anzugeben. Beim Schreiben können auch kürzere Strings geschrieben werden, wobei die übertragene Datenlänge immer die reservierte Stringlänge in Byte ist. Die nicht benötigten Bytes werden mit dem Wert 0 gefüllt.

## &lt;Anzahl&gt;

Anzahl der Elemente. Der Datentyp der Variable ist ein Feld mit Elementen des angegebenen Formats. Die Angabe einer Anzahl von Feldelementen führt immer zur Bildung eines Feldes vom entsprechenden Typ, auch wenn nur ein einziges Feldelement adressiert wird.

**Beispiele:**

ctrl1.i10.dr61450

Datensatz-Diagnoseinformationen des Device mit logischer Eingangsadresse 10. Gesamter Datensatz, Byte[], Les- und schreibbar

ctrl1.q99.dr32788,100.8,w,3

Datensatz mit Datensatzlänge, Subelementadressierung ab Byte 8, UInt16[3], Les- und schreibbar

ctrl1.q99.dr32788.8,dw

Datensatz ohne Datensatzlänge, Subelementadressierung ab Byte 8, UInt32, nur lesen

## 2.13.10 PROFINET-IO-OPC-UA-Templates

### 2.13.10.1 Template-Datenvariablen

Sie haben mit den IO-Datenvariablen und den Datensatz-Datenvariablen für das OPC-UA-PNIO-Protokoll flexible Einstellmöglichkeiten, um die Prozessdaten Ihrer Anlage in den gewünschten Datenformaten zu erhalten.

Die Vielfalt der Adressierungsmöglichkeiten lässt sich allerdings nicht in einen vollständig durchsuchbaren Namensraum fassen. Bereits ein IO-Modul mit der Länge eines einzelnen Bytes besitzt etwa 40 verschiedene Datenformatoptionen – angefangen vom Byte, SByte, Felder mit einem Element davon, einzelne Bits, Felder von Bits mit bis zu 8 Feldelementen an unterschiedlichen Bitoffsets beginnend.

Der OPC-UA-Server unterstützt den Anwender deshalb mit den Template-Datenvariablen im PNIO-Namensraum. In einem für einen OPC-UA-Client typischen Texteingabefeld können diese Templates durch Ändern einiger weniger Zeichen in gültige ItemIDs verwandelt werden.

#### Beispiel:

ctrl4.i<x>.<o>,dw	Template für ein UInt32 eines Eingangsmoduls
-------------------	--

Durch Ersetzen von <x> mit der Moduladresse und <o> dem Offset innerhalb des Moduls erhalten Sie eine gültige Nodeld. → ctrl4.i8.15,dw

#### Weiteres Beispiel:

ctrl4.i<x>.dr<dr>,<l>.<o>,c,<c>	Template für eine Datensatzvariable, Char[]
---------------------------------	---

→ ctrl4.i125.dr61450,100.0,c,100

Ein großer Vorteil dieses Konzepts ist, dass es von nahezu allen OPC-UA-Clients eingesetzt werden kann, ohne dass Anpassungen der Clients erforderlich sind. Ein weiterer Vorteil ist, dass die Realisierung vergleichsweise wenig aufwändig ist und dieses Konzept auch auf andere Protokollserver einschließlich COM der SIMATIC NET OPC-Umgebung übertragen werden kann.

#### Hinweis

Die Verwendbarkeit von OPC-UA-PNIO-Template-Datenvariablen kann im Konfigurationsprogramm "Kommunikations-Einstellungen" unter "OPC-Protokollauswahl" aktiviert und deaktiviert werden.

### Template-Datenvariablen innerhalb der Baumstruktur

Die Template-Datenvariablen sind neben den ihnen entsprechenden Ordnern in der Namensraum-Darstellung eingesortiert, so dass sie bei Bedarf leicht genutzt werden können.

### 2.13.10.2 Verzeichnis der Template-Datenvariablen

#### Spezielle Nutzung einiger Attribute der Template-Datenvariablen

Bei den Template-Variablen werden die Attribute "NodeID", "BrowseName" und "Description" besonders intelligent eingesetzt.

#### Syntax der Template-Datenvariablen

Es gibt zwei Möglichkeiten:

- `ctrl<Steckplatz>.i<x>,<o>,<PNIOTypTemplate>,<c>`
- `ctrl<Steckplatz>.q<x>,<o>,<PNIOTypTemplate>,<c>`

#### Erklärungen

**ctrl**

Kennzeichen für den PNIO-Controller.

**<Steckplatz>**

Steckplatz im "Komponenten Konfigurator".

**q**

Kennzeichen für ein IO-Ausgangsmodul. Ausgänge sind les- und schreibbar.

**i**

Kennzeichen für ein IO-Eingangsmodul. Eingänge sind nur lesbar.

**<x>**

Platzhalter für Nummer des Moduls, das den Ein- oder Ausgangsbereich enthält.

**<o>**

Platzhalter für Byteoffset im Adressraum des IO-Moduls.

**<PNIOTypTemplate>**

Ein PNIO-Template-Datentyp wird im OPC-UA-Server in den entsprechenden OPC-UA-Datentyp umgewandelt. Die folgende Tabelle listet den Typ-Bezeichner und den entsprechenden OPC-Datentyp auf, in dem der Variablenwert dargestellt werden kann.

Datentyp	OPC-UA-Datentyp	Hinweis
x<bit>	Boolean	Platzhalter für den Bitoffset (0...7)
b	Byte	Platzhalter für Byte (unsigned)
w	UInt16	Platzhalter für Wort (unsigned)
dw	UInt32	Platzhalter für Doppelwort (unsigned)
lw	UInt64	Platzhalter für Langwort (unsigned)
c	SByte	Platzhalter für Byte (signed)
i	Int16	Platzhalter für Wort (signed)
di	Int32	Platzhalter für Doppelwort (signed)
li	Int64	Platzhalter für Langwort (signed)
r	Float	Platzhalter für Fließkomma (4 Byte)

Datentyp	OPC-UA-Datentyp	Hinweis
lr	Double	Platzhalter für Fließkomma (8 Byte)
s<sl>	String	Platzhalter für die Länge des Strings

&lt;c&gt;

Platzhalter für Anzahl der Elemente.

**Beispiele:**

NodeId	BrowseName	Beschreibung
ctrl.i<x>.<o>,x<bit>,<c>	template input bit	<x> Adresse des Moduls <o> Offset innerhalb des Moduls <bit> Bitoffset (0...7) <c> Größe des Feldes
ctrl.i<x>.<o>,b<c>	template input byte	<x> Adresse des Moduls <o> Offset innerhalb des Moduls <c> Größe des Feldes
ctrl.i<x>.<o>,w,<c>	template input UInt16	<x> Adresse des Moduls <o> Offset innerhalb des Moduls <c> Größe des Feldes
ctrl.i<x>.<o>,s<sl>,<c>	template input string	<x> Adresse des Moduls <o> Offset innerhalb des Moduls <sl> Länge des String <c> Größe des Feldes
ctrl.q<x>.<o>,x<bit>,<c>	template output bit	<x> Adresse des Moduls <o> Offset innerhalb des Moduls <bit> Bitoffset (0...7) <c> Größe des Feldes
ctrl.i<x>.dr<dr>,<l>.<o>,b,<c>	template input dataset byte	<x> Adresse des Moduls <dr> Datenaufzeichnungsnummer <l> Länge der Datei <o> Offset innerhalb des Moduls <c> Größe des Feldes

## 2.14 Server-Diagnose

### Was ist die Server-Diagnose?

OPC-Client- und OPC-Server-Diagnoseinformationen werden ab der "SIMATIC NET PC Software V8.0" für OPC Data Access (COM, XML, UA) angeboten. Es kann z. B. die Anzahl der verbundenen Clients und die Anzahl der aktiven OPC-Gruppen ausgelesen werden. Zusätzlich sind Versionsdaten der OPC-Server auslesbar.

---

#### Hinweis

Die OPC-Server-Diagnose (im Namensraum unter SERVER:) steht nur beim Server OPC.SimaticNET mit mehreren ausgewählten Protokollen (in "Kommunikations-Einstellungen" ... Protokollauswahl") zur Verfügung. Den Einzel-Protokoll-Data-Access-OPC-Servern wie OPC.SimaticNET.DP und OPC.SimaticNET.PD nicht. Auch nicht den OPC Alarm & Event Servern.  
Die Einschränkung gilt auch für OPC XML Data Access.

---

### 2.14.1 Protokoll-ID

#### Was ist die Protokoll-ID?

Die Protokoll-ID für die Server-Diagnose lautet SERVER:

Diese ID wird für OPC DA und OPC XML DA verwendet.

Für OPC UA DA wird die Server-Diagnose von der Norm vorgegeben, dort finden sich die Diagnose Nodes im Namespace 0 unter Server.

### 2.14.2 OPC-DA-Server-Diagnose-Items

#### Welche OPC-DA-Server-Diagnose-Items gibt es?

Die OPC-Server bieten einen hierarchischen Namensraum an, in dem sich die verschiedenen Diagnose-Items befinden. In Anlehnung an die OPC-UA-Norm werden die Diagnoseinformationen wie folgt gegliedert:

Server:

Capabilities

MinSupportedUpdateRate

DiagnosticsSummary

CurrentSessionCount

CumulatedSessionCount

	CurrentSubscriptionCount
	CumulatedSubscriptionCount
VendorInfo	
	ComponentVersion
	VendorInfoString

Diagnose ItemID	Datentyp	Beschreibung
Server\Capabilities\MinSupportedUpdateRate	VT_UI4	Minimale Update-Rate für eine OPC-Gruppe
Server\DiagnosticsSummary\CurrentSessionCount	VT_UI4	Aktuelle Anzahl der OPC-Client-Sitzungen (bzw. Anzahl der OPC-Serverinstanzen)
Server\DiagnosticsSummary\CumulatedSessionCount	VT_UI4	Gesamtanzahl der OPC-Client-Sitzungen, die seit dem OPC-Serveranlauf gestartet wurden. Einige Sitzungen können aber inzwischen beendet worden sein.
Server\DiagnosticsSummary\CurrentSubscriptionCount	VT_UI4	Aktuelle Anzahl der OPC-DA-Subscriptions, d.h. der OPC-Gruppen, die aktiviert sind und OnDataChange-Callbacks an den OPC-Client schicken können.
Server\DiagnosticsSummary\CumulatedSubscriptionCount	VT_UI4	Gesamtanzahl der OPC-DA-Subscriptions, die seit dem OPC-Serveranlauf gestartet wurden. Einige können aber inzwischen beendet worden sein.
Server\VendorInfo\ComponentVersion	VT_BSTR	OPC-Server-Versionstring, z.B. "V03.08.00.00_01.14.00.01"
Server\VendorInfo\VendorInfoString	VT_BSTR	OPC-Server-Bezeichnung, z.B. "SIMATIC NET OPC-Server DataAccess-V2.05/3.0 (C) SIEMENS AG 2010"

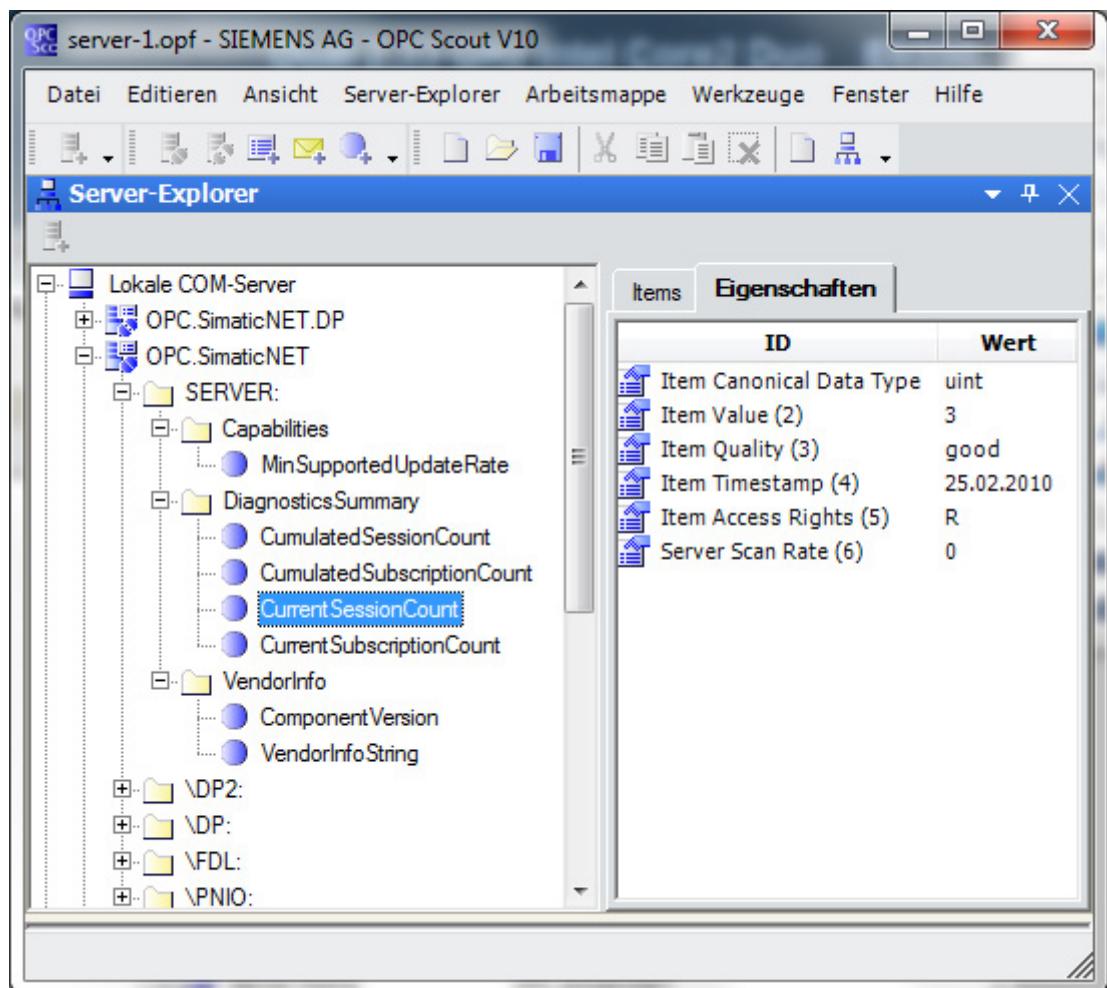


Bild 2-74 OPC-Server-Diagnose-Items für COM/DCOM OPC.SimaticNET

### Wie sehen die OPC-XML DA-Server-Diagnose-Items aus?

Es gelten dieselben Diagnose-Items wie im OPC DA-Server, bis auf MinSupportedUpdateRate, da es keine Callbacks bei OPC XML-DA gibt. Da auch keine Gruppenanmeldung benutzt wird, wird die Session-Anzahl CurrentSessionCount folgendermaßen umgesetzt.

Unter Session/Sitzung versteht man hier eine länger andauernde Kommunikationsbeziehung zwischen einem Client und einem Server. Von den XML DA-Methoden wirkt in diesem Sinn nur die Subscription. Alle anderen Methoden (GetStatus, Browse, Read, Write) sind einzelne Zugriffe, die keiner Session zugeordnet werden können. XML DA Subscriptions werden durch den Aufruf "Subscribe" angelegt. Der Client besorgt sich die Daten dieser Subscriptionen per PolledRefresh. Dabei ist sichergestellt, dass alle Subscriptions einer Liste demselben Client und damit zu derselben "Session" zählen. Der SessionCount berechnet sich aus der Zahl vorhandenen Subscriptions eines Client.

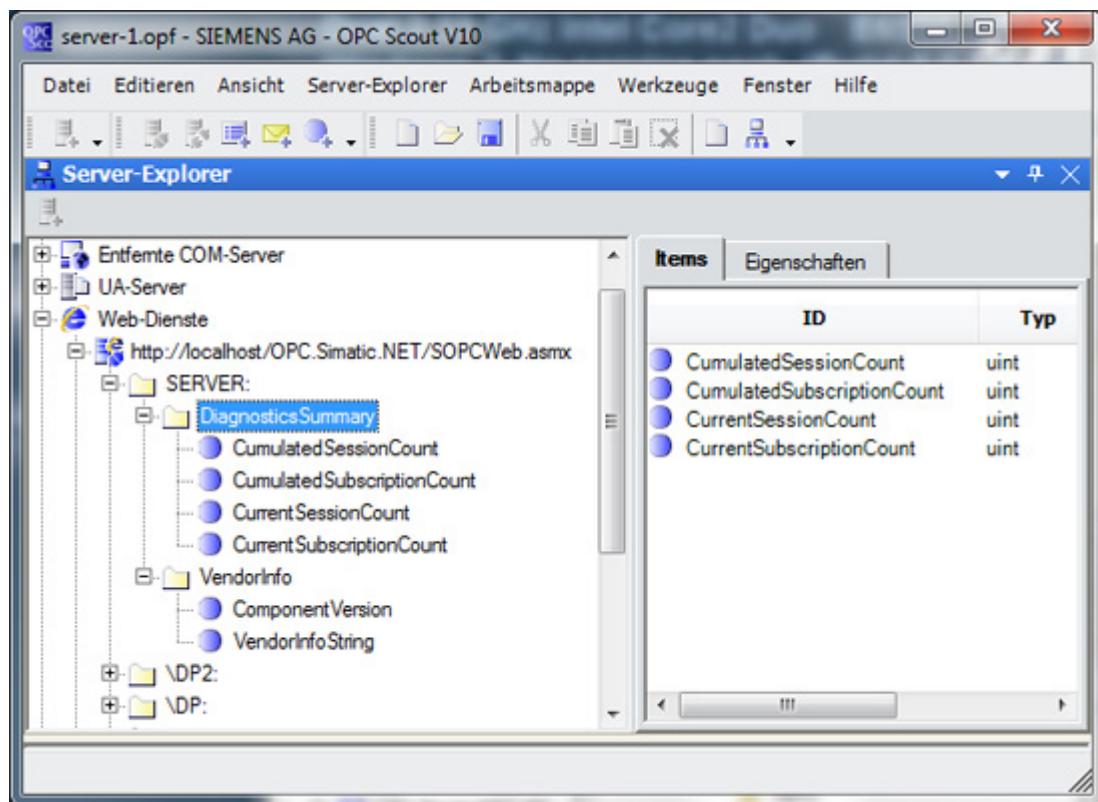


Bild 2-75 OPC-Server-Diagnose-Items für XML DA  
http://localhost/OPC.Simatic.NET/SOPCWeb.asmx

## Was bietet die OPC-UA-Server-Diagnose?

Die OPC-UA-Server-Diagnose dient als Vorbild und bietet die gleichen und darüber hinaus weitere Diagnose- und Informations-Nodes. Einen Eindruck bietet nachfolgendes Bild. Ziehen Sie hierfür die OPC-UA-Spezifikationen zur Referenz hinzu.

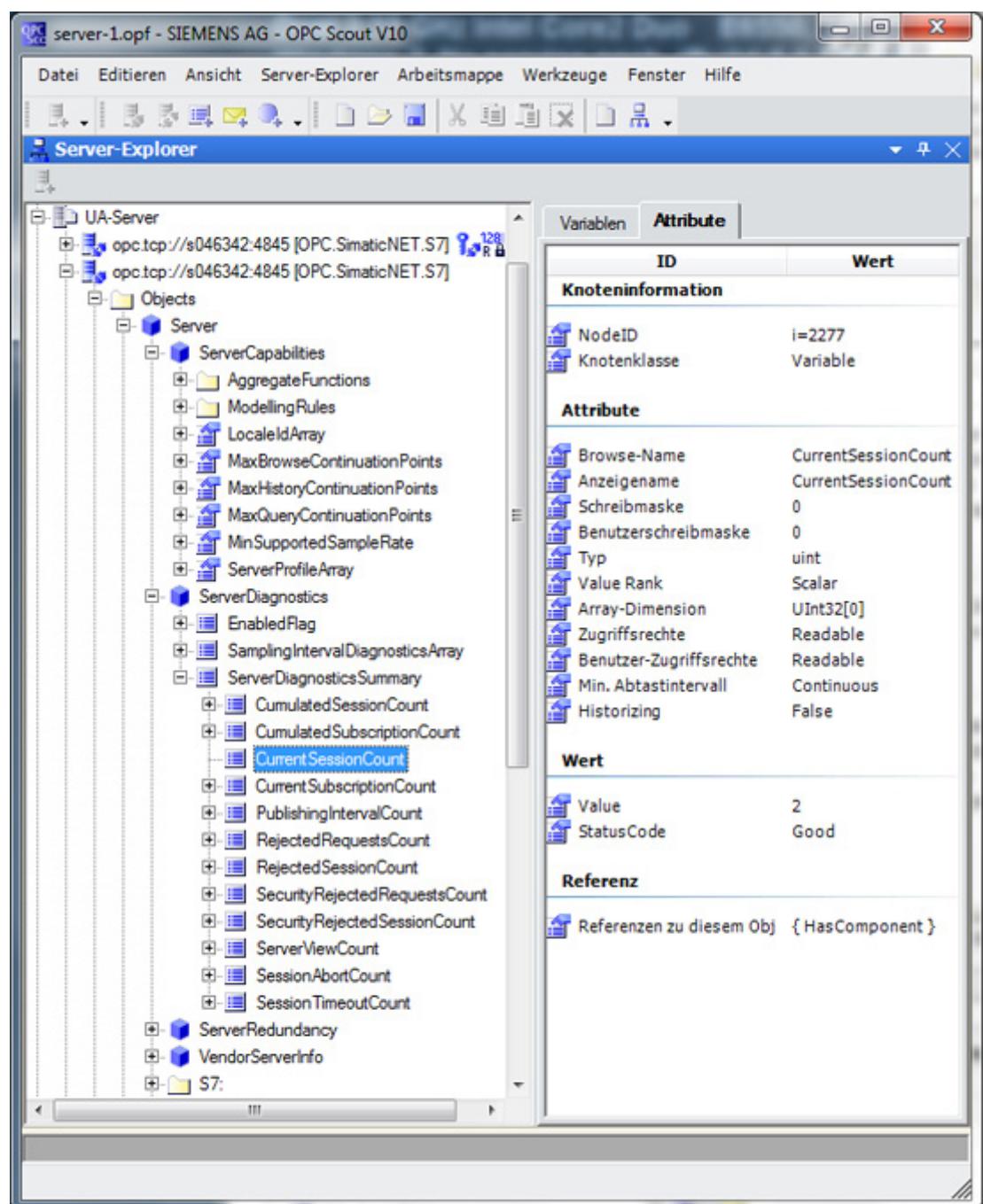


Bild 2-76 OPC-UA-Server-Diagnose-Nodes

## **2.15 Blockorientierte Dienste mit der OPC-Schnittstelle**

### **Einleitung**

Dieses Kapitel beschreibt die Verwendung blockorientierter Dienste mit OPC. Sie erfahren, wie Datenpuffer auf OPC-Variablen abgebildet werden. Außerdem lesen Sie, welche Besonderheiten bei der Verwendung von TCP/IP native im Zusammenhang mit blockorientierten Diensten zu berücksichtigen sind.

### **2.15.1 Blockdienste verwenden**

Für die Übertragung von großen Datenpaketen bieten die S7-Kommunikation und die S5-kompatible Kommunikation über Industrial Ethernet Blockdienste an. Dabei werden Datenpakete zwischen den Kommunikationspartnern versendet. Die Übermittlung der Daten belastet das Netz nur dann, wenn ein Partner explizit einen Sendeauftrag absetzt.

Mit dem OPC-Server für SIMATIC NET können Sie die Datenblöcke strukturieren. So können einzelne Teile des Datenpaketes OPC-Items zugeordnet werden.

---

#### **Hinweis**

Die S7-Blockdienste sind derzeit nur für Geräte der Baureihe S7-400, M7 und PC-Stationen verfügbar. Die S5-Blockdienste sind für nahezu alle Geräte der Baureihe S5, S7 und PC-Stationen verfügbar. Beachten Sie die Hinweise über die Kommunikationspartner bezüglich der zukünftigen Unterstützung der Blockdienste in der Liesmich-Datei auf dem Hauptverzeichnis der SIMATIC NET CD.

---

### **2.15.2 Eigenschaften blockorientierter Kommunikation**

#### **Unterstützte Dienste**

Bei blockorientierten Diensten werden Datenpuffer von einem Sender über das Kommunikationssystem an einen Empfänger gesendet.

Folgende Protokolle unterstützen blockorientierte Dienste:

- S7-Kommunikation (BSEND/BRECEIVE)
- Offene Kommunikationsdienste (SEND/RECEIVE) über Ethernet
- Offene Kommunikationsdienste (SEND/RECEIVE) über PROFIBUS (SDA, SDN/Indication)

Kennzeichnend für die Blockorientierten Dienste ist, dass nur dann Daten übertragen werden, wenn der Sender den Übertragungsvorgang anstößt. Der Empfänger kann die Kommunikation nicht auslösen.

## Aktionen von Sender und Empfänger beim Austausch von Datenpuffern

### Sender

- Stellt einen Sendepuffer mit den Inhalten zusammen
- Sendet den Puffer auf einer Verbindung an einen Kommunikationspartner ab
- Bekommt eine Quittung über das Ergebnis der Datenübertragung

### Empfänger

- Stellt einen Empfangspuffer für eine Verbindung bereit
- Wird benachrichtigt, wenn ein Partner ihm einen Datenpuffer zusendet
- Sendet eine Quittung an den Sender
- Wertet die empfangenen Daten aus

## 2.15.3 Abbildung von Datenpuffern auf OPC-Variablen

### Unterschiede zwischen Sende- und Empfangs-Items

Die OPC Data Access-Schnittstelle kennt nur Prozessvariablen. Damit die Vorteile der blockorientierten Dienste auch mit OPC genutzt werden können, muß eine Abbildung auf OPC-Items erfolgen:

#### Sende-Item (S7-Kommunikation: BSEND, Offene Kommunikation: Send)

- Ein OPC-Item repräsentiert einen Sendepuffer oder einen Teilbereich eines Sendepuffers.
- Wenn das OPC-Item geschrieben wird (synchron/asynchron), wird ein Schreibauftrag auf dem Netz ausgelöst.
- Werden mehrere Items, die einen Teilbereich eines Puffers repräsentieren, in einer Mengenoperation auf einmal geschrieben, so wird erst der gesamte Sendepuffer aus allen Teilen gebildet und dann gesendet.
- Der Lesezugriff liefert die zuletzt gesendeten Daten aus dem Sendepuffer. Wurde noch nicht gesendet, ist das Item unter den genannten Umständen lesbar, hat aber die Qualität BAD.

#### Empfangs-Item (S7-Kommunikation: BRCV, Offene Kommunikation: RECEIVE)

- Ein OPC-Item repräsentiert einen Empfangspuffer.
- Wenn das OPC-Item vom Gerät gelesen wird (synchron/asynchron), wird die Kommunikationsbaugruppe empfangsbereit. Dieser Zustand bleibt so lange bestehen, bis ein Datenpaket empfangen wurde oder der verbindungspezifische Timeout abgelaufen ist. Wenn kein Datenpaket während der Timeout-Zeit empfangen wurde, ist die Qualität des OPC-Items BAD.

- Wenn das OPC-Item beobachtet wird (Aktives Empfangs-Item in aktiver Gruppe), wird dauerhaft ein Empfangspuffer auf der Kommunikationsbaugruppe eingerichtet. Wird ein Datenpaket empfangen, so signalisiert der OnDataChange-Rückruf dies der Anwendung, sofern sich die Daten von den zuvor empfangenen Daten unterscheiden.
- Empfangs-Items sind nicht schreibbar.

## 2.15.4 Anwendung der blockorientierten Dienste

### Handhabung von Sende- und Empfangs-Items

Sende-Items sollten nur beschrieben werden, das Lesen oder Beobachten der Sende-Items liefert nur die zuvor geschriebenen Daten.

Empfangs-Items sollten beobachtet werden, d. h. das Empfangs-Item sollte als aktives Item in einer aktiven Gruppe vorhanden sein. Damit wird - unabhängig von der Implementierung einer Rückruf-Funktion - der Cache des OPC-Servers mit jedem Eintreffen eines Datenpaketes aktualisiert.

### Lesen aus dem Cache und direkt vom Gerät

Leseaufträge - synchron wie asynchron - sollten auf dem Cache ausgeführt werden. Der Cache enthält den zuletzt empfangenen Datenblock, wenn das Empfangs-Item beobachtet wird. Durch Verwendung des Cache ist auch sichergestellt, dass mehrere Clients, die zum gleichen Zeitpunkt das Item lesen, den gleichen Wert erhalten.

Leseaufträge an ein Device (direkt an das Gerät) sind nicht sinnvoll, da nur während des Timeout-Intervalls der Verbindung (üblicherweise einige Sekunden) ein Empfangspuffer bereitsteht. Ein Sender muss innerhalb dieses Zeitfensters senden, damit der Datenpuffer vom Empfänger angenommen wird.

### Update Rate

Es sollte darauf geachtet werden, dass der OPC-Server die Empfangspuffer schneller abholt, als sie vom Sender abgeschickt werden. Die Rate, mit der die Empfangspuffer abgeholt werden, wird durch den gruppenspezifischen OPC-Parameter *Update Rate* festgelegt.

Je nach Protokoll werden ansonsten empfangene Datenblöcke überschrieben, ohne dass der OPC-Client dies erfährt (bei S7, SEND/RECEIVE mit ISO und RFC 1006) oder Datenpakete stauen sich auf, so dass der Empfänger veraltete Daten erhält (SEND/RECEIVE mit TCP/IP native).

## 2.15.5 Besonderheiten der blockorientierten Dienste über TCP/IP native

### Bedeutung der Socket-Schnittstelle

Das SEND/RECEIVE-Protokoll für offene Kommunikation über TCP/IP native basiert auf den Socket-Diensten von Windows. Es werden dabei keine expliziten Puffer übermittelt, sondern an der Socket-Schnittstelle wird ein kontinuierlicher Datenstrom übertragen. Das Protokoll verwendet dabei sowohl auf Sende- als auch auf Empfangsseite einen Zwischenspeicher, der die ein- bzw. ausgehenden Daten nach dem FIFO-Prinzip (First-In First-out) überträgt.

### TCP/IP native und OPC-Variablen

Die oben beschriebenen Protokolleigenschaften wirken sich auch auf die Anwendung mit OPC-Variablen aus:

- Auch wenn der Partner keine Daten übernimmt, werden Sendaufträge so lange positiv quittiert, bis der sendeseitige Speicher überläuft. Verhindern Sie einen Speicherüberlauf anwenderprogrammseitig.
- Der Zwischenspeicher zum Datenempfang steht unabhängig von der Beobachtung einer Variablen oder dem Absetzen eines Leseauftrags immer bereit.
- Wenn auf Seiten des Empfängers die Daten nicht schnell genug abgeholt werden, gibt es einen *Datenstau* im Empfangspuffer. Das führt dazu, dass der Empfänger nicht die zuletzt gelesenen Daten übernimmt, sondern veraltete Daten. Verhindern Sie einen Datenstau anwenderprogrammseitig.

## 2.16 Zugriffsrechte von OPC-Variablen einschränken

### Zugriffsrechte in STEP 7 bzw. NCM festlegen

In STEP 7 bzw. NCM können Sie im Eigenschaftendialog des OPC-Servers Zugriffsrechte aktivieren. Die Zugriffsrechte der Variablen werden durch das Aktivieren des entsprechenden Kontrollkästchens standardmäßig auf Read/Write (RW) gesetzt und können bei Bedarf auf Read (R), Write (W) oder None geändert werden.

Im Dialogfeld Item-spezifische-Zugriffsrechte können Sie Zugriffsrechte definieren, die von der Festlegung für alle Items abweichen. Die dort eingetragenen Rechte überschreiben dann die Default-Zugriffsrechte.

### OPC-Item-spezifische Zugriffsrechte vergeben

Tragen Sie hier für eine oder mehrere OPC-Items Zugriffsrechte ein. Die für die übrigen OPC-Items definierten Default-Rechte werden dadurch nicht berührt. Die Syntax ist wie folgt definiert:

```
<OPCItem>=<Rechte>
```

**<OPCItem>**

Angabe eines oder mehrerer OPC-Items entsprechend der in der OPC-Dokumentation definierten Syntax. Die Verwendung von Aliasnamen ist möglich. Folgende Platzhalter können verwendet werden:

- \* eine beliebige Anzahl Zeichen
- ? genau ein Zeichen

**<Rechte>**

<i>RW</i>	Schreib- und Lesezugriff
<i>R</i>	nur Lesezugriff
<i>W</i>	nur Schreibzugriff
<i>NONE</i>	weder Schreib- noch Lesezugriff

**Regeln bei der Auswertung**

Die hier spezifisch vergebenen Rechte haben Vorrang vor den Default-Rechten, die Sie unter dem Eingabebereich Zugriffsschutz vergeben haben. Die hier spezifisch vergebenen Rechte werden in der hier eingegebenen Reihenfolge ausgewertet. Bei mehrfacher Zuweisung zu einem OPC-Item ist immer die letzte Zuweisung gültig.

**Beispiel:**

```
DP:[CP 5611]Slave040_AB*=RW
DP:[CP 5611]Slave040_AB1=R
DP:[CP 5611]Slave040_AB2=W
DP:[CP 5611]Slave040_AB1=W
DP:[CP 5611]Slave040_AB1*=R
```

Auf Grund dieses Eintrages sind u.a. folgende Zugriffsrechte wirksam:

```
DP:[CP 5611]Slave040_AB2=W
DP:[CP 5611]Slave040_AB1=R
```

**Hinweis**

Führende Nullen in der Syntax der Zugriffsrechtevergabe werden grundsätzlich ignoriert. Verwenden sie deshalb keine Platzhalterzeichen ? oder \* an Stellen, wo auch eine führende Null stehen kann!

**Beispiel:**

Die Definition *Slave0?9M06\_AB1=R* muss durch folgende beide Definitionen ersetzt werden:  
*Slave?9M06\_AB1=R*  
*Slave9M06\_AB1=R*

---

### **Hinweis**

Die hier beschriebene Einschränkung für Zugriffsrechte ist für OPC-UA-Server nicht gültig.

---



# OPC Alarms & Events-Server für SIMATIC NET

## 3.1 Event-Server für S7-Kommunikation

### 3.1.1 Funktionsprinzip und Alarmkategorien

#### Einleitung

Dieses Kapitel beschreibt, welche Ereignisse vom OPC-Server geliefert werden und welche Zusatzinformationen als Attribute zur Verfügung stehen.

#### A&E-Server für SIMATIC NET

Der OPC-Server für SIMATIC NET kann als "Simple Event-Server" oder als "Conditional Alarm- und Event-Server" genutzt werden. Ein Simple Event-Server verfügt in einem Bedien- und Beobachtungssystem über keine Projektierungsinformationen bezüglich der Alarm- und Ereignisbehandlung.

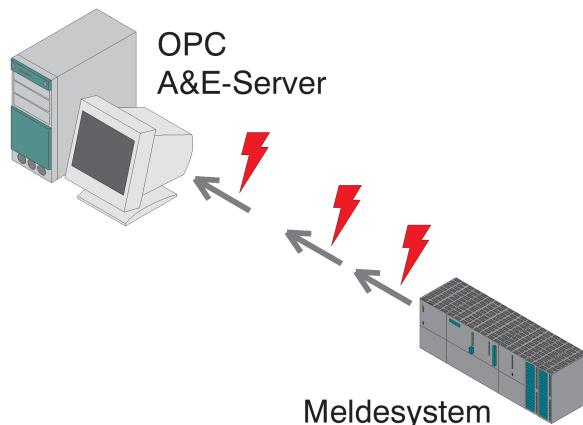


Bild 3-1 Prinzip der Alarms & Events-Kommunikation zwischen Client und A&E-Server

'Condition-related Events' können Unterbedingungen erhalten. Es ist möglich, für einen Alarm auch mehrere Unterbedingungen zu definieren. Ein solches Event tritt auf, wenn eine der Unterbedingungen erfüllt ist.

Darüber hinaus bieten condition-related Events die Möglichkeit, die Auswertung der Bedingungen durch den Server zu aktivieren oder zu deaktivieren, sowie eine Quittierungsmöglichkeit.

## Einbindung des Event-Server

Ein Simple Event-Server gibt nur einfache Meldungen der unterlagerten Komponenten weiter. Ein übergeordneter Alarm/Event-Management-Server verarbeitet unter Beachtung von Projektierungsinformationen die Meldungen der unterlagerten Simple Server.

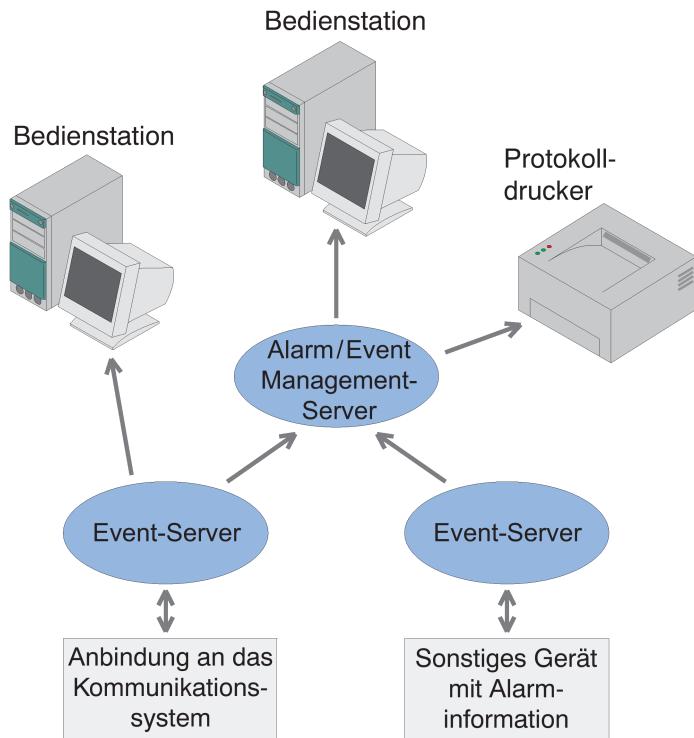


Bild 3-2 Einbindung des Event-Server

## S7-Protokoll

Das S7-Protokoll bietet den Protokollmechanismus "Programmierte Meldungen (ALARM)" zur Übertragung von Ereignissen an.

Der Meldungstyp wird vom Simple Event-Server verwendet.

## Alarmkategorie

Für die S7-Kommunikation werden die folgenden *Alarms&Events*-Kategorien bereitgestellt:

Event-Kategorie Wert	Beschreibung
EVENTCATEGORY_S7SIMPLE 2	Simple Events für folgende S7-Alarne: NOTIFY ALARM ALARM_8 ALARM_8P
EVENTCATEGORY_DIAGNOSIS 12	Enthalten die Einträge im Diagnosepuffer der Baugruppe (Simple Events).
EVENTCATEGORY_S7CONDITION 13	Condition Events für folgende S7-Alarne: NOTIFY ALARM ALARM_8 ALARM_8P ALARM_S ALARM_SQ
EVENTCATEGORY_STATEPATH 14	Alarne für die Anzeige einer unterbrochenen Verbindung zu einem Gerät (Condition related Events).
EVENTCATEGORY_CAT_LEVEL 40	Levelalarm Alarm signalisiert Überschreitungen von Min/Max-Werten
EVENTCATEGORY_CAT_DEVIATION 41	Toleranzalarm Alarm signalisiert Abweichungen von Toleranzen
EVENTCATEGORY_CAT_ROC 42	RateOfChange-Alarm Alarm signalisiert Wechselhäufigkeiten
EVENTCATEGORY_CAT_OFFNORMAL 43	OffNormal-Alarm Alarm signalisiert Abweichungen von einem Normalzustand Default für alle S7-Alarne SFB, SFC, SCAN
EVENTCATEGORY_CAT_TRIP 44	Trip-Alarm Alarm signalisiert das Auslösen einer Abschalt-sicherung
EVENTCATEGORY_CAT_COS 45	ChangeOfState-Alarm Alarm signalisiert eine Zustandsänderung
EVENTCATEGORY_CAT_DEVICEFAILURE 46	Gerätefehler Alarm signalisiert einen Gerätefehler
EVENTCATEGORY_CAT_SYSTEMFAILURE 47	Systemfehler Alarm signalisiert einen Systemfehler
EVENTCATEGORY_CAT_SYSTEMMESSAGE 60	Systemnachricht (Simple-Event)

**Hinweis**

Die Event-Kategorien 2 bis 14 werden für Projektierungen des OPC-Server < 8.0 und über die Meldetextdatei verwendet. Für OPC-Server >= 8.0 gelten die Kategorien 40 bis 60.

Tabelle 3-1 Alarme und Scans werden der Alarmkategorie 43 (CAT\_OFFNORMAL) zugeordnet oder entsprechend der eingestellten Step 7-Meldeklasse gemäß folgender Tabelle:

STEP 7 Meldeklasse	OPC-Alarmkategorie
Keine Meldung	Meldung wird unterdrückt (durch Anwender einstellbar, z.B. bei überzähligen Alarm_8p Alarmbits).
Unspezifizierte oder unbekannte Meldeklasse	Alarmkategorie 43 (EVENTCATEGORY_CAT_OFFNORMAL) (Default)
Alarm - oben Alarm – unten Warnung – oben Warnung – unten	Alarmkategorie 40 (EVENTCATEGORY_CAT_LEVEL) Condition "HIHI" Condition "LOLO" Condition "HI" Condition "LO"
Toleranz – oben Toleranz – unten	Alarmkategorie 41 (EVENTCATEGORY_CAT_DEVIATION) Condition "HI" Condition "LO"
AS-Leittechnik Meldung - Störung AS-Leittechnik Meldung - Fehler OS-Leittechnik Meldung - Störung Vorbeugende Wartung - allgemein Prozeßmeldung - mit Quittierung Betriebsmeldung - ohne Quittierung Bedienanforderung - allgemein Bedienmeldung - allgemein Statusmeldung - AS Statusmeldung – OS	Alarmkategorie 43 (EVENTCATEGORY_CAT_OFFNORMAL) Condition "CFN"

### 3.1.2 Parameter für Ereignisse

#### Parameter für alle Events

Mit dem Aufruf der Rückruf-Funktion *OnEvent* des Client erhält der Client eine Liste von Ereignissen. Welche Parameter der OPC-Event-Server liefert, hängt von der Art des Events ab. Die folgenden Parameter sind für alle Events relevant:

Parameter	Bedeutung
szSource	Als Quelle für die Benachrichtigung wird vom OPC-Event-Server die Verbindungsinformation des meldenden S7-Geräts angegeben. Sie setzt sich wie folgt zusammen: S7:<Verbindungsname>
ftTime	Der Zeitpunkt, zu dem das Event auftrat.
szMessage	Der Name der Nachricht setzt sich aus dem Meldemechanismus und der projektierten Meldungsnummer zusammen: ALARM <Meldungsnummer> Beispiele: ALARM55 (siehe auch Hinweis unten)
dwEventType	Der OPC-Event-Server unterstützt die folgenden Events: OPC_SIMPLE_EVENT Diese Konstante hat den Wert 0x0001. OPC_CONDITION_EVENT Diese Konstante hat den Wert 0x0004.
dwEventCategory	Als Ereigniskategorie sind folgende Werte möglich: Für OPC-Server < 8.0: EVENTCATEGORY_S7SIMPLE (2) EVENTCATEGORY_S7T0 (11) EVENTCATEGORY_DIAGNOSIS (12) EVENTCATEGORY_S7CONDITION (13) EVENTCATEGORY_STATEPATH (14) Für OPC-Server >= 8.0: 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	Für die Schwere des Meldungsgewichts wird ein Vorgabewert zurückgeliefert. Dieser Vorgabewert kann für einzelne Meldungsnummern eines Kommunikationspartners in der Konfigurationsdatenbank von STEP 7 verändert werden.
dwNumEventAttrs	Die Anzahl der mit der Meldung gelieferten Attribute hängt von der Anzahl der mitgelieferten Begleitwerte ab.
pEventAttributes	Diese Struktur beinhaltet die mit dem Ereignis gelieferten Attribute. Die Attribute beinhalten auch die vom Partnergerät gelieferten Begleitwerte der Meldung.

**Hinweis****Alarmbaustein-Meldungsgewicht vor Vorgabe-Priorität für Alarmmeldungen**

Mit dem OPC Alarms & Events-Server "OPC.SimaticNetAlarms" können S7-Alarne mit Meldungsgewicht ("Severity") empfangen werden. Diese Meldungen können in STEP 7 / NetPro unter Eigenschaften S7-Verbindung des OPC-Servers projektiert werden. Beachten Sie:

Das programmierbare Alarm-Meldungsgewicht eines Bausteins ALARM, ALARM\_8P oder NOTIFY in einem S7-Programm setzt sich gegen die projektierbare Vorgabe-Priorität für Alarmmeldungen durch.

Weiterhin setzen sich in NetPro projektierte Alarm-Prioritäten für bestimmte Alarm-Nummern gegenüber der allgemeinen Vorgabe-Priorität für Alarmmeldungen und auch gegenüber programmierten Meldungsgewichten durch. ALARM\_S und ALARM\_SQ besitzen keine Severity, so dass immer die projektierten Meldungsgewichte verwendet werden.

**Parameter für condition-related Events**

Für *condition-related Events* werden zusätzlich folgende Parameter übertragen:

Parameter	Bedeutung
szConditionName	Der Name der Bedingung.
szSubconditionName	Der Name der Unterbedingung. Falls es keine Unterbedingung gibt, enthält dieser Parameter den Namen der Bedingung.
wChangeMask	Zeigt an, welche Zustandsänderung das Senden der Benachrichtigung veranlasst hat. Mögliche Werte sind z. B. OPC_CHANGE_QUALITY oder OPC_CHANGE_SUBCONDITION.
wNewState	Der Status der im Parameter szSubconditionName übermittelten Unterbedingung. Folgende Werte sind möglich: OPC_CONDITION_ACTIVE OPC_CONDITION_ENABLE OPC_CONDITION_ACKED
wQuality	Enthält eine Information über die Qualität des Wertes, auf den sich die Benachrichtigung bezieht.
bAckRequired	Zeigt an, ob die Benachrichtigung eine Quittierung erfordert.
ftActiveTime	Der Zeitpunkt, zu dem die Unterbedingung aktiviert wurde. Wenn keine Unterbedingung vorhanden ist, enthält dieser Parameter den Zeitpunkt, ab dem die Bedingung erfüllt war.
dwCookie	Vom Server vergebenes Cookie für die Benachrichtigung.
szActorID	Enthält eine anwendungsspezifische Information über die Ursache des Ereignisses.

### 3.1.3 Ereignisattribute

Zu einer Meldung oder einem Alarm liefert eine S7-Station einen Begleitwert mit. Bei EVENT\_ATTR\_S7\_DATA $n$ , EVENT\_ATTR\_S7\_DATA $n$ \_DATATYPE und EVENT\_ATTR\_S7\_DATA $n$ \_VALUE\_LEN können dies für  $n$  bis zu 10 Begleitwerte sein. Die Begleitwerte stehen über die OPC Alarms & Events-Schnittstelle in den Ereignisattributen zur Verfügung.

Programmierte Meldungen (ALARM) liefern folgende Attribute:

- 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**

Wert:	-1
Datentyp:	VT_BSTR

Attributwert: Projektierter Meldetext, sprachabhängig. Falls nicht projektiert, identisch zu EventID

### **EVENT\_ATTR\_S7\_ALARMSTATE**

Wert:	-2
Datentyp:	VT_UI4

Attributwert: Alarmzustände (EVENT\_ATTR\_S7\_ALARMSTATE)

EVENT_ATTR_S7_ALARMSTATE	Beschreibung
CHANGE_ACTIVE   ACTIVE   ACKREQUIRED	Die Werte der einzelnen #Defines (ACTIVE...) sind in den opc_ae.h Header-Dateien bestimmt. ALARM/ALARM_8/ALARM_8P/ALARM_SQ/ALARM_DQ/ SCAN gekommen
CHANGE_ACTIVE   ACTIVE   ACKED	ALARM_S/ALARM_D/NOTIFY/NOTIFY_8P gekommen (implizit quittiert) Das Flag CHANGE_ACKED darf nicht gesetzt werden.
ACTIVE   ACKREQUIRED	ALARM/ALARM_8/ALARM_8P/ALARM_SQ/ALARM_DQ/ SCAN aktiv (nur bei Refresh)
ACTIVE   ACKED	ALARM_S/ALARM_D/NOTIFY/NOTIFY_8P aktiv (implizit quittiert) (nur bei Refresh)
ACTIVE   CHANGE_ACKED   ACKED	ALARM/ALARM_8/ALARM_8P/ALARM_SQ/ALARM_DQ/ SCAN aktiv und Quittierung gekommen
ACTIVE   ACKED	ALARM/ALARM_8/ALARM_8P/ALARM_SQ/ALARM_DQ/ SCAN aktiv und quittiert (Nur bei Refresh)
CHANGE_INACTIVE   ACKREQUIRED	ALARM/ALARM_8/ALARM_8P/ALARM_SQ/ALARM_DQ/ SCAN gegangen und noch nicht quittiert
CHANGE_INACTIVE   ACKED	ALARM_S/ALARM_D/NOTIFY/NOTIFY_8P gegangen (implizit quittiert)
CHANGE_INACTIVE   ACKED	ALARM/ALARM_8/ALARM_8P/ALARM_SQ/ALARM_DQ/ SCAN gegangen und quittiert
ACKREQUIRED	ALARM/ALARM_8/ALARM_8P/ALARM_SQ/ ALARM_DQ/SCAN inaktiv und noch nicht quittiert (nur bei Refresh)

**EVENT\_ATTR\_S7\_TYPE**

Wert:	-5
Datentyp:	VT_UI4

Attributwert: EVENT\_TYPE\_CONDITION

**EVENT\_ATTR\_S7\_SEVERITY**

Wert:	-6
Datentyp:	VT_I4

Attributwert: Severity

Der Severity-Wert zeigt die Gewichtung oder Priorität einer Sub-Condition an. Der Wertebereich liegt zwischen 1 und 1000. Je höher der Wert, desto höher ist die Gewichtung.

**EVENT\_ATTR\_S7\_CATEGORYID**

Wert:	-9
Datentyp:	VT_I4

Attributwert: Category Identifier

Mögliche Werte werden oben in der Tabelle "Alarms&Event-Kategorien" im Kapitel "Funktionsprinzip und Alarmkategorien (Seite 391)" aufgelistet.

**EVENT\_ATTR\_S7\_CATEGORYDESC**

Wert:	-10
Datentyp:	VT_BSTR

Attributwert: Kategorie Beschreibung

**EVENT\_ATTR\_S7\_COMMENT**

Wert:	-13
Datentyp:	VT_BSTR

Attributwert: Projektierter Zusatztext 9, der sprachabhängig ist.

**EVENT\_ATTR\_S7\_ACTIVETIME**

Wert:	-15
Datentyp:	VT_DATE

Attributwert: Zeitpunkt des Signalwechsels von "Alarm gekommen" aus dem Zeitstempel.

**EVENT\_ATTR\_S7\_PCTIME**

Wert:	6000
Datentyp:	VT_DATE

Attributwert: Zeitpunkt, an dem der OPC-Event-Server die Meldung bekommen hat.

**EVENT\_ATTR\_S7\_S7TIME**

Wert:	6001
Datentyp:	VT_DATE

Attributwert: Zeitpunkt, zu dem die Meldung auf dem Partnergerät erzeugt wurde.

**EVENT\_ATTR\_S7\_STATE**

Wert:	6002
Datentyp:	VT_UI1 bei Condition Events VT_UI2 bei Simple Events

Attributwert: Allgemeiner Zustand, der angibt, ob eine Meldung vorhanden ist.

**ALARM Meldungen**

Das Bit	zeigt an:
0x00H (alle Bits 0)	Die Meldung ist vorhanden
Bit 0 gesetzt	Initanlauf
Bit 1 gesetzt	Overflow Signal
Bit 2 gesetzt	Overflow Instanz
Bit 3 ... 5	0, reserviert
Bit 6 gesetzt	Zusatzwerte nicht eintragbar (Größe)
Bit 7 gesetzt	Zusatzwerte nicht erreichbar

**EVENT\_ATTR\_S7\_ACK\_STATE**

Wert:	6003
Datentyp:	VT_UI2

Attributwert: Quittierungszustand des Alarms im S7-Automatisierungsgerät

Über den OPC-Event-Server für SIMATIC NET ist eine Quittierung von Meldungen möglich. Meldungen können jedoch auch von anderen Bediensystemen quittiert werden.

ALARM Meldungen:

<b>Das Bit</b>	<b>zeigt an:</b>
0	Quittierung erfolgt für "Meldung 1 gekommen"
1 ... 6	Quittierung erfolgt für "Meldung 2 bis 7 gekommen" (nur ALARM_8 / ALARM_8P)
7	Quittierung erfolgt für "Meldung 8 gekommen" (nur ALARM_8 / ALARM_8P)
8	Quittierung erfolgt für "Meldung 1 gegangen"
9 ... 14	Quittierung erfolgt für "Meldung 2 bis 7 gegangen" (nur ALARM_8 / ALARM_8P)
15	Quittierung erfolgt für "Meldung 8 gegangen" (nur ALARM_8 / ALARM_8P)

Über den OPC-Event-Server für SIMATIC NET ist eine Quittierung nur für "Meldung n gekommen" möglich. Meldungen können jedoch auch von anderen Bediensystemen quittiert werden.

### **EVENT\_ATTR\_S7\_EVENT\_STATE**

Wert:	6004
Datentyp:	VT_UI1 bei Condition-related Events VT_UI2 bei Simple Events

Attributwert: Ereigniszustand

<b>Das Bit</b>	<b>zeigt an:</b>
0	Aktueller Zustand "Meldung 1" (1 = aktiv)
1 ... 6	Aktueller Zustand "Meldung 2 bis 7" (1 = aktiv) (nur ALARM_8 / ALARM_8P)
7	Aktueller Zustand "Meldung 8" (1 = aktiv) (nur ALARM_8 / ALARM_8P)

### **EVENT\_ATTR\_S7\_NO\_DATA**

Wert:	6005
Datentyp:	VT_UI1 bei Condition Events VT_UI2 bei Simple Events

Attributwert: Anzahl der Begleitwerte. Wertebereich: 1...10

**EVENT\_ATTR\_S7\_DATAAn**

Wert:	für $n = 0$ für $n = 1$ für $n = 2$ für $n = 3$ für $n = 4$ für $n = 5$ für $n = 6$ für $n = 7$ für $n = 8$ für $n = 9$	6008 6012 6016 6020 6024 6028 6032 6036 6040 6044
Datentyp:	VT_ARRAY VT_UI1	

Attributwert: Die relevanten Bytes des Begleitwertes Nr. "n" als Feld von Bytes.

**EVENT\_ATTR\_S7\_DATAAn\_DATATYPE**

Wert:	für $n = 0$ für $n = 1$ für $n = 2$ für $n = 3$ für $n = 4$ für $n = 5$ für $n = 6$ für $n = 7$ für $n = 8$ für $n = 9$	6006 6010 6014 6018 6022 6026 6030 6034 6038 6042
Datentyp:	VT_UI2:	bei Simple Events
	VT_UI1:	bei Condition-related Events

Attributwert: Datentyp des Begleitwertes Nr. "n".

Parameterwert	Beschreibung
S7_DATATYPE_ERROR	Fehler (0x0)
S7_DATATYPE_BOOLEAN	Boolean (0x03)
S7_DATATYPE_INTEGER	Integer (0x05)
S7_DATATYPE_UNSIGNED	Integer (0x06)
S7_DATATYPE_FLOAT	Float (0x07)
S7_DATATYPE_OCTET_STRING	String (0x09)
S7_DATATYPE_BITSTRING	Bit-String (0x04) Hinweis: Längenangabe in Byte statt in Bit
S7_DATATYPE_DATE	Datum (0x30) Hinweis: Tage seit 01.01.1990

Parameterwert	Beschreibung
S7_DATATYPE_TIME_OF_DAY	Zeit (0x31) Hinweis: ms seit Tagesbeginn
S7_DATATYPE_TIME	Zeit (0x32) Hinweis: ms
S7_DATATYPE_S5TIME	Zeit (0x33) Hinweis: BCD codiert
S7_DATATYPE_DATE_AND_TIME	Datum und Uhrzeit (0x34)

**EVENT\_ATTR\_S7\_DATA<sub>n</sub>\_VALUE\_LEN**

Wert:	für $n = 0$	6007
	für $n = 1$	6011
	für $n = 2$	6015
	für $n = 3$	6019
	für $n = 4$	6023
	für $n = 5$	6027
	für $n = 6$	6031
	für $n = 7$	6035
	für $n = 8$	6039
	für $n = 9$	6043
Datentyp:	VT_UI2: VT_UI2:	bei Simple Events bei Condition-related Events

Attributwert: Anzahl der relevanten Bytes des Begleitwertes Nr. "n".

**EVENT\_ATTR\_S7\_EVENT\_EVENTID**

Wert:	6046
Datentyp:	VT_UI4 (nur bei Condition-related Events)

Attributwert: Alarmnummer

**EVENT\_ATTR\_S7\_EVENT\_SUBEVENTID**

Wert:	6047
Datentyp:	VT_UI1 (nur bei Condition-related Events)

Attributwert: SubEventID (Wertebereich 1 ... 8) bei Meldungen vom Typ ALARM\_8 / ALARM\_8P,  
sonst 1.

**EVENT\_ATTR\_S7\_CONNECTION**

Wert:	6050
Datentyp:	VT_BSTR

Attributwert: S7-Verbindungsname

**EVENT\_ATTR\_S7 AREAS**

Wert:	6051
Datentyp:	VT_ARRAY of VT_BSTR

Attributwert: Areas, unter denen das Diagnoseereignis aufgezählt wird.

**EVENT\_ATTR\_S7\_INFOTEXT**

Wert:	6060
Datentyp:	VT_BSTR

Attributwert: Projektierter Infotext, der sprachabhängig ist.

**EVENT\_ATTR\_S7\_TEXT1**

Wert:	6061
Datentyp:	VT_BSTR

Attributwert: Projektierter Zusatztext 1, der sprachabhängig ist.

**EVENT\_ATTR\_S7\_TEXT2**

Wert:	6062
Datentyp:	VT_BSTR

Attributwert: Projektierter Zusatztext 2, der sprachabhängig ist.

**EVENT\_ATTR\_S7\_TEXT3**

Wert:	6063
Datentyp:	VT_BSTR

Attributwert: Projektierter Zusatztext 3, der sprachabhängig ist.

**EVENT\_ATTR\_S7\_TEXT4**

Wert:	6064
Datentyp:	VT_BSTR

Attributwert: Projektierter Zusatztext 4, der sprachabhängig ist.

**EVENT\_ATTR\_S7\_TEXT5**

Wert:	6065
Datentyp:	VT_BSTR

Attributwert: Projektierter Zusatztext 5, der sprachabhängig ist.

**EVENT\_ATTR\_S7\_TEXT6**

Wert:	6066
Datentyp:	VT_BSTR

Attributwert: Projektierter Zusatztext 6, der sprachabhängig ist.

**EVENT\_ATTR\_S7\_TEXT7**

Wert:	6067
Datentyp:	VT_BSTR

Attributwert: Projektierter Zusatztext 7, der sprachabhängig ist.

**EVENT\_ATTR\_S7\_TEXT8**

Wert:	6068
Datentyp:	VT_BSTR

Attributwert: Projektierter Zusatztext 8, der sprachabhängig ist.

**EVENT\_ATTR\_S7\_TEXT9**

Wert:	6069
Datentyp:	VT_BSTR

Attributwert: Projektierter Zusatztext 9, der sprachabhängig ist.

## Bedeutung der verschiedenen Zeitstempel

Attribute	Eintreffen neuer Meldungen vom S7-Gerät:	Refresh
Parameter TimeStamp	<p>Abhängig von der Einstellung der Projektierung der Zeitstempelherkunft:</p> <p>1) Eintreffen von Flankenwechsel: S7-Zeitpunkt der jeweiligen Flanke des Signals</p> <p>2) Eintreffen der Quittierungsmeldung: S7-Zeitpunkt der Quittierung</p> <p>Oder</p> <p>1) Eintreffen von Flankenwechsel: S7-Zeitpunkt der jeweiligen Flanke des Signals +- Offset</p> <p>2) Eintreffen der Quittierungsmeldung: S7-Zeitpunkt der Quittierung +- Offset</p> <p>Oder</p> <p>1) Eintreffen von Flankenwechsel: PC-Zeitpunkt der Benachrichtigung</p> <p>2) Eintreffen der Quittierungsmeldung: PC-Zeitpunkt der Benachrichtigung</p>	<p>Abhängig von der Einstellung der Projektierung der Zeitstempelherkunft: letzter S7-Zeitpunkt der jeweiligen Flanke des Signals oder der Quittierung</p> <p>Oder letzter S7-Zeitpunkt der jeweiligen Flanke des Signals oder der Quittierung +- Offset</p> <p>Oder PC-Zeitpunkt der letzten Benachrichtigung über Flankenwechsel oder Quittierung</p>
EVENT_ATTR_S7_S7TIME	<p>1) Eintreffen von Flankenwechsel: S7-Zeitpunkt der jeweiligen Flanke des Signals</p> <p>2) Eintreffen der Quittierungsmeldung: S7-Zeitpunkt der Quittierung</p>	letzter S7-Zeitpunkt der jeweiligen Flanke des Signals oder der Quittierung
EVENT_ATTR_S7_PCTIME	<p>1) Eintreffen von Flankenwechsel: PC-Zeitpunkt der Benachrichtigung</p> <p>2) Eintreffen der Quittierungsmeldung: PC-Zeitpunkt der Benachrichtigung</p>	PC-Zeitpunkt der letzten Benachrichtigung über Flankenwechsel oder Quittierung

Beim ersten Verbindungsaufbau oder einer Verbindungsunterbrechnung liefert der S7-Meldungsupdate bei den Signaltypen ALARM/ALARM\_8/ALARM\_8P/ALARM\_SQ/ALARM\_DQ/SCAN keinen S7-Zeitstempel. Das Attribut ATTR\_S7\_S7TIME enthält dann den Zeitstempel 01.01.1990 (ältestmögliche S7-Zeit), der Timestamp wird Abhängig von der Einstellung der Projektierung der Zeitstempelherkunft andernfalls aus dem Zeitstempel 01.01.1990 oder aus der PC-Zeit gebildet.

### Hinweis

Beachten Sie, dass nicht alle Zustandswechsel eines Alarms an der OPC-Schnittstelle durch einen Event weitergereicht werden. Es kann vorkommen, dass bei schneller Änderung des Alarmzustands, nicht alle Zustandswechsel mit einem Event mitgeteilt werden, sondern nur der letzte nun aktuelle Zustand gesendet wird. Dies ist abhängig von OPC-Parametern, wie z.B. "dwBufferTime" sowie der Projektierung und der Rechnerleistung.

### 3.1.4 Attribute für Einträge im Diagnosepuffer der Baugruppe

#### EVENT\_ATTR\_DIAGNOSIS\_MSGTEXT

Wert:	-1
Datentyp:	VT_BSTR

Attributwert: Projektierter Meldetext, sprachabhängig. Falls nicht projektiert, identisch zu EventIDVor-zugsweise verwendet jedoch eine SIMOTION Umgebung ihren eigenen Meldetextserver.

#### EVENT\_ATTR\_DIAGNOSIS\_TYPE

Wert:	-5
Datentyp:	VT_I4

Attributwert: EVENT\_TYPE\_SIMPLE

#### EVENT\_ATTR\_DIAGNOSIS\_SEVERITY

Wert:	-6
Datentyp:	VT_I4

Attributwert: Severity

#### EVENT\_ATTR\_DIAGNOSIS\_CATEGORYID

Wert:	-9
Datentyp:	VT_I4

Attributwert: Kategorie Identifier

#### EVENT\_ATTR\_DIAGNOSIS\_CATEGORYDESC

Wert:	-10
Datentyp:	VT_I4

Attributwert: Kategorie Beschreibung

#### EVENT\_ATTR\_DIAGNOSIS\_S7\_PCTIME

Wert:	6001
Datentyp:	VT_DATE

Attributwert: Zeitstempel des SIMATIC NET S7-OPC-Servers

**EVENT\_ATTR\_DIAGNOSIS\_S7\_DIAGNOSISID**

Wert:	6048
Datentyp:	VT_I4

Attributwert: Diagnoseereignisnummer, analog EventID

**EVENT\_ATTR\_DIAGNOSIS\_S7\_DIAGNOSISDATA**

Wert:	6049
Datentyp:	VT_ARRAY oder VT_UI1

Attributwert: Informationen zum Diagnoseereignis bzw. zu dessen Wirkung, Datenlänge abhängig von der Ereignisklasse.

**Hinweis**

Eine detaillierte Beschreibung zu SFC 52 (S7DiagnosisData) finden Sie im Dokument "STEP 7 System- und Standardfunktionen für S7-300/400".

**EVENT\_ATTR\_DIAGNOSIS\_S7\_CONNECTION**

Wert:	6050
Datentyp:	VT_BSTR

Attributwert: S7-Verbindungsname

**EVENT\_ATTR\_DIAGNOSIS\_S7.Areas**

Wert:	6051
Datentyp:	VT_ARRAY VT_BSTR

Attributwert: Areas, unter denen das Diagnoseereignis aufgezählt wird. Gegenwärtig ist höchstens eine Area verfügbar. Default: leer

Oder

Falls irgendwelche Projektierungsinformationen über den Alarm bestehen:  
connections\<verbindungsname>

Oder

Projektierte Area

**EVENT\_ATTR\_DIAGNOSIS\_S7\_INFOTEXT**

Wert:	6060
Datentyp:	VT_BSTR

Attributwert: Projektierter Infotext, sprachabhängig. Falls nicht projektiert, leer.

**EVENT\_ATTR\_DIAGNOSIS\_S7\_TEXT1**

Wert:	6061
Datentyp:	VT_BSTR

Attributwert: Projektierter Zusatztext 1 (ggf. identisch mit Source), sprachabhängig. Falls nicht projektiert, leer.

**EVENT\_ATTR\_DIAGNOSIS\_S7\_TEXT2**

Wert:	6062
Datentyp:	VT_BSTR

Attributwert: Projektierter Zusatztext 2 (ggf. verwendet als Area), sprachabhängig. Falls nicht projektiert, leer.

**EVENT\_ATTR\_DIAGNOSIS\_S7\_TEXT3**

Wert:	6063
Datentyp:	VT_BSTR

Attributwert: Projektierter Zusatztext 3, sprachabhängig. Falls nicht projektiert, leer.

**EVENT\_ATTR\_DIAGNOSIS\_S7\_TEXT4**

Wert:	6064
Datentyp:	VT_BSTR

Attributwert: Projektierter Zusatztext 4, sprachabhängig. Falls nicht projektiert, leer.

**EVENT\_ATTR\_DIAGNOSIS\_S7\_TEXT5**

Wert:	6065
Datentyp:	VT_BSTR

Attributwert: Projektierter Zusatztext 5, sprachabhängig. Falls nicht projektiert, leer.

**EVENT\_ATTR\_DIAGNOSIS\_S7\_TEXT6**

Wert:	6066
Datentyp:	VT_BSTR

Attributwert: Projektierter Zusatztext 6, sprachabhängig. Falls nicht projektiert, leer.

**EVENT\_ATTR\_DIAGNOSIS\_S7\_TEXT7**

Wert:	6067
Datentyp:	VT_BSTR

Attributwert: Projektierter Zusatztext 7, sprachabhängig. Falls nicht projektiert, leer.

**EVENT\_ATTR\_DIAGNOSIS\_S7\_TEXT8**

Wert:	6068
Datentyp:	VT_BSTR

Attributwert: Projektierter Zusatztext 8, sprachabhängig. Falls nicht projektiert, leer.

**EVENT\_ATTR\_DIAGNOSIS\_S7\_TEXT9**

Wert:	6069
Datentyp:	VT_BSTR

Attributwert: Projektierter Zusatztext 9, sprachabhängig. Falls nicht projektiert, leer.

**EVENT\_ATTR\_DIAGNOSIS\_S7\_S7TIME**

Wert:	6100
Datentyp:	VT_DATE

Attributwert: Zeitstempel

**Bedeutung der verschiedenen Zeitstempel**

Attribute	Eintreffen neuer Diagnoseereignisse vom S7-Gerät:
Parameter TimeStamp	Abhängig von der Einstellung des EventTimeBias: S7-Zeitpunkt des Diagnoseereignisses Oder S7-Zeitpunkt des Diagnoseereignisses +- Offset Oder PC-Zeitpunkt des Eintreffens des Diagnoseereignisses.
EVENT_ATTR_DIAGNOSIS_S7TIME	S7-Zeitpunkt des Diagnoseereignisses
EVENT_ATTR_DIAGNOSIS_PCTIME	PC-Zeitpunkt des Eintreffens des Diagnoseereignisses

**Siehe auch**

Attribute für Alarme, die eine unterbrochene Verbindung anzeigen (Seite 411)

**3.1.5 Attribute für Alarme, die eine unterbrochene Verbindung anzeigen****EVENT\_ATTR\_STATEPATH\_MSGTEXT**

Wert:	-1
Datentyp:	VT_BSTR

Attributwert: Projektierter Meldetext, sprachabhängig. Falls nicht projektiert, identisch zu EventID.

**EVENT\_ATTR\_STATEPATH\_ALARMSTATE**

Wert:	-2
Datentyp:	VT_UI4

Attributwert: Alarmzustände (EVENT\_ATTR\_STATEPATH\_ALARMSTATE)

EVENT_ATTR_STATEPATH_ALARMSTATE	Beschreibung
CHANGE_ACTIVE   ACTIVE	Die Werte der einzelnen #Defines (ACTIVE...) sind in den opc_ae.h Header-Dateien bestimmt. STATEPATH gekommen (nicht bei "Refresh")
ACTIVE	STATEPATH aktiv
CHANGE_INACTIVE   INACTIVE	STATEPATH gegangen (nicht bei "Refresh")

**EVENT\_ATTR\_STATEPATH\_TYPE**

Wert:	-5
Datentyp:	VT_I4

Attributwert: EVENT\_TYPE\_CONDITION

**EVENT\_ATTR\_STATEPATH\_SEVERITY**

Wert:	-6
Datentyp:	VT_I4

Attributwert: Severity

**EVENT\_ATTR\_STATEPATH\_CATEGORYID**

Wert:	-9
Datentyp:	VT_I4

Attributwert: Category Identifier

**EVENT\_ATTR\_STATEPATH\_CATEGORYDESC**

Wert:	-10
Datentyp:	VT_BSTR

Attributwert: Kategorie Beschreibung

**EVENT\_ATTR\_STATEPATH\_COMMENT**

Wert:	-13
Datentyp:	VT_BSTR

Attributwert: Kommentar

**EVENT\_ATTR\_STATEPATH\_ACTIVETIME**

Wert:	-15
Datentyp:	VT_DATE

Attributwert: Zeitpunkt des Signalwechsels auf "Alarm gekommen" aus dem Zeitstempel.

**EVENT\_ATTR\_STATEPATH\_CONNECTION**

Wert:	6050
Datentyp:	VT_BSTR

Attributwert: S7-Verbindungsname

**EVENT\_ATTR\_STATEPATH.Areas**

Wert:	6051
Datentyp:	VT_ARRAY of VT_BSTR

Attributwert: Areas, unter denen der Statepathalarm aufgezählt wird.

### 3.1.6 Projektierung von Meldetexten, Source und Area

#### Welche Meldungen werden unterstützt?

Der OPC Alarm & Event-Server unterstützt folgende Alarne:

- Symbolbezogene Meldungen (SCANs)  
Ermöglichen asynchron zum SPS-Anwenderprogramm die Überwachung von Bits in den Bereichen E, A, M und DB der CPU.
- Bausteinbezogene Meldungen (Alarm-SFB, Alarm-SFC)  
Quittierbare Alarm-SFBs sind: ALARM (SFB 33), ALARM\_8 (SFB 34) und ALARM\_8P (SFB 35)  
Nicht quittierbar sind die SDBs: NOTIFY (SFB 36) und NOTIFY\_8P (SFB 31)  
Quittierbare Alarm-SFCs sind: ALARM\_SQ (SFC 17) und ALARM\_DQ (SFC 107)  
Nicht quittierbar sind die SFCs: ALARM\_S (SFC 18) und ALARM\_D (SFC 108)
- Diagnosemeldungen  
Systemdiagnose (ID 0x1000-0x79FF, 0xC000-0xFFFF, 0xF900-0xF9FF)  
Anwenderdiagnose (ID 0x8000-0xB9FF) mit WR\_USMSG (SFC 52)
- Statepath-Alarm  
Wird vom OPC Alarm&Event-Server erzeugt, wenn sich der Verbindungszustand ändert.

#### Was kann je Meldung angegeben werden?

In diesem Abschnitt werden auszugsweise die Einstellungen für Meldetexte, Sourcen und Areas in der Projektierung gezeigt.

### Projektierte Meldetexte

- Eingabe für programmierte bausteinbezogene Meldungen (Alarm-SFBs-SFCs):  
Der Meldetext wird in STEP 7 beim zugehörigen Instanzdatenbaustein unter ... "Spezielle Objekteigenschaften" > "Meldung ..." erfasst.  
Im Dialog wird der Meldetext in der Spalte "Meldetext" eingetragen. Begleitwerte können in den Text integriert werden.

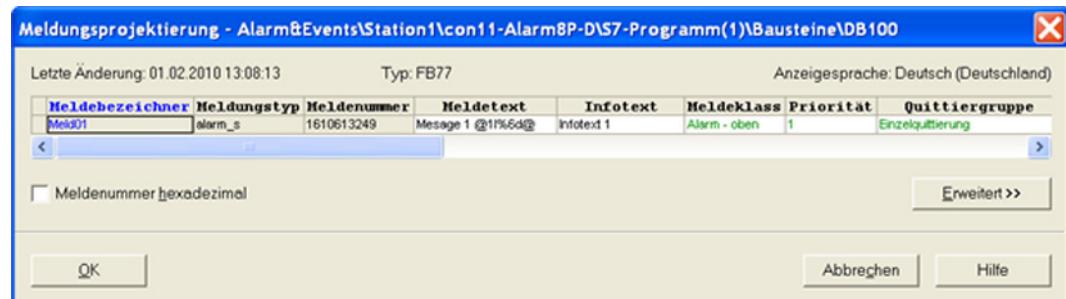


Bild 3-3 Projektierungs-Dialog eines Alarm-SFCs mit dem ALARM\_SQ (SFC17), der im FB1 aufgerufen wurde, Instanz-DB ist DB1.

- Eingabe für symbolbezogene Meldungen (SCANs):  
Der Meldetext wird in STEP 7 im Symbol-Editor Kontextmenü "Spezielle Objekteigenschaften" > "Meldung ..." eines Meldesymbols erfasst. Bei diesen Symbolen ist die Checkbox "Meldung" gesetzt.  
Im Dialog wird der Meldetext in der Spalte "Meldetext" eingetragen.

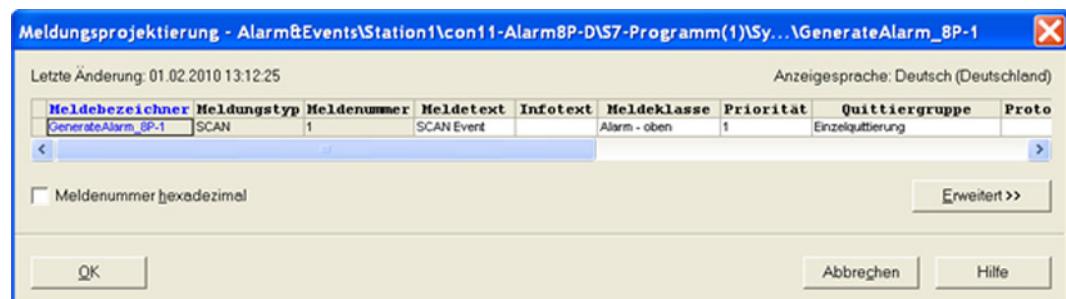


Bild 3-4 Projektierungs-Dialog eines symbolbezogenen Alarms für das Merkerbit 3.1

- Eingabe für Diagnosemeldungen:

Kommende und Gehende Meldetexte zu einer Anwenderdiagnosemeldung WR\_UMSG (SFC52) werden im Dialog in den Spalten "Meldetext kommend" und "Meldetext gehend" eingegeben.

Der Eingabe-Dialog wird im SIMATIC Manager im Kontextmenü von S7-Programmen von SIMATIC S7-Stationen "Spezielle Objekteigenschaften" > "Meldung ..." angeboten



Bild 3-5 Beispiel für die Projektierung einer Anwenderdiagnosemeldung zum WR\_UMSG (SFC52)

- Projektierte Herkunft:  
In den Dialogen in denen auch der Meldetext erfasst wird, kann auch die Meldungsherkunft (Source) eingegeben werden. Als Herkunft wird der 1. Zusatztext (Herkunft bei gesetztem Attribut am Datenbaustein S7\_alarm\_ui=1) verwendet, siehe Abbildung 3-6.  
Ist dieser nicht gesetzt, wird der Pfadname zu dem Baustein verwendet, der dem Meldetext in STEP 7 zugeordnet ist.  
z. B. SIMATIC400(1)/CPU416.3DP/S7-Programm(1)/DB10/DB100.  
Beim StatepathAlarm wird als Herkunft der projektierte Verbindungsname verwendet.
- Projektierter Bereich (Area):  
Als Bereich wird der 2. Zusatztext (OS-Bereich bei gesetztem Attribut S7\_Alarm\_ui=1) verwendet, siehe Abbildung 3-6.  
Ist dieser nicht gesetzt, wird auch hier der Pfadname vom STEP 7-Projekt verwendet.  
Beim StatepathAlarm wird als Bereich der projektierte Verbindungsname verwendet.

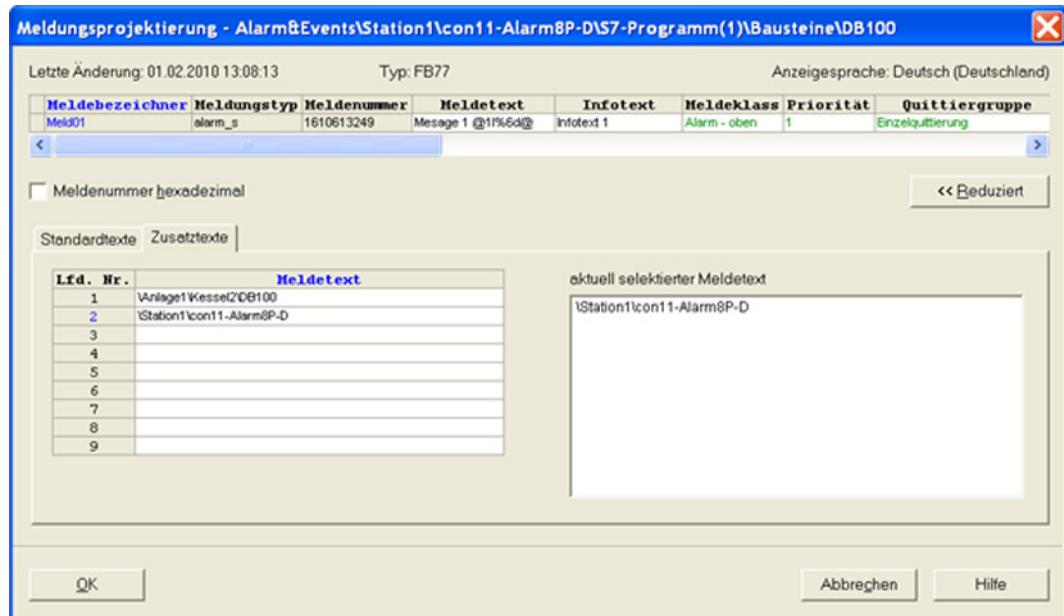


Bild 3-6 1. Zusatztext (Source) und 2. Zusatztext (Area)

### Wie werden Begleitwerte in Meldungen eingebettet?

Melde- und Infotexte können formale Parameter enthalten. Diese Parameter können teilweise durch formatierte Begleitwerte der Meldung ersetzt werden.

Der Aufbau und die Bedeutung der möglichen formalen Parameter können Sie in STEP 7 unter "Hilfe" > "Begleitwerte in Meldungen einfügen" nachlesen.

Abweichend dazu unterstützt der OPC-Server für S7-Kommunikation weitere Formatangaben, die in der nachfolgenden Auflistung, mit \*) gekennzeichnet sind. Die formalen Parameter werden nur in Melde- und im Infotexten durch formatierte Begleitwerte ersetzt. Bei den OPC-Servern für S7-Kommunikation werden in den Zusatztexten die formalen Parameter nicht ersetzt.

Die formalen Parameter haben den Aufbau: @<Nummer><Typ><Format>@

<Nummer>	Steht für die Nummer des Begleitwertes, der für den formalen Parameter verwendet werden soll.																												
<Typ>	<p>Beschreibt den Typ des Begleitwertes. Folgende Werte sind möglich:</p> <p>Y - Byte W - Word D - DWord I - Integer D - Double Integer B - Bool C - Char R - Real &lt;leer&gt; - keine Typangabe: Hinweis: Für die Formate "%t", "%s" und "%Y" darf kein Typ angeben werden. Für die anderen Formate wählen Sie einen zum Begleitwert und Format passenden Typ. Es erfolgt keine Typkonvertierung des angegebenen Begleitwertes.</p>																												
<Format>	<p>Beschreibt die Formatierung des Begleitwertes im Text. Folgende Formatierungen sind möglich:</p> <table border="1"> <tr> <td>%[i]X</td><td>Hexadezimale Darstellung des Begleitwertes mit Großbuchstaben.</td></tr> <tr> <td>%[i]x</td><td>Hexadezimale Darstellung des Begleitwertes mit Kleinbuchstaben. *)</td></tr> <tr> <td>%[i]u</td><td>Darstellung als Dezimalzahl ohne Vorzeichen.</td></tr> <tr> <td>%[i]d</td><td>Darstellung als Dezimalzahl mit Vorzeichen.</td></tr> <tr> <td>%[i]i</td><td>Darstellung als Dezimalzahl mit Vorzeichen. *)</td></tr> <tr> <td>%[i]b</td><td>Darstellung als Binärzahl.</td></tr> <tr> <td>%[i].[y]f</td><td>Darstellung als Festpunktzahl mit y Nachkommastellen.</td></tr> <tr> <td>%[i].[y]e</td><td>Darstellung als Gleitpunktzahl mit y Nachkommastellen in Exponentendarstellung mit Kleinbuchstaben. *)</td></tr> <tr> <td>%[i].[y]E</td><td>Darstellung als Gleitpunktzahl mit y Nachkommastellen in Exponentendarstellung mit Großbuchstaben. *)</td></tr> <tr> <td>%[i].[y]g</td><td>Darstellung als Festpunktzahl oder als Gleitpunktzahl mit y Nachkommastellen mit Kleinbuchstaben. *)</td></tr> <tr> <td>%[i].[y]G</td><td>Darstellung als Festpunktzahl oder als Gleitpunktzahl mit y Nachkommastellen mit Großbuchstaben. *)</td></tr> <tr> <td>%[i]s</td><td>Der Begleitwert wird als mit 0 terminierte Zeichenkette interpretiert.</td></tr> <tr> <td>%t#&lt;datei&gt;</td><td>Der Begleitwert wird als Index in die Textbibliothek &lt;datei&gt; interpretiert und der Text aus der Textbibliothek eingefügt.</td></tr> <tr> <td>%Y[locale] [#DT-Format]</td><td> <p>Der Begleitwerte wird als DATE_AND_TIME interpretiert und nach dem Format DT-Format unter Berücksichtigung von locale formatiert. *)</p> <p>Die Angabe "locale" steht für eine Locale-Bezeichnung, z. B. german, deu oder us. Fehlt diese Angabe, wird die aktuelle Locale verwendet.</p> <p>DT-Format entspricht der Formatangabe der Standardfunktion strftime() mit der Erweiterung, dass %s für die dreistellige Ausgabe der Millisekunden steht. Wird DT-Format nicht angegeben, so wird %c verwendet, was der Datum- und Uhrzeit-Repräsentation entspricht.</p> </td></tr> </table>	%[i]X	Hexadezimale Darstellung des Begleitwertes mit Großbuchstaben.	%[i]x	Hexadezimale Darstellung des Begleitwertes mit Kleinbuchstaben. *)	%[i]u	Darstellung als Dezimalzahl ohne Vorzeichen.	%[i]d	Darstellung als Dezimalzahl mit Vorzeichen.	%[i]i	Darstellung als Dezimalzahl mit Vorzeichen. *)	%[i]b	Darstellung als Binärzahl.	%[i].[y]f	Darstellung als Festpunktzahl mit y Nachkommastellen.	%[i].[y]e	Darstellung als Gleitpunktzahl mit y Nachkommastellen in Exponentendarstellung mit Kleinbuchstaben. *)	%[i].[y]E	Darstellung als Gleitpunktzahl mit y Nachkommastellen in Exponentendarstellung mit Großbuchstaben. *)	%[i].[y]g	Darstellung als Festpunktzahl oder als Gleitpunktzahl mit y Nachkommastellen mit Kleinbuchstaben. *)	%[i].[y]G	Darstellung als Festpunktzahl oder als Gleitpunktzahl mit y Nachkommastellen mit Großbuchstaben. *)	%[i]s	Der Begleitwert wird als mit 0 terminierte Zeichenkette interpretiert.	%t#<datei>	Der Begleitwert wird als Index in die Textbibliothek <datei> interpretiert und der Text aus der Textbibliothek eingefügt.	%Y[locale] [#DT-Format]	<p>Der Begleitwerte wird als DATE_AND_TIME interpretiert und nach dem Format DT-Format unter Berücksichtigung von locale formatiert. *)</p> <p>Die Angabe "locale" steht für eine Locale-Bezeichnung, z. B. german, deu oder us. Fehlt diese Angabe, wird die aktuelle Locale verwendet.</p> <p>DT-Format entspricht der Formatangabe der Standardfunktion strftime() mit der Erweiterung, dass %s für die dreistellige Ausgabe der Millisekunden steht. Wird DT-Format nicht angegeben, so wird %c verwendet, was der Datum- und Uhrzeit-Repräsentation entspricht.</p>
%[i]X	Hexadezimale Darstellung des Begleitwertes mit Großbuchstaben.																												
%[i]x	Hexadezimale Darstellung des Begleitwertes mit Kleinbuchstaben. *)																												
%[i]u	Darstellung als Dezimalzahl ohne Vorzeichen.																												
%[i]d	Darstellung als Dezimalzahl mit Vorzeichen.																												
%[i]i	Darstellung als Dezimalzahl mit Vorzeichen. *)																												
%[i]b	Darstellung als Binärzahl.																												
%[i].[y]f	Darstellung als Festpunktzahl mit y Nachkommastellen.																												
%[i].[y]e	Darstellung als Gleitpunktzahl mit y Nachkommastellen in Exponentendarstellung mit Kleinbuchstaben. *)																												
%[i].[y]E	Darstellung als Gleitpunktzahl mit y Nachkommastellen in Exponentendarstellung mit Großbuchstaben. *)																												
%[i].[y]g	Darstellung als Festpunktzahl oder als Gleitpunktzahl mit y Nachkommastellen mit Kleinbuchstaben. *)																												
%[i].[y]G	Darstellung als Festpunktzahl oder als Gleitpunktzahl mit y Nachkommastellen mit Großbuchstaben. *)																												
%[i]s	Der Begleitwert wird als mit 0 terminierte Zeichenkette interpretiert.																												
%t#<datei>	Der Begleitwert wird als Index in die Textbibliothek <datei> interpretiert und der Text aus der Textbibliothek eingefügt.																												
%Y[locale] [#DT-Format]	<p>Der Begleitwerte wird als DATE_AND_TIME interpretiert und nach dem Format DT-Format unter Berücksichtigung von locale formatiert. *)</p> <p>Die Angabe "locale" steht für eine Locale-Bezeichnung, z. B. german, deu oder us. Fehlt diese Angabe, wird die aktuelle Locale verwendet.</p> <p>DT-Format entspricht der Formatangabe der Standardfunktion strftime() mit der Erweiterung, dass %s für die dreistellige Ausgabe der Millisekunden steht. Wird DT-Format nicht angegeben, so wird %c verwendet, was der Datum- und Uhrzeit-Repräsentation entspricht.</p>																												

In eckige Klammern [] gesetzte Angaben sind dabei optional. Der Wert "i" steht für die Anzahl von Zeichen, die für die Darstellung verwendet werden sollen.

Die nachfolgenden Zeichenfolgen werden außerhalb von formalen Parametern wie folgt ersetzt:

- \n Wird ersetzt durch das Zeichen <neue Zeile> (0x0a).
- \r Wird ersetzt durch das Zeichen <CR> (0x0d).
- \t Wird ersetzt durch das Tabulatorzeichen (0x09).
- \\" Wird ersetzt durch das doppelte Anführungszeichen (0x22).
- \@ Wird ersetzt durch das @-Zeichen (0x64). Das Zeichen leitet damit keine formalen Parameter ein.

### 3.1.7 Wie können Herkunftsangaben, Quellen und Bedingungen im Namensraum durchsucht und gefiltert werden?

Die Schnittstelle "IOPC EventAreaBrowser" bietet Methoden zum Suchen von Herkunftsgebieten, Quellen und Bedingungen (Conditions). Mit den zugehörigen Filterfunktionen kann nach Teilbereichen gefiltert werden.

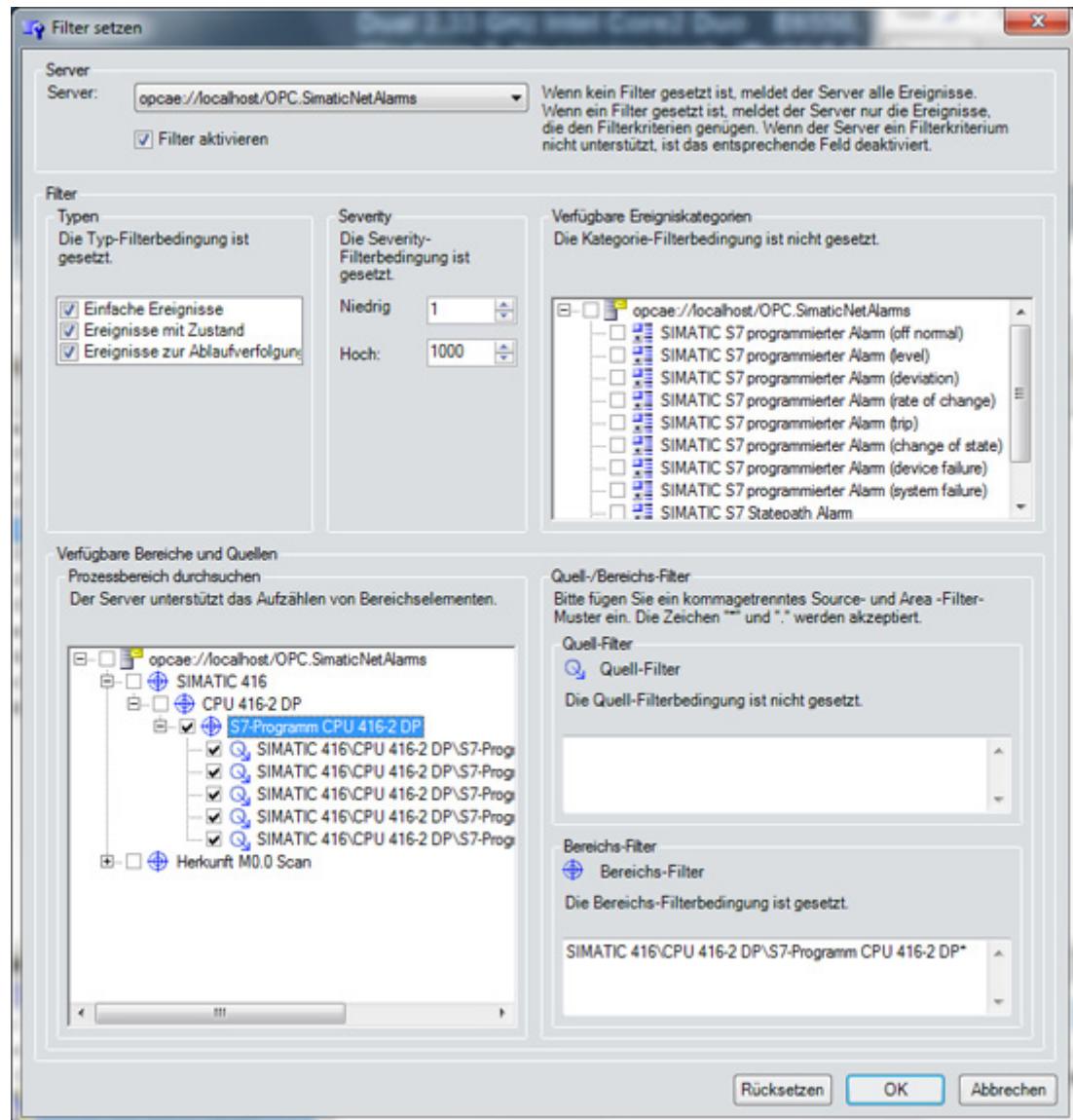


Bild 3-7 Filter-Dialog im OPC Scout V10

Um auch unterlagerte Bereiche mit einbeziehen zu können, muss in der Filterzeichenkette ein "\*" -Zeichen angehängt werden.

z. B. alle Alarme einer CPU 416 erhält den Filter "SIMATIC 400(1)/CPU 416-3DP\*\*"

## 3.1 Event-Server für S7-Kommunikation

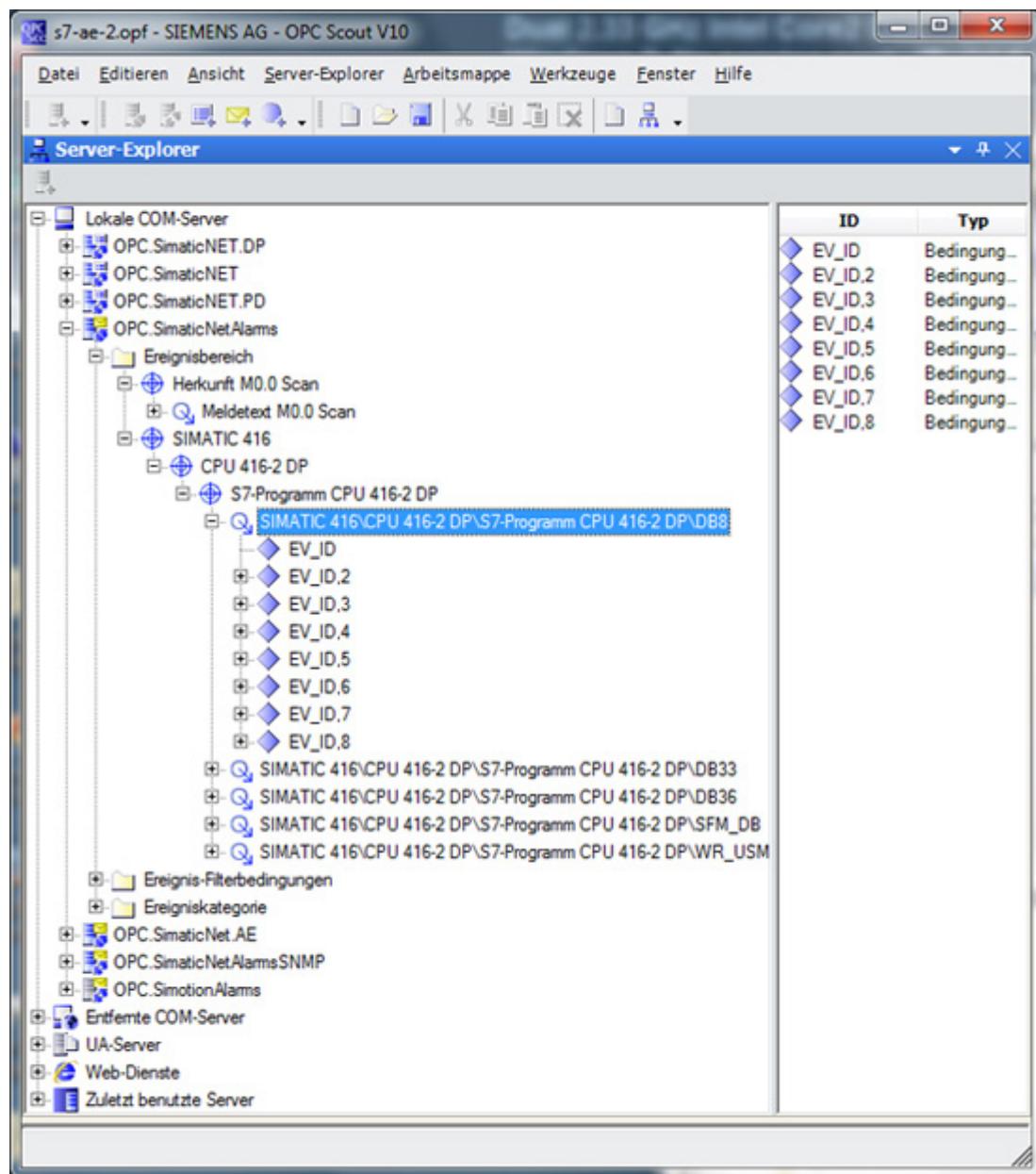


Bild 3-8 Herkunfts-Namensraum einer CPU 416. Über den 1. Zusatztext kann die Herkunft spezifisch projektiert werden.

### 3.1.8 Abbildung der STEP 7-Projektierungswerte auf OPC S7 Alarme & Events-Parameter

In der folgenden Tabelle wird die Abbildung von Projektierungsdaten, wie die "EventID" aus STEP 7 auf OPC A&E-Parameter wie "Condition" dargestellt:

Projektierte Parameter	OPC-Wert	Bedeutung
EventID ⇒ Condition ⇒ projektierte EventID wird auf Condition abgebildet	"HIHI" "HI" (Default) "LO" "LOLO"	Zustand "HIHI" Zustand "HI" Zustand "LO" Zustand "LOLO"
EVENTCATEGORY_CAT_LEVEL EVENTCATEGORY_CAT_DEVIATION EVENTCATEGORY_CAT_ROC		Eine anwenderseitig projektierte EventID darf nur einen der vier möglichen Werte annehmen.
EventID ⇒ Condition EVENTCATEGORY_CAT_OFFNORMAL	"CFN"	Zustand "CFN"
EventID ⇒ Condition EVENTCATEGORY_CAT_TRIP	"TRIP"	Zustand "TRIP"
EventID ⇒ Condition EVENTCATEGORY_CAT_COS	"COS"	Zustand "COS"
EventID ⇒ Condition EVENTCATEGORY_CAT_DEVICEFAILURE EVENTCATEGORY_CAT_SYSTEMFAILURE	"FAILURE"	Zustand "FAILURE"
Herkunft ⇒ Source	<Verbindungsname>/ ALARM"<Event-ID> {,<Subevent-ID>} Oder <Verbindungsname>/ STATEPATH z.B. "s7_verb1/ALARM10, 5" z.B. "s7_verb1/STATEPA TH"	Eindeutige Kombination aus Verbindungsname und EventID bzw. STATEPATH  Alternativ: Projektierte Source, sprachabhängig. Source muss eindeutig sein!
Meldungsgewicht ⇒ Severity	1 ... 1000 siehe folgende Tabelle	Severity aus der Projektierung oder aus den Alarmdaten

Projektierte Parameter	OPC-Wert	Bedeutung
Meldeklasse ⇒ Category	40	Kategorie CAT_LEVEL
	41	Kategorie CAT_DEVIATION
	42	Kategorie CAT_ROC
	43	Kategorie CAT_OFFNORMAL
	44	Kategorie CAT_TRIP
	45	Kategorie CAT_COS
	46	Kategorie CAT_DEVICEFAILURE
	47	Kategorie CAT_SYSTEMFAILURE
Projektierte Herkunft des Zeitstempels ⇒TimeStamp	00.00.0000 00:00:00.0000	Zeitstempel (CPU, CPU + OFFSET, PCZeit)

Priorität: Die Priorität wird in eine OPC-Alarmseverity nach dieser Tabelle umgewandelt werden:

S7-Priorität	OPC-Alarmseverity
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
13	813
14	875
15	938
16	1000

### 3.1.9

### S7-Demoalarme

Mit der DEMO-Verbindung werden Datenbaustein-Objekte im Namensraum für Data Access angeboten. Die DEMO-Verbindung ist zum Kennenlernen des SIMATIC NET OPC-Systems gedacht und kann über die Konfiguration hinzugeschaltet werden.

Es können auch S7-Meldungen simuliert werden. Diese Meldungen können durch Setzen und Rücksetzen der Bits des ersten Bytes des DEMO-Datenbausteins "db20" ausgelöst werden.

#### COM DA S7

**ItemID:**

S7: [DEMO] DB20, X0.0

S7: [DEMO] DB20, X0.1

...

S7: [DEMO] DB20, X0.7

**OPC UA S7****NameSpaceID**

S7:

**Nodeld:**

...

DEMO.db20.0, x7

Setzen der Bits von "False" auf "True" lösen kommende OPC-Alarme aus, Rücksetzen der Bits von "True" auf "False" lösen gehende OPC-Alarme aus,

Die Alarme sind implizit quittiert wenn entsprechende Bits im dritten Byte des DEMO-Datenbausteins "db20" auf "True" gesetzt sind:

S7: [DEMO] DB20, X2.0 ... bzw.

S7: DEMO.db20.2, x0 ...

Durch Rücksetzen dieser Bits auf "False" können quittierbare Alarme eingestellt werden.

Durch Auslösung der Alarme über Data Access werden

- für OPC Alarms & Events für S7-Kommunikation  
Conditional Alarme der Kategorie EVENTCATEGORY\_S7OFFNORMAL = 43
- für OPC UA Alarms & Conditions für S7-Kommunikation  
UA Alarme des EventType = Nodeld (ns=S7TYPES:, i=43)  
-> Referenzierte Nodeld des S7OffNormalAlarmType

erzeugt.

## 3.2 Simple Event-Server für SNMP-Kommunikation

### SNMP-Traps

Traps sind Nachrichten, die ohne Aufforderung des OPC-Servers an diesen gesendet werden. Es gibt sieben generische Traps, die bei vielen SNMP-fähigen Gerät verfügbar sind. Darüber hinaus gibt es gerätespezifische Traps, die in der MIB-Datei beschrieben sind.

---

#### Hinweis

Gerätespezifische MIBs müssen über den MIB-Compiler (Bestandteil von STEP 7) eingebunden werden.

Traps müssen in den jeweiligen Geräten aktiviert und ihre Zielstation (hier: SNMP-OPC-Server) projektiert werden.

---

### Generische Traps

#### warmStart

Reboot des Rechners, wenn dieser bereits Verbindung zum Netz hatte.  
Wird gesendet, wenn das Gerät einen Warmstart durchgeführt hat.

#### coldStart

Neustart des Rechners, wenn dieser noch keine Verbindung zum Netz hatte.  
Wird gesendet, wenn das Gerät einen Kaltstart durchgeführt hat.

#### linkDown

Wird gesendet, wenn eine vom Gerät ausgehende Verbindung abgebaut wurde.

#### linkUp

Wird gesendet, wenn eine vom Gerät ausgehende Verbindung aufgebaut wurde.

#### authenticationFailure

Wird gesendet, wenn ein unbefugter Zugriff auf das Gerät erfolgte.

#### egpNeighborLoss

Der EGP-Nachbar (EGP = Exterior Gateway Protocol) des Geräts ist außer Betrieb. Das Exterior Gateway Protocol dient zum Austausch von Routing-Information zwischen zwei benachbarten Gateway-Hosts.

#### enterpriseSpecific

Wird gesendet, wenn ein gerätespezifischer Trap gesendet wurde.

### Weiterleitung der Traps

Die Traps werden als sogenannte "Simple Events" zum OPC A&E-Server weitergereicht. Daneben wird die Häufigkeit des Auftretens und die Beschreibung eines Traps als OPC DA-Item bereitgestellt.

Voraussetzungen:

- Die IP Adresse des PC mit dem OPC-SNMP-Server muss beim SNMP-fähigen Gerät als Trap-Empfänger eingetragen sein und die Traps müssen aktiviert sein. Die Umsetzung

erfolgt durch gerätespezifische Projektierungstools (siehe Handbuch "PC-Stationen in Betrieb nehmen"; Web based Management von OSM/ESM/SCALANCE).

- Der SNMP Dienst von Windows muss installiert sein.

OPC-Alarm-Server-Name: OPC.SimaticNetAlarmsSNMP

## Alarms & Events Kategorien

Für SNMP Geräte werden folgende Alarms & Events Kategorien bereitgestellt, die nachfolgend näher beschrieben werden:

Kategorie-Nummer	Bezeichnung	Beschreibung
20	EVENTCATEGORY_SNMP_GENERICTRAP	SNMP Trap Event (Generischer Trap)
21	EVENTCATEGORY_SNMP_SPECIFICTRAP	SNMP Trap Event (Spezifischer Trap)

Generische Traps sind durch Standards festgelegt und für alle Geräte in Ihrer Bedeutung gleich. Spezifische Traps sind geräteabhängig. Die Information, ob ein Trap generisch oder spezifisch ist, erhält der SNMP-OPC-Server über die Geräteprofildatei.

Diese Kategorien "generisch" bzw. "spezifisch" sind im Zusammenhang mit dem Begriff "Category" zu verstehen. Bei jedem Partnergerät ("Source") können Alarne und Events jeder Kategorie auftreten. Verschiedene Alarne und Events ein und derselben Kategorie vom gleichen Partnergerät werden mit unterschiedlichen EventIDs gekennzeichnet.

Traps werden vom Kommunikationspartner ausgelöst und vom SNMP-Protokoll an den SIMATIC NET SNMP-OPC-Server weitergereicht. Traps können bei hohen Kommunikationsbelastungen vom SNMP-Protokoll verworfen werden, so dass der SIMATIC NET SNMP-OPC-Server möglicherweise nicht alle Traps enthält. Dies ist eine allgemeine Einschränkung des SNMP-Protokolls.

Traps haben keinen Zustand und sind nicht zu quittieren, daher verwaltet der SIMATIC NET SNMP-OPC-Server für die Traps keine Zustandsmaschine.

Mit dem Aufruf der Rückruffunktion *OnEvent* des Client erhält der Client eine Liste von Ereignissen. Der SNMP-OPC-Event-Server liefert dabei folgende serverspezifische Parameter:

Parameter	Bedeutung
dwEventCategory	Als Ereigniskategorie ist folgender Wert möglich: EVENTCATEGORY_SNMP_GENERICTRAP = 20 EVENTCATEGORY_SNMP_SPECIFICTRAP = 21
dwEventType	Der SNMP OPC-Event-Server unterstützt nur den folgenden Typ: OPC_SIMPLE_EVENT Diese Konstante hat den Wert 0x0001.
dwNumEventAttrs	Die Anzahl der mit der Meldung gelieferten Attribute hängt von der Anzahl der mitgelieferten Begleitwerte ab.

Parameter	Bedeutung
dwSeverity	Für die Schwere des Fehlers wird ein Vorgabewert zurückgeliefert. Dieser Vorgabewert kann für einzelne Meldungsnummern eines Kommunikationspartners in der Konfigurationsdatenbank verändert werden.
pEventAttributes	Diese Struktur beinhaltet die mit dem Ereignis gelieferten Attribute. Die Attribute beinhalten auch die vom Partnergerät gelieferten Begleitwerte der Meldung.
szMessage	Trap-Ereignisnamen aus der Geräteprofildatei: <Namenstext> z. B. TrapPowerDown
szSource	Als Quelle für die Benachrichtigung wird vom SNMP-OPC-Event-Server die Geräteinformation des meldenden SNMP-Geräts angegeben. Diese entspricht dem Gerätenamen einer SNMP-Variablen für OPC Data Access: SNMP:\<Gerät> z. B. SNMP:\Switch33

Zu diesem Event können zusätzliche Informationen in Form von Attributen mitgeliefert werden. SNMP-Traps liefern folgende Attribute:

Attribut ID Wert	Attribute	Bedeutung
EVENT_ATTR_SNMP_ PCTIME 6100	VT_DATE	Zeitpunkt, an dem der OPC-Event-Server die Meldung bekommen hat.
EVENT_ATTR_SNMP_ TIMESTAMP 6101	AsnTimeticks (VT_UI4)	Zeitpunkt, zu dem die Meldung auf dem Partnergerät erzeugt wurde.
EVENT_ATTR_SNMP_ EVENTID 6102	LONG (VT_I4)	innerhalb des Geräts eindeutige Trap Nummer
EVENT_ATTR_SNMP_ ENTERPRISE 6103	AsnObjectIdentifier (VT_BSTR)	Gerätespezifischer Trap
EVENT_ATTR_SNMP_ AGENTADDRESS 6104	AsnNetworkAddress (VT_BSTR)	IP-Adresse Agent
EVENT_ATTR_SNMP_ SOURCEADDRESS 6105	AsnNetworkAddress (VT_BSTR)	IP-Adresse Trap-Source
EVENT_ATTR_SNMP_ VARBINDINGS 6106	SnmpVarBindList (VT_ARRAY   VT_VARIANT)	variable Bindings des gesendeten Traps

## 3.3 Alarms & Events-Server für S7- und SNMP-Kommunikation

### 3.3.1 Die A&E-Server von SIMATIC NET

#### Welche A&E-Server gibt es bei SIMATIC NET?

Ähnlich wie bei OPC Data Access , wo der Server "OPC.SimaticNET" alle Kommunikations-Protokolle zusammenfasst, kann für OPC Alarms & Events der Server "OPC.SimaticNET.AE" verwendet werden, der die Kommunikations-Protokolle SNMP und S7 zusammenfasst. Damit können innerhalb eines Servers S7-Alarne und SNMP-Traps empfangen werden.

Die einzelnen A&E-Server wie "OPC.SimaticNetAlarms" und "OPC.SimaticNetAlarmsSNMP" können parallel benutzt werden.

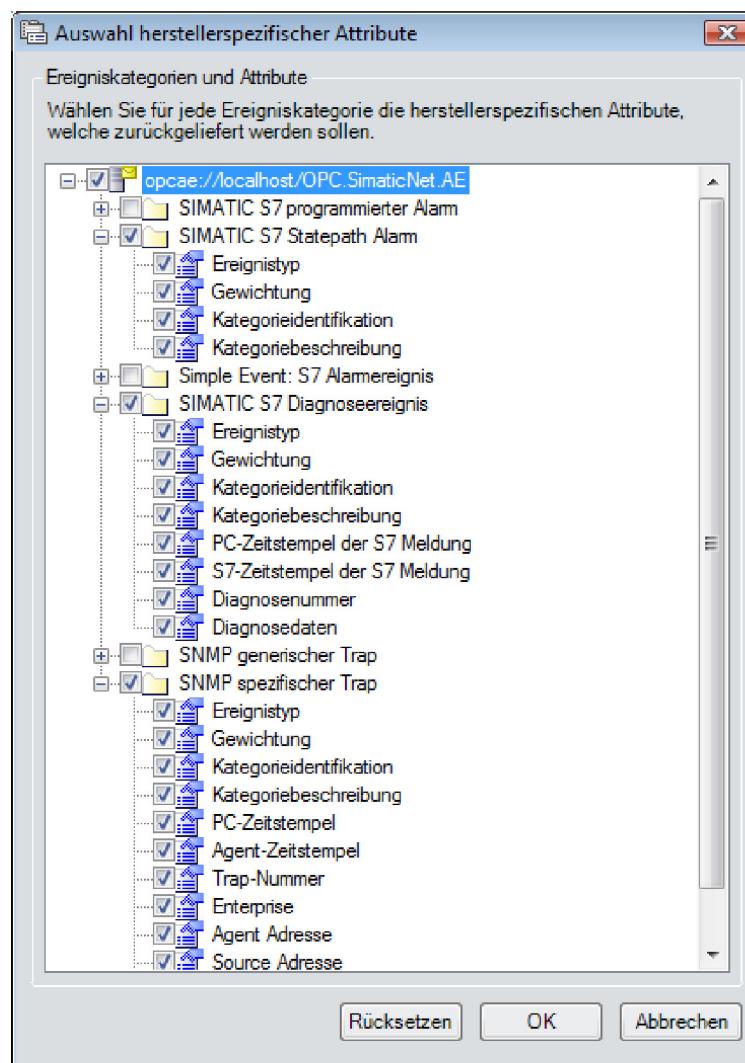


Bild 3-9 Der Alarms & Events-Server "OPC.SimaticNET.AE"; Ansicht im OPC Scout V10

### **3.3.2 Wie sind die Namen (ProgID) der Server?**

#### **ProgID**

Die ProgIDs der SIMATIC NET OPC Alarms & Events-Server lauten:

- OPC.SimaticNET.AE
- OPC.SimaticNETAlarms
- OPC.SimaticNETAlarmsSNMP

### **3.3.3 Wie kann das Protokoll der Alarmquelle erkannt werden?**

#### **Wie kann ich Alarmquellen suchen?**

Zum Durchsuchen der möglichen Alarmquellen steht die Schnittstelle "IOPCEventAreaBrowser" zur Verfügung. Sie unterstützt Methoden zum Filtern und Abfragen der OPC-Bereichs- (area) und Quell- (source) Parameter.

Folgende feste Bereichs- und Quell-Präfixe werden geliefert:

- \S7::
- \SNMP::

Entsprechend den Bereichen "\S7::" und "\SNMP::" kann gefiltert werden.

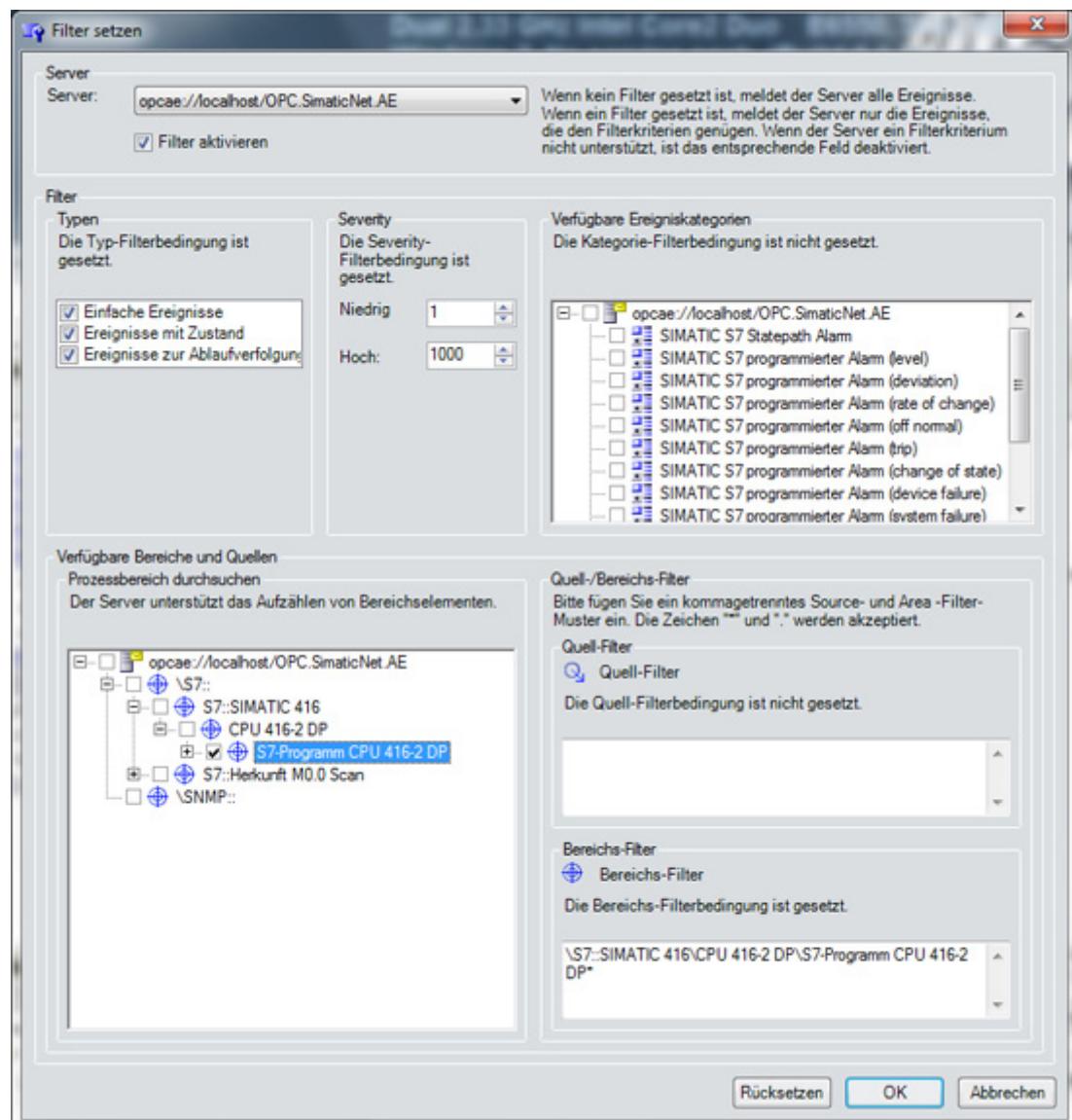


Bild 3-10      Filtern der Alarmquellen im OPC Scout V10

### Besonderheiten des Servers "OPC.SimaticNET.AE"

Wenn der Server "OPC.SimaticNET.AE" verwendet wird, dann ergibt sich für den Quell-Parameter von Alarmen ein unterschiedliches Verhalten:

- Zusätzlich zu dem protokollspezifischen Quell-Parameter (z.B. "\S7:\S7\_Verbindung\_8") kommt das Bereichs-Präfix "\S7::" hinzu. Damit ergibt sich folgender Quell-Parameter:  
"\S7::\S7:\S7\_Verbindung\_8"
- Für einen projektierbaren Quell-Text "Kesseldruck" des S7-Protokolls kann der Text auch folgendermaßen aussehen:  
"\S7::Kesseldruck"

Ansonsten verhält sich der Server "OPC.SimaticNET.AE" wie die Einzel-Server  
"OPC.SimaticNetAlarms" und "OPC.SimaticNetAlarmsSNMP".

# OPC-Server nutzen

Dieses Kapitel zeigt Ihnen verschiedene Möglichkeiten, wie Sie OPC-Server nutzen können.

## 4.1 Automation-Schnittstelle programmieren

Die Automation-Schnittstelle ist ein Zusatz für die Custom-Schnittstelle. Mit der Automation-Schnittstelle können Sie den Komfort moderner Entwicklungssysteme und Skriptsprachen auch bei der OPC-Programmierung nutzen. Es gibt sowohl eine Automation-Schnittstelle für den Zugriff auf Prozessvariablen (Data Access) als auch für die Verarbeitung von Ereignissen und Alarms (Alarms & Events).

### Anwendung

Sie arbeiten mit der Automation-Schnittstelle, wenn Sie aus einer Office-Anwendung oder mit Visual Basic eine Anwendung mit mittlerer Variablenanzahl und mittlerem Datendurchsatz erstellen wollen.

### 4.1.1 Automation-Schnittstelle programmieren für Data Access

Für Data Access gibt es ein einfaches Klassenmodell, das die Schnittstellen und deren Methoden in Klassen einteilt.

Die Automation-Schnittstelle verfeinert das für die Custom-Schnittstelle gültige Klassenmodell, um den Möglichkeiten und Eigenschaften von strukturierten Entwicklungssystemen wie Visual Basic Rechnung zu tragen.

---

#### Hinweis

Das OPC-Automation-Schnittstelle für OPC Data Access wird in einer von Siemens bereitgestellten, fehlerbereinigten Version auf Basis des von der OPC Foundation zur Verfügung gestellten Musters freigegeben.

---

#### 4.1.1.1 Was leistet das Objektmodell von OPC Data Access?

Die Klassen des Data Access-Klassenmodells enthalten folgende Objekte:

- OPCServer
- OPCGroup
- OPCItem

Für die Automation-Schnittstelle können noch weitere Objekte hinzugefügt werden. Für die

Verwaltung der Objekte "OPCGroup" und "OPCItem" gibt es eigene Collection-Objekte "OPCGroups" und "OPCItems". Die Collection-Objekte bieten Funktionen zur Verwaltung der Objekte, die ihnen zugeordnet sind.

Weiterhin gibt es ein Collection-Objekt "OPCBrowser", in dem die Browsing-Funktionen zusammengefasst werden.

## Objektmodell

Die folgende Abbildung zeigt die Objekte und die Zusammenhänge zwischen den Objekten.

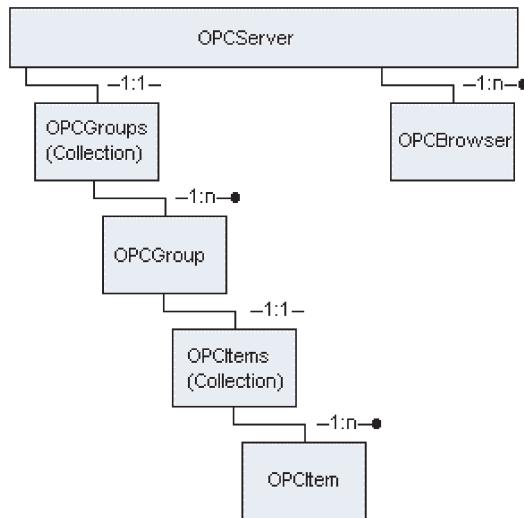


Bild 4-1 Das Objektmodell von OPC Data Access

### 4.1.1.2 Was müssen Sie bei der Programmierung beachten?

- Bei der Automation-Schnittstelle müssen optionale Parameter als Variant übergeben werden. In Visual Basic sollten die optionalen Parameter aber direkt mit dem vorgesehenen Typ der optionalen Variablen deklariert werden. So wird sichergestellt, dass der Variant den richtigen Datentyp enthält.
- Asynchrone Funktionen  
Die gruppenspezifische Eigenschaft *IsSubscribed* muss auf *True* gesetzt werden, damit die Variablen beobachtet werden.
- Die Deklaration der Objekte, die Ereignisse empfangen sollen, muss in Visual Basic mit dem Zusatz *withEvents* erfolgen.

#### Beispiel:

*Dim withEvents MyOPCGroup as OPCGroup*

- Laut OPC-Spezifikation beginnen die Arrays stets mit dem Index 1. Definieren Sie das in Ihrem Programm wie folgt:

```
Option Base 1 'ARRAYs are always Option Base 1
```

### 4.1.1.3 Objekte der Automation-Schnittstelle für Data Access

Im Folgenden werden die Eigenschaften, Methoden und Ereignisse für Data Access aufgelistet. Es werden nur die SIMATIC NET spezifischen Besonderheiten beschrieben. Eine detaillierte Beschreibung der Eigenschaften, Methoden und Ereignisse finden Sie in den entsprechenden OPC-Spezifikationen.

#### Objekte

Es gibt folgende Objekte für die Automation-Schnittstelle für Data Access:

#### Objekt OPCServer

Die Objekte OPCServer der Klasse OPC-Server werden durch den Client erzeugt.

Die Eigenschaften der Objekte OPCServer enthalten allgemeine Informationen über den Server. Mit der Erzeugung eines Objekts wird auch ein Collection-Object OPCGroups als Eigenschaft des Objekts OPCServer angelegt.

#### Eigenschaften von OPCServer

Im Folgenden werden die Eigenschaften für das Objekt OPCServer aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Eigenschaften finden Sie in der folgenden OPC-Spezifikation:

Data Access Automation Interface

Version 2.02

February 4, 1999

Die Eigenschaft	bedeutet
Bandwidth	Liefert die Bandbreite des Servers. Bandwidth wird von OPC-Server für SIMATIC NET nicht unterstützt.
BuildNumber	Liefert die Build-Nummer des Servers.
ClientName	Bestimmt den Namen des Client. ClientName wird vorwiegend für Testzwecke verwendet.
CurrentTime	Liefert die aktuelle Zeit in UTC.
LastUpdateTime	Liefert den Zeitpunkt in UTC, an dem der Server das letzte Mal Daten an den Client geschickt hat.
LocaleID	Bestimmt die Sprache für Ausgabetexte. Für SIMATIC NET sind dies Deutsch und Englisch.
MajorVersion	Liefert die Hauptversionsnummer des Servers.
MinorVersion	Liefert die untergeordnete Versionsnummer des Servers.
OPCGroups	Legt die Collection von Objekten OPCGroup fest.
PublicGroupNames	Liefert den Namen der Public Group von OPC-Server. Optionale Public Groups werden von OPC-Server für SIMATIC NET nicht unterstützt.
ServerName	Liefert den Namen des verbundenen OPC-Server.
ServerNode	Liefert den Namen des Netzknottens, auf dem OPC-Server ausgeführt wird.
ServerState	Liefert den Status des Servers.

Die Eigenschaft	bedeutet
StartTime	Liefert den Startzeitpunkt des Servers in UTC.
VendorInfo	Liefert die Herstellerinformation. Der OPC-Server für SIMATIC NET liefert als Herstellerinformation "SIMATIC NET OPC-Server".

## Methoden von OPCServer

Im Folgenden werden die Methoden für das Objekt OPCServer aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Automation Interface  
Version 2.02  
February 4, 1999

Die Methode	bedeutet
Connect	Baut eine Verbindung zu OPC-Server auf. Die <i>ProgID</i> für OPC-Server lautet z. B. "OPC.SimaticNET".
CreateBrowser	Erzeugt ein Collection-Objekt OPCBrowser.
Disconnect	Baut eine Verbindung zu OPC-Server ab. OPC-Server für SIMATIC NET baut nach dem Abbau der letzten Verknüpfung zu einem OPC-Client alle Kommunikationsverbindungen ab.
GetErrorString	Liefert Fehlermeldung für einen Fehlercode. OPC-Server für SIMATIC NET unterstützt deutsche und englische Fehlermeldungen. Fehlermeldungen des Windows-Betriebssystems werden in der Installationssprache des Betriebssystems ausgegeben.
GetItemProperties	Liefert eine Liste mit Werten für die angeforderten Eigenschaften.
GetOPCServers	Liefert die Namen der registrierten OPC-Server. Der Name für OPC-Server für SIMATIC NET lautet z. B. "OPC.SimaticNET".
LookupItemIDs	Liefert eine Liste mit ItemIDs, die den Property-IDs entsprechen. OPC-Server für SIMATIC NET liefert keine PropertyIDs, die als ItemIDs dargestellt werden könnten.
QueryAvailableLocaleIDs	Liefert verfügbare Sprachcodes. OPC-Server für SIMATIC NET unterstützt deutsche und englische Fehlermeldungen. Fehlermeldungen des Windows-Betriebssystems werden in der Installationssprache des Betriebssystems ausgegeben.
QueryAvailableProperties	Liefert Eigenschaftscodes und Eigenschaften für ein OPC-Item.

## Ereignisse von OPCServer

Im Folgenden wird das Ereignis für das Objekt OPCServer aufgeführt. Es werden nur die Besonderheiten für SIMATIC NET beschrieben. Eine Beschreibung des Ereignisses finden Sie in der folgenden OPC-Spezifikation:

Data Access Automation Interface  
Version 2.02  
February 4, 1999

Das Ereignis	bedeutet
ServerShutdown	Wird ausgelöst, wenn OPC-Server heruntergefahren wird. OPC-Server für SIMATIC NET löst dieses Ereignis aus, wenn im Konfigurationsprogramm die Aufforderung zum Beenden gegeben wird oder die PC-Station neue Konfigurationsdaten erhält.

### Collection-Objekt OPCBrowser

Mit dem Collection-Objekt OPCBrowser kann der Namensraum von OPC-Server untersucht werden.

Ein Objekt der Klasse OPCBrowser wird durch die Methode *CreateBrowser* des Objekts OPCServer erzeugt. Für einen Server können mehrere Objekte OPCBrowser angelegt werden.

### Eigenschaften von OPCBrowser

Im Folgenden werden die Eigenschaften für das Collection-Objekt OPCBrowser aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Eigenschaften finden Sie in der folgenden OPC-Spezifikation:

Data Access Automation Interface  
Version 2.02  
February 4, 1999

Die Eigenschaft	bedeutet
AccessRights	Bestimmt die Zugriffsrechte für die Methode <i>ShowLeafs</i> .
Count	Liefert die Anzahl der Einträge.
CurrentPosition	Liefert die aktuelle Position im Baum des Namensraums.
DataType	Bestimmt den Datentyp für die Methode <i>ShowLeafs</i> .
Filter	Bestimmt den Filter für die Methoden <i>ShowLeafs</i> und <i>ShowBranches</i> .
Organisation	Liefert die Organisationsstruktur des Namensraums. Der Namensraum von OPC-Server für SIMATIC NET ist hierarchisch aufgebaut.

### Methoden von OPCBrowser

Im Folgenden werden die Methoden für das Collection-Objekt OPCBrowser aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Automation Interface  
Version 2.02  
February 4, 1999

Die Methode	bedeutet
GetAccessPaths	Ermittelt den Access Path einer ItemID. Für OPC-Server für SIMATIC NET sollten Sie den Parameter Acces- sPath nicht verwenden.
GetItemID	Liefert - wenn möglich - die ItemID für ein Element im Namensraum des OPC-Servers.
Item	Bestimmt den Namen eines Eintrags
MoveDown	Bewegt die aktuelle Position im Namensraum eine Ebene tiefer
MoveTo	Bewegt die aktuelle Position im Namensraum auf die angegebene Posi- tion
MoveToRoot	Bewegt die aktuelle Position im Namensraum auf die Wurzel
MoveUp	Bewegt die aktuelle Position im Namensraum eine Ebene höher
ShowBranches	Bestimmt den Namen der Zweige der aktuellen Browse-Position
ShowLeafs	Bestimmt den Namen der Blätter der aktuellen Browse-Position

## Collection-Objekt OPCGroups

Das Collection-Objekt OPCGroups ist eine Sammlung von Objekten OPCGroup zur Erzeugung und Verwaltung von OPC-Gruppen. Die Standard-Eigenschaften von OPCGroups legen Standardwerte für die Erzeugung aller OPC-Gruppen fest.

Wenn das OPC-Server-Objekt erfolgreich einen *Connect*-Aufruf ausführt, wird automatisch ein Collection-Objekt OPCGroups als Eigenschaft des OPC-Server-Objekts angelegt.

---

### Hinweis

Die optionalen Public Groups werden von OPC-Server für SIMATIC NET nicht unterstützt.

---

## Eigenschaften von OPCGroups

Im Folgenden werden die Eigenschaften für das Collection-Objekt OPCGroups aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Eigenschaften finden Sie in der folgenden OPC-Spezifikation:

Data Access Automation Interface  
Version 2.02  
February 4, 1999

Die Eigenschaft	bedeutet
Count	Liefert die Anzahl der Gruppen.
DefaultGroupDeadband	Bestimmt den Anfangswert für <i>Deadband</i> für neu generierte Objekte OPCGroup.
DefaultGroupIsActive	Bestimmt den Anfangswert für <i>ActiveState</i> für neu generierte Objekte OPCGroup.

Die Eigenschaft	bedeutet
DefaultGroupLocaleID	Bestimmt den Anfangswert für <i>LocaleID</i> für neu generierte Objekte OPCGroup.
DefaultGroupTimeBias	Bestimmt den Anfangswert für <i>TimeBias</i> für neu generierte Objekte OPCGroup.
DefaultGroupUpdateRate	Bestimmt den Anfangswert für <i>UpdateRate</i> für neu generierte Objekte OPCGroup.
Parent	Liefert die Referenz auf das zugehörige Objekt OPCServer.

## Methoden von OPCGroups

Im Folgenden werden die Methoden für das Collection-Objekt OPCGroups aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Automation Interface

Version 2.02

February 4, 1999

Die Methode	bedeutet
Add	Erzeugt ein neues Objekt OPCGroup und fügt es zu der Collection hinzu.
GetOPCGroup	Bestimmt die Referenz auf Namen oder Server-Handle eines Objekts OPCGroup.
Item	Liefert die Referenz auf das indizierte Objekt der Collection.
Remove	Löscht eine Gruppe des Servers.
RemoveAll	Löscht alle Gruppen und Items des Servers.

## Ereignisse von OPCGroups

Im Folgenden wird das Ereignis für das Collection-Objekt OPCGroups beschrieben. An dieser Stelle werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine detaillierte Beschreibung des Ereignisses finden Sie in der folgenden OPC-Spezifikation:

Data Access Automation Interface

Version 2.02

February 4, 1999

Das Ereignis	bedeutet
GlobalDataChange	Benachrichtigt über Wertänderung und Zustand der aktiven Items aller aktiven Gruppen.

## Objekt OPCGroup

Die Klasse OPC-Group verwaltet die einzelnen Prozessvariablen, die OPC-Items. Mit Hilfe des Objekts OPCGroup kann ein Client semantisch sinnvolle Einheiten von OPC-Items bilden und sie so gemeinsam bearbeiten.

Die Beobachtung von Variablen und Lese- und Schreibzugriffe erfolgen gruppenspezifisch. Beispielsweise kann die Beobachtung aller OPC-Items in einer Gruppe durch den Aufruf einer einzigen Funktion aktiviert werden.

Deshalb sollten Sie z.B. alle Prozessvariablen, die auf einer Anzeigemaske eines Bedien- und Beobachtungssystems dargestellt sind, in einer Gruppe zusammenfassen und die Beobachtung der Variablen mit dem Öffnen der Anzeigemaske aktivieren.

---

#### Hinweis

Die optionalen Public Groups werden von OPC-Server für SIMATIC NET nicht unterstützt.

---

### Eigenschaften von OPCGroup

Im Folgenden werden die Eigenschaften für das Objekt OPCGroup aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Eigenschaften finden Sie in der folgenden OPC-Spezifikation:

Data Access Automation Interface  
Version 2.02  
February 4, 1999

Die Eigenschaft	bedeutet
ClientHandle	Bestimmt das Handle zur Lokalisierung eines Datums.
DeadBand	Bestimmt die Bandbreite, in der Werteänderungen nicht zu Benachrichtigungen führen.
IsActive	Bestimmt den Aktivstatus der Gruppe. Sie müssen <i>IsActive</i> auf <i>True</i> setzen, damit die Variablen dieser Gruppe beobachtet werden.
IsPublic	Zeigt, ob die Gruppe eine Public Group ist. Die optionalen Public Groups werden bei OPC-Server für SIMATIC NET nicht unterstützt.
IsSubscribed	Zeigt, ob die Variablen einer Gruppe überwacht werden sollen. Sie müssen <i>IsSubscribed</i> auf <i>True</i> setzen, damit die Variablen dieser Gruppe beobachtet werden.
LocaleID	Bestimmt die Sprache für vom Server gelieferte Zeichenketten.
Name	Bestimmt den Namen der Gruppe.
OPCItems	Bestimmt das Collection-Objekt zur Verwaltung der Items der Gruppe.
Parent	Liefert die Referenz auf das zugehörige Objekt OPCServer.
ServerHandle	Liefert ein für die Gruppe eindeutiges Handle.
TimeBias	Bestimmt das Zeit-Offset für die Umrechnung des Zeitstempels in die lokale Zeit.
UpdateRate	Bestimmt die Rate, in der ein Client über Änderungen der Werte oder Zustände von Items benachrichtigt wird.

## Methoden von OPCGroup

Im Folgenden werden die Methoden für das Objekt OPCGroup aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Automation Interface

Version 2.02

February 4, 1999

Die Methode	bedeutet
AsyncCancel	Bricht asynchronen Auftrag ab.
AsyncRead	Setzt asynchronen Lesebefehl ab.
AsyncRefresh	Erzeugt ein Ereignis für jedes aktive OPC-Item mit dem aktuellen Wert aus dem Cache.
AsyncWrite	Setzt asynchronen Schreibbefehl ab.
SyncRead	Startet synchrones Lesen von Werten, Statusinformationen und Zeitstempel eines oder mehrerer Items einer Gruppe.
SyncWrite	Startet synchrones Schreiben von Werten für ein oder mehrere Items einer Gruppe.

## Ereignisse von OPCGroup

Die OPC-Automation-Schnittstelle liefert die Änderungen der Werte aktiver Items und die Ergebnisse asynchroner Operationen über Ereignisse zurück.

Objekte in Visual Basic, die Ereignisse empfangen sollen, müssen mit *withEvents* deklariert werden.

Im Folgenden werden die Ereignisse für das Objekt OPCGroup aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Ereignisse finden Sie in der folgenden OPC-Spezifikation:

Data Access Automation Interface

Version 2.02

February 4, 1999

Das Ereignis	bedeutet
AsyncCancelComplete	Wird ausgelöst, wenn ein Cancel-Auftrag abgeschlossen ist.
AsyncReadComplete	Wird ausgelöst, wenn ein Leseauftrag abgeschlossen ist.
AsyncWriteComplete	Wird ausgelöst, wenn ein Schreibauftrag abgeschlossen ist.
DataChange	Wird ausgelöst, wenn ein oder mehrere Items einen geänderten Wert oder eine geänderte Qualität haben.

## Collection-Objekt OPCItems

Das Collection-Objekt OPCItems ist eine Sammlung von Objekten OPCItem zur Erzeugung und Verwaltung von OPC-Items. Die Eigenschaften von OPCItems legen Standardwerte für alle OPC-Items fest, die neu erzeugt werden.

Wenn ein Objekt OPCGroup erzeugt wird, wird automatisch ein Collection-Objekt OPC-Items angelegt. OPCItems ist als Eigenschaft einer Gruppe immer vorhanden und wird für die Überwachung der OPC-Variablen verwendet.

## Eigenschaften von OPCItems

Im Folgenden werden die Eigenschaften für das Objekt OPCItems aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Eigenschaften finden Sie in der folgenden OPC-Spezifikation:

Data Access Automation Interface

Version 2.02

February 4, 1999

Die Eigenschaft	bedeutet
Count	Liefert die Anzahl der Items in der Gruppe.
DefaultAccessPath	Bestimmt den Anfangswert für AccessPath für neu hinzugefügte OPC-Items. Für den OPC-Server für SIMATIC NET sollte <i>DefaultAccessPath</i> leer sein.
DefaultIsActive	Bestimmt den Anfangswert für <i>ActiveState</i> für neu hinzugefügte OPC-Items.
DefaultRequestedDataType	Bestimmt den Anfangswert für <i>RequestedDataType</i> für neu hinzugefügte OPC-Items.
Parent	Liefert die Referenz auf das zugehörige Objekt OPCGroup.

## Methoden von OPCItems

Im Folgenden werden die Methoden für das Objekt OPCItems aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Automation Interface

Version 2.02

February 4, 1999

Die Methode	bedeutet
AddItem	Fügt dem Collection-Objekt ein neues OPC-Item hinzu.
AddItems	Fügt dem Collection-Objekt mehrere OPC-Items hinzu.
GetOPCItem	Bestimmt die Referenz auf das durch <i>AddItem</i> erzeugte Server-Handle.
Item	Bestimmt die Referenz auf ein Item der Collection.
Remove	Löscht ein oder mehrere Items aus einer Gruppe.
SetActive	Setzt den Aktivstatus eines oder mehrerer Items einer Gruppe.
SetClientHandles	Ändert das Client-Handle für ein oder mehrere Items.
SetDataTypes	Setzt den Datentyp für ein oder für mehrere Items.
Validate	Prüft ein oder mehrere OPC-Items auf Gültigkeit.

## Objekt OPCItem

Ein Objekt der Klasse OPC-Item repräsentiert eine Prozessvariable, zum Beispiel ein Eingabemodul einer speicherprogrammierbaren Steuerung. Eine Prozessvariable ist ein schreibbares und/oder lesbares Datum der Prozessperipherie, wie zum Beispiel die Temperatur eines Kessels. Mit jeder Prozessvariablen ist ein Wert, eine Qualität und ein Zeitstempel verbunden.

## Eigenschaften von OPCItem

Im Folgenden werden die Eigenschaften für das Objekt OPCItem aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Eigenschaften finden Sie in der folgenden OPC-Spezifikation:

Data Access Automation Interface

Version 2.02

February 4, 1999

Die Eigenschaft	bedeutet
AccessPath	Liefert den Zugriffspfad des Items. Für den OPC-Server für SIMATIC NET sollte Access Path leer sein.
AccessRights	Liefert die Zugriffsrechte der Variablen.
CanonicalDataType	Liefert den ursprünglicher Datentyp des Items.
ClientHandle	Bestimmt das Handle zur einfacheren Zuordnung der Prozessvariablen in interne Datenstrukturen des Clients.
EUInfo	Liefert die Informationen über die Einheiten des Wertes (optional). OPC-Server für SIMATIC NET unterstützt keine Einheiten (Engineering Units).
EUType	Liefert die Einheit des gelieferten Wertes (optional). OPC-Server für SIMATIC NET unterstützt keine Einheiten (Engineering Units).
IsActive	Bestimmt, ob für das Item Benachrichtigungseignisse generiert werden sollen.
ItemID	Liefert den eindeutigen Namen des Items.
Parent	Liefert die Referenz auf das übergeordnete Objekt OPCGroup.
Quality	Liefert die Qualität des zuletzt gelesenen Wertes.
RequestedDataType	Bestimmt den geforderten Datentyp für den Wert des Items.
ServerHandle	Liefert das Server-Handle für die Identifikation eines Items.
TimeStamp	Liefert den Zeitpunkt, an dem der letzte Wert erfasst wurde.
Value	Liefert den zuletzt gültiger Wert eines Items.

## Methoden von OPCItem

Im Folgenden werden die Methoden für das Objekt OPCItem aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Automation Interface

Version 2.02

February 4, 1999

Die Methode	bedeutet
Read	Liest synchron Wert, Qualität und/oder Zeitstempel der Variablen.
Write	Setzt synchron den Wert der Variablen.

## 4.1.2 Automation-Schnittstelle programmieren für Alarms & Events

Für Alarms & Events gibt es ein einfaches Klassenmodell, das die Schnittstellen und deren Methoden in Klassen einteilt.

Die Automation-Schnittstelle verfeinert das für die Custom-Schnittstelle gültige Klassenmodell, um den Möglichkeiten und Eigenschaften von strukturierten Entwicklungssystemen wie Visual Basic Rechnung zu tragen.

### 4.1.2.1 Was leistet das Objektmodell von OPC Alarms & Events?

Die Klassen des Klassenmodells von Alarms & Events enthalten folgende Objekte:

- OPCEventServer
- OPCEventSubscriptions
- OPCEventSubscription
- OPCEventAreaBrowsers
- OPCEventAreaBrowser
- OPCEvents
- OPCEvent
- OPCEventCondition
- OPCEventSubConditions
- OPCEventSubCondition

Für die Automation-Schnittstelle können noch weitere Objekte hinzugefügt werden.

---

#### Hinweis

Da der Event Server von SIMATIC NET ein Simple Event Server ist, werden folgende Objekte nicht unterstützt: OPCEventAreaBrowsers, OPCEventAreaBrowser, OPCEventCondition, OPCEventSubConditions, OPCEventSubCondition.

---

## Objektmodell

Die folgende Abbildung zeigt die Objekte und die Zusammenhänge zwischen den Objekten.

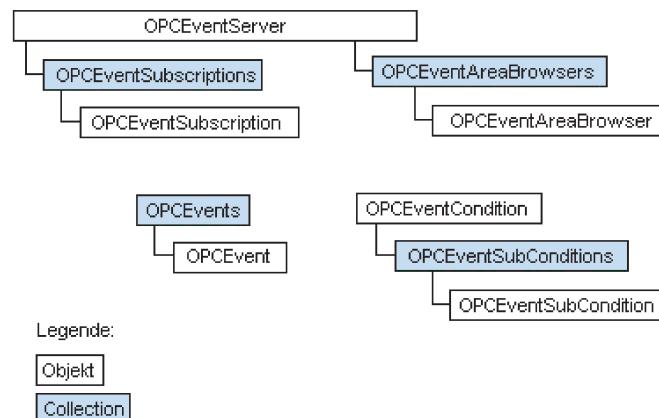


Bild 4-2 Das Objektmodell von OPC Alarms & Events

### 4.1.2.2 Was müssen Sie bei der Programmierung beachten?

Bei der Automation-Schnittstelle müssen optionale Parameter als Variant übergeben werden. In Visual Basic sollten die optionalen Parameter aber direkt mit dem Zieltyp deklariert werden. So wird sichergestellt, dass der Variant den richtigen Datentyp enthält.

- Asynchrone Funktionen  
Die gruppenspezifische Eigenschaft */sSubscribed* muss auf *True* gesetzt werden, damit die Variablen beobachtet werden.
- Die Deklaration der Objekte, die Ereignisse empfangen sollen, muss in Visual Basic mit dem Zusatz *withEvents* erfolgen.

Beispiel:

`Dim WithEvents MyOPCGroup as OPCGroup`

- Laut OPC-Spezifikation beginnen die Arrays stets mit dem Index 1. Definieren Sie das in Ihrem Programm wie folgt:  
*Option Base 1 ARRAYS are always Option Base 1*

### 4.1.2.3 Objekte der Automation-Schnittstelle für Alarms & Events

Im Folgenden werden die Eigenschaften, Methoden und Ereignisse für Alarms & Events aufgelistet. Es werden nur die SIMATIC NET spezifischen Besonderheiten beschrieben. Eine detaillierte Beschreibung der Eigenschaften, Methoden und Ereignisse finden Sie in den entsprechenden OPC-Spezifikationen.

## Objekte

Es gibt folgende Objekte für die Automation-Schnittstelle für Alarms & Events:

## Objekt OPCEventServer

Objekte OPCEventServer der Klasse OPC-Event-Server werden vom Client erzeugt. Erst nachdem OPCEventServer erzeugt ist, kann der Client auch auf andere Objekte von Alarms & Events zugreifen.

Die Eigenschaften der Objekte OPCEventServer enthalten allgemeine Informationen über den Event-Server. Mit der Erzeugung eines Objekts wird auch ein Collection-Object OPCEventServerSubscription angelegt.

OPCEventServer wird durch die Methode *Connect* mit dem Event-Server verbunden.

## Eigenschaften von OPCEventServer

Im Folgenden werden die Eigenschaften für das Objekt OPCEventServer aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Eigenschaften finden Sie in der folgenden OPC-Spezifikation:

Alarm & Events Automation Interface Standard

Version 1.01

December 15, 1999

Die Eigenschaft	bedeutet
BuildNumber	Liefert die Build-Nummer des Servers.
ClientName	Bestimmt den Namen des Client.
CurrentTime	Liefert die aktuelle Zeit in UTC.
FiltersByArea	Liefert, ob der Server nach Bereichen (Areas) filtern kann. OPC-Event-Server unterstützt keine Areas.
FiltersByCategory	Liefert, ob der Server nach Ereigniskategorien filtern kann.
FiltersByEventType	Liefert, ob der Server nach Ereignistypen filtern kann.
FiltersBySeverity	Liefert, ob der Server nach der Schwere der Ereignisse filtern kann.
FiltersBySource	Liefert, ob der Server nach Ereignisquellen (Sources) filtern kann.
LastUpdateTime	Liefert den Zeitpunkt in UTC, an dem der Server das letzte mal Daten an den Client geschickt hat.
LocaleID	Bestimmt die Sprache für Ausgabetexte.
MajorVersion	Liefert die Hauptversionsnummer des Servers.
MinorVersion	Liefert die untergeordnete Versionsnummer des Servers.
OPCEventAreaBrowsers	Bestimmt die Collection von Objekten <i>OPCAutoEventAreaBrowser</i> . OPC-Event-Server von SIMATIC NET unterstützt <i>OPCEventAreaBrowsers</i> nicht.
OPCEventSubscriptions	Bestimmt die Collection von Objekten <i>OPCEventSubscription</i> .
ServerName	Liefert den Namen des verbundenen OPC-Server.
ServerNode	Liefert die Namen des Netzknotens, auf dem OPC-Server ausgeführt wird.
ServerState	Liefert den Server-Status.
StartTime	Liefert den Zeitpunkt, an dem OPC-Server gestartet wurde.
VendorInfo	Liefert die Herstellerinformation.

## Methoden von OPCEventServer

Im Folgenden werden die Methoden für das Objekt OPCEventServer aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Alarm & Events Automation Interface Standard  
Version 1.01  
December 15, 1999

Die Methode	bedeutet
AckCondition	Bestätigt eine oder mehrere Bedingungen im Event-Server.
Connect	Baut die Verbindung zu OPC-Server auf.
Disconnect	Baut die Verbindung zu OPC-Server ab.
EnableConditionsByArea	Setzt alle Bedingungen für alle Bereiche (Areas) auf einen festgelegten Status.
EnableConditionBySrc	Setzt alle Bedingungen für alle Quellen (Sources) auf einen festgelegten Status.
GetConditionState	Gibt aktuelle Statusinformationen für die Bedingung.
GetErrorString	Wandelt einen Fehlercode in einen Fehler text um.
GetOPCEventServers	Gibt die Namen aller registrierten Event-Server zurück.
QueryAvailableLocaleIDs	Liefert verfügbare Sprachcodes.
QueryConditionNames	Liefert die Namen der Bedingungen, die für eine bestimmte Ereigniskategorie gültig sind.
QueryEventAttributes	Liefert herstellerspezifische Attribute
QueryEventCategories	Liefert die Kategorien, die der Event-Server unterstützt.
QuerySourceConditions	Liefert die Namen von Bedingungen, die mit einer bestimmten Quelle verbunden sind.
QuerySubConditionNames	Liefert die Namen von Unterbedingungen, die mit einer bestimmten Quelle verbunden sind.

## Ereignisse von OPCEventServer

Im Folgenden wird das Ereignis für das Objekt OPCEventServer aufgeführt. Es werden nur die Besonderheiten für SIMATIC NET beschrieben. Eine Beschreibung des Ereignisses finden Sie in der folgenden OPC-Spezifikation:

Alarm & Events Automation Interface Standard  
Version 1.01  
December 15, 1999

Das Ereignis	bedeutet
EventServerShutDown	Wird ausgelöst, wenn OPC-Server heruntergefahren wird. OPC-Server für SIMATIC NET löst dieses Ereignis aus, wenn im Konfigurationsprogramm die Aufforderung zum Beenden gegeben wird oder die PC-Station neue Konfigurationsdaten erhält.

## Collection-Objekt OPCEventSubscriptions

Das Collection-Objekt OPCEventSubscriptions ist eine Sammlung von Objekten OPCEventSubscription und den Methoden, um Sie zu erstellen, zu entfernen und zu verwalten.

Die Eigenschaften von OPCEventSubscriptions legen Standardwerte für alle Objekte OPCEventSubscription fest, die neu erzeugt werden.

Mit der erfolgreichen Ausführung des *Connect*-Aufrufs des Objekts OPCEventServer wird automatisch ein Collection-Objekt OPCEventSubscriptions angelegt.

OPCEventSubscriptions ist als Eigenschaft des Objekts OPCEventServer immer vorhanden und wird verwendet, um Ereignisanmeldungen zu verwalten.

## Eigenschaften von OPCEventSubscriptions

Im Folgenden werden die Eigenschaften für das Objekt OPCEventSubscriptions aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Eigenschaften finden Sie in der folgenden OPC-Spezifikation:

Alarm & Events Automation Interface Standard

Version 1.01

December 15, 1999

Die Eigenschaft	bedeutet
Count	Liefert die Anzahl der Einträge.
DefaultIsActive	Bestimmt den Anfangswert für den Aktivstatus für neu generierte Objekte OPCEventSubscription.
DefaultbufferTime	Bestimmt den Anfangswert dafür, wie oft bei neu generierten Objekten OPCEventSubscription Ereignisbenachrichtigungen verschickt werden.
DefaultMaxSize	Bestimmt den Anfangswert für die Höchstzahl von Ereignissen, die mit einer einzigen Ereignisbenachrichtigung geschickt werden für neu generierte Objekte OPCEventSubscription.

## Methoden von OPCEventSubscriptions

Im Folgenden werden die Methoden für das Objekt OPCEventSubscriptions aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Alarm & Events Automation Interface Standard

Version 1.01

December 15, 1999

Die Methode	bedeutet
Add	Legt ein neues Objekt OPCEventSubscription an und fügt es zu der Collection hinzu.
Item	Bestimmt die Referenz auf das indizierte Objekt der Collection.
Remove	Löscht ein Objekt OPCEventSubscription.
RemoveAll	Löscht alle Objekte OPCEventSubscription.

## Objekt OPCEventSubscription

Das Objekt OPCEventSubscription repräsentiert eine bestimmte Ereignis-Subscription. Unter Subscription versteht man ein Abonnement auf eine Menge von Ereignissen. Der Client startet einen Auftrag zur regelmäßigen Versendung von Ereignissen an den Event-Server.

## Eigenschaften von OPCEventSubscription

Im Folgenden werden die Eigenschaften für das Objekt OPCEventSubscription aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Eigenschaften finden Sie in der folgenden OPC-Spezifikation:

Alarm & Events Automation Interface Standard  
Version 1.01  
December 15, 1999

Die Eigenschaft	bedeutet
bufferTime	Bestimmt, wie oft beim Objekt OPCEventSubscription Ereignisbenachrichtigungen verschickt werden.
IsActive	Bestimmt den Aktivstatus des Objekts OPCEventSubscription.
MaxSize	Bestimmt die Höchstzahl von Ereignissen, die mit einer einzigen Ereignisbenachrichtigung geschickt werden.
Name	Bestimmt den Namen des Objekts OPCEventSubscription.

## Methoden von OPCEventSubscription

Im Folgenden werden die Methoden für das Objekt OPCEventSubscription aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

## Alarm &amp; Events Automation Interface Standard

Version 1.01

December 15, 1999

Die Methode	bedeutet
GetFilter	<p>Liefert den aktuellen Filter für das Objekt OPCEventSubscription. Die verwendeten Parameter des S7-OPC-Event-Servers für SIMATIC NET haben folgende Bedeutung:</p> <p><i>EventType</i> Es werden die Ereignistypen OPC_SIMPLE_EVENT und OPC_CONDITION_EVENT unterstützt.</p> <p><i>EventCategory</i> Eine Beschreibung der Ereigniskategorien finden Sie in Kapitel "Event-Server für S7-Kommunikation (Seite 391)"</p> <p><i>LowSeverity</i> <i>HighSeverity</i></p> <p><i>Areas</i> OPC-Event-Server unterstützt keine Areas.</p> <p><i>Sources</i> Sie können als Quelle einen Verbindungsnamen eingeben.</p>
GetReturnedAttributes	Liefert Attribute für jede Ereigniskategorie, die der Server mit den Ereignisbenachrichtigungen schickt.
Refresh	Aktualisiert alle Bedingungen.
RefreshCancel	Unterbricht den Ablauf der Methode Refresh. Der S7-OPC-Event-Server für SIMATIC NET unterstützt RefreshCancel nicht.
SelectReturnedAttributes	Legt für jede Ereigniskategorie die Attribute fest, die mit den Ereignisbenachrichtigungen über die Methode OnEvent geliefert werden.
SetFilter	<p>Setzt alle Filter so, dass die Filtereigenschaften denen der erstellten Ereignisse entsprechen. Die verwendeten Parameter des S7-OPC-Event-Servers für SIMATIC NET haben folgende Bedeutung:</p> <p><i>EventType</i> Es werden die Ereignistypen OPC_SIMPLE_EVENT und OPC_CONDITION_EVENT unterstützt.</p> <p><i>EventCategory</i> S7_PROCESS_ALARM</p> <p><i>LowSeverity</i> <i>HighSeverity</i></p> <p><i>Areas</i> S7-OPC-Event-Server unterstützt keine Areas.</p> <p><i>Sources</i> Sie können als Quelle eine Verbindungsnamen eingeben.</p>

## Ereignisse von OPCEventSubscription

Im Folgenden werden die Ereignisse für das Objekt OPCEventSubscription aufgeführt. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung des Ereignisses finden Sie in der folgenden OPC-Spezifikation:

Alarm & Events Automation Interface Standard  
Version 1.01  
December 15, 1999

Das Ereignis	bedeutet
ConditionEvent	Tritt auf, wenn Bedingungsereignisse vom Server gesendet werden.
RefreshCancel	Tritt auf, wenn die Methode <i>Refresh</i> unterbrochen ist. OPC-Event-Server für SIMATIC NET unterstützt <i>RefreshCancel</i> nicht.
RefreshComplete	Tritt auf, wenn die Methode <i>Refresh</i> abgeschlossen ist. OPC-Event-Server für SIMATIC NET unterstützt <i>RefreshComplete</i> nicht.
RefreshConditionEvent	Tritt auf, wenn Ereignisse über die Aktualisierung von Bedingungen vom Server gesendet werden.
SimpleEvent	Tritt auf, wenn eine Gruppe einfacher Ereignisse vom Server gesendet wird.
TrackingEvent	Tritt auf, wenn Protokollierereignisse vom Server gesendet werden. OPC-Event-Server für SIMATIC NET unterstützt <i>TrackingEvent</i> nicht.

## Collection-Objekt OPCAutoEventAreaBrowsers

Das Collection-Objekt OPCAutoEventAreaBrowsers ist eine Sammlung von Objekten OPCAutoEventAreaBrowser und den Methoden, um Sie zu erstellen, zu entfernen und zu verwalten.

---

### Hinweis

OPC-Event-Server unterstützt OPCAutoEventAreaBrowsers nicht.

---

## Objekt OPCAutoEventAreaBrowser

Mit dem Objekt OPCAutoEventAreaBrowser kann ein Client durch die Anlagenbereiche (Areas) und Quellen (Sources) des Servers browsen. Die Anlagenbereiche werden zu OPCEventAreas zusammengefasst, die Quellen zu OPCEventSources.

Die Anlagebereiche und Quellen können zum Filtern von Ereignissen verwendet werden.

## Namensraum des Servers

Server haben einen flachen oder hierarchischen Namensraum. Wenn der Namensraum flach ist, enthalten OPCEventAreas und OPCEventSources die gesamten Anlagebereiche und Quellen im Server.

Ist der Namensraum hierarchisch, können die Anlagenbereiche als Äste in einem Baum angesehen werden und die Quellen als Blätter. Das Objekt OPCAutoEventAreaBrowser bewegt sich wie ein Zeiger die Anlagenbereiche entlang. OPCEventAreas und OPCEventSources enthalten nur die Anlagenbereiche und Quellen, bei denen sich das Objekt befindet.

#### Hinweis

OPC-Event-Server unterstützt OPCAutoEventAreaBrowser nicht.

### Collection-Objekt OPCEvents

Das Collection-Objekt OPCEvents ist der Übergabeparameter der Methoden zur Ereignisbehandlung. Er enthält eine Sammlung von Objekten OPCEvent und den Methoden, um Sie zu erstellen, zu entfernen und zu verwalten.

OPCEvents wird aus den Ereignisbenachrichtigungen, die vom Server versendet werden, generiert und liefert damit die aufgetretenen Ereignisse.

### Eigenschaften von OPCEvents

Im Folgenden werden die Eigenschaften für das Objekt OPCEvents aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Eigenschaften finden Sie in der folgenden OPC-Spezifikation:

Alarm & Events Automation Interface Standard  
Version 1.01  
December 15, 1999

Die Eigenschaft	bedeutet
Count	Liefert die Anzahl der Einträge.
LastRefresh	Bestimmt, dass das Collection-Objekt OPCEvents das letzte in einer Reihe von Refresh-Benachrichtigungen ist.
Refresh	Bestimmt, dass das Collection-Objekt OPCEvents zu einer Refresh-Benachrichtigung gehört.

### Methoden von OPCEvents

Im Folgenden werden die Methoden für das Objekt OPCEvents aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Alarm & Events Automation Interface Standard  
Version 1.01  
December 15, 1999

Die Methode	bedeutet
Add	Legt ein neues Objekt OPCEvent an und fügt es zu der Collection hinzu.
Item	Bestimmt den Namen eines Eintrags.

## Objekt OPCEvent

Das Objekt OPCEvent enthält die Benachrichtigung für ein bestimmtes Ereignis.

### Eigenschaften von OPCEvent

Im Folgenden werden die Eigenschaften für das Objekt OPCEvent aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Eigenschaften finden Sie in der folgenden OPC-Spezifikation:

Alarm & Events Automation Interface Standard

Version 1.01

December 15, 1999

Die Eigenschaft	bedeutet
AckRequired	Zeigt, dass die Bedingung eine Bestätigung benötigt.
ActiveTime	Liefert den Zeitpunkt, an dem die Bedingung aktiviert wurde.
ActorID	Liefert die Protokollier- und Bedingungsereignisse, die eine Bestätigung benötigen.
ChangeAckState	Liefert die Ereignisbenachrichtigung für die Änderung der Eigenschaft <i>Acknowledge</i> einer Bedingung.
ChangeActiveState	Liefert die Ereignisbenachrichtigung für die Änderung der Eigenschaft <i>ActiveState</i> einer Bedingung.
ChangeAttribute	Liefert die Ereignisbenachrichtigung für die Änderung der Eigenschaft <i>Attribute</i> einer Bedingung.
ChangeEnableState	Liefert die Ereignisbenachrichtigung für die Änderung der Eigenschaft <i>Enable</i> einer Bedingung.
ChangeMessage	Liefert die Ereignisbenachrichtigung für die Änderung der Eigenschaft <i>Message</i> einer Bedingung. OPC-Event-Server für SIMATIC NET unterstützt <i>ChangeMessage</i> nicht.
ChangeQuality	Liefert die Ereignisbenachrichtigung für die Änderung der Eigenschaft <i>Quality</i> einer Bedingung. OPC-Event-Server für SIMATIC NET unterstützt <i>ChangeQuality</i> nicht.
ChangeSeverity	Liefert die Ereignisbenachrichtigung für die Änderung der Eigenschaft <i>Severity</i> einer Bedingung. OPC-Event-Server für SIMATIC NET unterstützt <i>ChangeSeverity</i> nicht.
ChangeSubCondition	Liefert die Ereignisbenachrichtigung für die Änderung der Eigenschaft <i>SubCondition</i> einer Bedingung.
ConditionAcknowledged	Liefert, dass der neue Status der Bedingung <i>Acknowledged</i> ist.
ConditionActive	Liefert, dass der neue Status der Bedingung <i>Active</i> ist.
ConditionName	Liefert den Namen der Bedingung, die sich auf die Ereignisbenachrichtigung bezieht.
Cookie	Liefert ein vom Server definiertes Cookie, das mit der Ereignisbenachrichtigung verbunden ist. OPC-Event-Server für SIMATIC NET unterstützt <i>Cookie</i> nicht.
EventCategory	Liefert den Code für die Ereigniskategorie des Ereignisses.
Message	Liefert einen Text, der die Benachrichtigung beschreibt.
OPCEventAttributes	Liefert die herstellerspezifische Ereignisattribute, die von der Ereignisbenachrichtigung zurückgegeben werden.

Die Eigenschaft	bedeutet
Quality	Liefert die Qualität, die mit dem Status der Bedingung verbunden ist. OPC-Event-Server für SIMATIC NET unterstützt <i>Quality</i> nicht.
Severity	Liefert die Schwere des Ereignisses.
Source	Liefert die Quelle der Ereignisbenachrichtigung.
SubConditionName	Liefert den Namen der aktuellen Unterbedingung. OPC-Event-Server für SIMATIC NET unterstützt <i>SubConditionName</i> nicht.
Time	Liefert den Zeitpunkt in UTC, an dem ein Ereignis stattgefunden hat.

### Objekt OPCEventCondition

Das Objekt OPCEventCondition beschreibt den aktuellen Zustand einer Bedingung.

### Collection-Objekt OPCEventSubConditions

Das Collection-Objekt OPCEventSubConditions ist eine Sammlung von Objekten OPCEventSubCondition. Die Sammlung repräsentiert die unterschiedlichen Stati einer Bedingung in einem OPC-Event-Server.

### Objekt OPCEventSubCondition

Das Objekt OPCEventSubCondition repräsentiert eine bestimmte Unterbedingung von OPC-Event-Server. OPCEventSubCondition beinhaltet die Attribute der Unterbedingung.

## 4.2 Custom-Schnittstelle programmieren

Die Custom-Schnittstelle ist so konzipiert, dass sie entsprechend den Anforderungen optimal funktioniert. Für den Zugriff durch Script-Sprachen ist sie nicht geeignet. Hierfür gibt es die Automation-Schnittstelle.

Es gibt sowohl eine Custom-Schnittstelle für den Zugriff auf Prozessvariablen (Data Access) als auch für die Verarbeitung von Ereignissen und Alarmen (Alarms & Events).

### Anwendung

Sie programmieren die Custom-Schnittstelle, wenn Sie mit C++ eine Anwendung mit hoher Variablenanzahl und hohem Datendurchsatz erstellen wollen.

### CLSID

Jede COM-Klasse kann durch einen Identifikationscode eindeutig identifiziert werden. Er ist 128 Bit lang und heißt CLSID. Durch die CLSID kann das Betriebssystem die "\*.dll"- oder "\*.exe"-Dateien finden, in denen die COM-Klasse implementiert ist. Wenn ein Client ein Objekt einer Klasse benutzen will, spricht er es über die CLSID an.

## ProgID

Zur vereinfachten Identifikation von OPC-Servern ist den CLSIDs eine lesbare ProgID zugeordnet. CLSID und ProgID werden durch den Hersteller des OPC-Servers festgelegt.

Für den OPC-Server von SIMATIC NET gilt für alle Protokolle:

Schnittstelle	ProgID
Data Access	OPC.SimaticNET (Standard-OPC-Server für Multiprotokollbetrieb) OPC.SimaticNET (Perfomanter OPC-Server für S7, SR, DP Einzelprotokollbetrieb. Symbolik und DX sind zusätzlich erlaubt) OPC.SimaticNET.DP (OPC-Server für hochperformanten DP-Zugriff)
Alarms & Events für S7	OPC.SimaticNetAlarms (A&E-Server für S7) OPC.SimaticNetAlarms (Perfomanter A&E-Server für S7 Einzelprotokollbetrieb, Symbolik und DX sind zusätzlich erlaubt) OPC.SimaticNetAlarmsSNMP (A&E-Server für SNMP)

## Vorgehensweise

Sie erzeugen ein COM-Objekt und rufen dessen Methoden auf.

---

### Hinweis

Ein Windows-Objekt ist eine Instanz einer COM-Klasse. Die COM-Klasse unterscheidet sich von der C++-Klasse. Eine Klasse in C++ ist eine Typ-Definition. Eine COM-Klasse ist eine Objektbeschreibung und enthält keine Typen.

---

### 4.2.1

## COM-Objekt erzeugen und Status des OPC-Servers abfragen

Eine detaillierte Beschreibung der Verwendung des Custom-Interface für OPC Data Access können Sie der Beschreibung der Beispiele entnehmen. Die folgende Beschreibung skizziert den grundsätzlichen Ablauf, ist jedoch in Bezug auf Fehlerbehandlung unvollständig.

Sie erzeugen ein COM-Objekt und fragen den Status in fünf Schritten ab.

Anschauliche Beispiele für das Erzeugen eines COM-Objekts in Visual C++ finden Sie im Kapitel "Beispielprogramme (Seite 633)".

#### 4.2.2 Objekte der Custom-Schnittstelle für Data Access

In diesem Abschnitt werden die Schnittstellen der Objekte mit Ihren Methoden aufgelistet. Es werden nur die SIMATIC NET spezifischen Besonderheiten beschrieben. Eine detaillierte Beschreibung der Schnittstellen finden Sie in den entsprechenden OPC-Spezifikationen.

##### Rückgabewerte der Methoden der Schnittstellen

Alle Methoden liefern ein Ergebnis vom Typ *HResult*.

###### 4.2.2.1 Objekt OPCServer

Ein Objekt der Klasse OPC-Server hat verschiedene Attribute, die Informationen beispielsweise über den Status oder die Version eines Objekts OPCServer enthalten. Weiterhin hat sie Methoden, mit denen ein Client die Objekte der Klasse OPC-Group verwaltet.

Eine Client-Anwendung wendet sich nur an das Objekt eines OPC-Servers direkt über COM-Mechanismen. Alle anderen Objekte werden durch entsprechende OPC-Methoden erzeugt.

##### Schnittstellen von OPCServer

Das Objekt OPCServer besitzt folgende Schnittstellen:

- IOPCServer
- IOPCBrowse (neu ab 3.00)
- IOPCItemIO (neu ab 3.00)
- IOPCBrowseServerAddressSpace (2.0)
- IOPCCommon
- IConnectionPointContainer
- IOPCItemProperties

Die OPC DA-Spezifikation, mit der die Schnittstelle unterstützt wird, steht in Klammern.

Das folgende Bild zeigt die Schnittstellen des Objekts OPCServer.

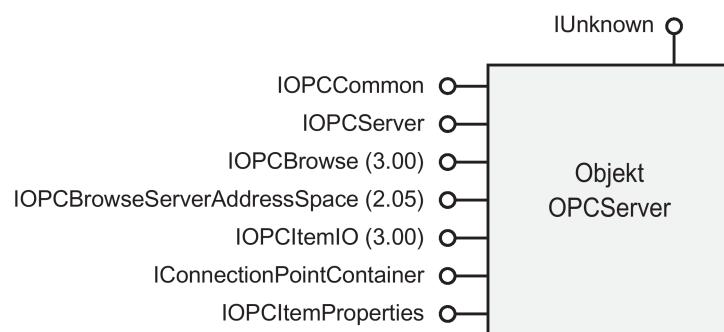


Bild 4-3 Das Objekt OPCServer

## Schnittstelle IOPCServer

Die Schnittstelle IOPCServer enthält Methoden zur Verwaltung der Objekte der Klasse OPC-Group. Weiterhin können Informationen über den aktuellen Status des Servers ermittelt werden.

Im Folgenden werden die Methoden für die Schnittstelle IOPCServer aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt.

Die Methode	bedeutet
AddGroup	<p>Erzeugt eine Gruppe im Server-Objekt.  Der Parameter <i>LCID</i> wird vom OPC-Server für SIMATIC NET nicht ausgewertet.  Der Parameter <i>pTimeBias</i> gibt die Zeitdifferenz zur lokalen Zeitzone an.  Die vom OPC-Server für SIMATIC NET verwendeten Aktualisierungsarten (Update Rate) sind Vielfache der bei der Projektierung festgelegten Zykluszeit. Die kleinste zulässige Aktualisierungsrate (Minimum Update Rate) entspricht der Zykluszeit.  Wenn <i>szName</i> leer ist, wird ein Name generiert, der mit einer Tilde beginnt (z. B. ~GROUP_1). Benutzerdefinierte Namen sollten deshalb nicht mit einer Tilde beginnen.  Der Parameter <i>pPercentDeadband</i> bewirkt eine eingeschränkte Änderungsmeldung an den OPC-Client, falls ein Bereich mit einer Obergrenze (EUHigh) und Untergrenze (EULow) projektiert wurde und der Itemwert sich um (<i>pPercentDeadband</i>/100) * (EU High - EU Low) ändert. Dies gilt nur für analoge Daten (EUType =1) und kanonische Typen VT_I2, I4, R4, R8, BOOL, UI1.</p>
CreateGroupEnumerator	<p>Erzeugt verschiedene Enumeratorn für die Gruppe.  OPC-Server für SIMATIC NET unterstützt keine Public Groups. Deshalb sind die Rückgabewerte der Methode bei Eingabewerten ...PRIVATE und ...PUBLIC (für Public Groups) für den Parameter <i>dwScope</i> identisch.</p>
GetErrorString	<p>Liefert für einen Fehlercode die entsprechende Fehlermeldung.  OPC-Server für SIMATIC NET unterstützt deutsche und englische Fehlermeldungen. Fehlermeldungen des Windows-Betriebssystems werden jedoch immer in der Installationssprache des Betriebssystems ausgegeben.</p>
GetGroupByName	<p>Liefert zum Namen einer privaten Gruppe einen zusätzlichen Schnittstellenzeiger, der Referenzzähler wird erhöht.</p>
GetStatus	<p>Liefert die Statusinformationen des Servers.  Der Rückgabewert ist der Name und die Version von OPC-Server.</p>
RemoveGroup	<p>Löscht eine Gruppe im Server.  OPC-Server für SIMATIC NET unterstützt die Verwendung von <i>bForce</i> nicht. Sie können keine Gruppen löschen, auf die noch Referenzen bestehen.</p>

Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Custom Interface Standard  
Version 3.00  
March 4, 2003

## Schnittstelle IOPCBrowse

Wesentlicher Vorteil dieser neuen Browse-Schnittstelle ist, dass ItemIDs und Properties bereits beim Durchsuchen des Namensraums ausgelesen werden können. Damit kann die Anzahl der Aufrufe zum OPCServer deutlich reduziert werden. Die Funktionalität entspricht der von OPC XML-DA.

Die Methode	bedeutet
Browse	Diese Methode durchsucht einen Zweig im Namensraum und ermittelt keinen oder mehrere Namensraumelemente. Der Namensraum sollte hierarchisch aufgebaut sein, eine flache Struktur wird dem OPC-Client ohne Unterelemente vermittelt. Ein hierachischer Namensraum weist Zweig- und Blatt-Unterelemente auf. Es können zusätzlich die ItemIDs und Properties ausgelesen werden.
GetProperties	Diese Methode liefert ein Feld von OPCItemProperties für jede ItemID.

Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Custom Interface Standard

Version 3.00

March 4, 2003

## Schnittstelle IOPCItemIO

Mit IOPCItemIO ist das Lesen / Schreiben von Items ohne Gruppen möglich. Dies ist insbesondere für Einzelauftrufe (Lesen von Konfigurationsvariablen) einfacher und effizienter.

Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Custom Interface Standard

Version 3.00

March 4, 2003

Die Methode	bedeutet
Read	Liest synchron einen oder mehrere Werte, Qualitäten und Zeitstempel. Dies Funktion ist mit der Methode IOPCSyncIO::Read vergleichbar.
WriteVQT	Schreibt synchron einen oder mehrere Werte. Das Schreiben von Qualität und Zeitstempel wird nicht unterstützt, dies wird mit der Meldung OPC_E_NOTSUPPORTED abgelehnt.

## Schnittstelle IOPCServerPublicGroups

OPC-Server für SIMATIC NET unterstützt keine Public Groups und so auch nicht die optionale Schnittstelle IOPCServerPublicGroups.

## Schnittstelle IOPCBrowseServerAddressSpace

Über die OPC-DA-2.0-Schnittstelle IOPCBrowseServerAddressSpace können Sie den Namensraum des Servers untersuchen. Der Namensraum enthält alle dem Server bekannten OPC-Items.

Im Folgenden werden die Methoden für die Schnittstelle IOPCBrowseServerAddressSpace aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Custom Interface Standard  
Version 2.05  
December 17, 2001

<b>Die Methode</b>	<b>bedeutet</b>
BrowseAccessPaths	Ermittelt den Access Path einer ItemID. BrowseAccessPaths wird bei OPC-Server für SIMATIC NET nicht benötigt.
BrowseOPCItemIDs	Liefert einen String des Typs <i>IEnumString</i> , dessen Inhalt durch die Parameter des Aufrufs festgelegt wird. Wird der Eingabeparameter <i>dwBrowseFilterType</i> auf OPC_BRANCH gesetzt, so sind die Parameter <i>vtDataTypeFilter</i> und <i>dwAccessRightsFilter</i> wirkungslos. Die Regeln zur Erzeugung eines Filters lauten: *: jede Folge von Zeichen, auch leere Strings +: jede Folge von Zeichen, mindestens jedoch ein Zeichen ?: genau ein beliebiges Zeichen //(eckige Klammern): genau ein Zeichen aus der angegebenen Menge Wenn Sie ein Filterzeichen verwenden, müssen Sie einen inversen Schrägstrich () voranstellen.
ChangeBrowsePosition	Wechselt im Namensraum in die übergeordnete Ebene oder in einen Ast.
GetItemID	Stellt eine vollständige ItemID im hierarchischen Namensraum zusammen. OPC-Server für SIMATIC NET unterstützt GetItemID nur für einzelne Blätter (LEAF).
QueryOrganization	Liefert die Struktur des Namensraums. Die Struktur des Namensraums von OPC-Server für SIMATIC NET ist hierarchisch aufgebaut.

## Schnittstelle IOPCItemProperties

Die Schnittstelle IOPCItemProperties enthält Methoden, um serverspezifische Informationen über ein Item abzufragen.

Im Folgenden werden die Methoden für die Schnittstelle IOPCItemProperties aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Custom Interface Standard  
Version 2.05  
December 17, 2001

<b>Die Methode</b>	<b>bedeutet</b>
QueryAvailableProperties	Liefert zu einem Item eine Liste der verfügbaren Eigenschaften.
GetPropertyIDs	Liefert die Werte der in einer Liste von PropertyIDs übergebenen Eigenschaften eines Items.
LookupItemIDs	Liefert eine Liste von ItemIDs zu einer Liste von PropertyIDs.

## Schnittstelle IConnectionPointContainer

Die Schnittstelle IConnectionPointContainer ist eine COM-Standardschnittstelle zur Meldung asynchroner Ereignisse über ConnectionPoints. Detaillierte Informationen über die Verwendung von Connection Points entnehmen Sie bitte der Literatur zu COM.

## Schnittstelle IOpcCommon

Die Schnittstelle IOpcCommon enthält Methoden, um dem Server Spracheinstellungen und den Namen des Client bekannt zu machen.

Im Folgenden werden die Methoden für die Schnittstelle IOpcCommon aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Custom Interface Standard

Version 2.05

December 17, 2001

Die Methode	bedeutet
SetLocaleID	Setzt den Sprachcode des Servers. Der OPC-Server für SIMATIC NET unterstützt Deutsch und Englisch.
GetLocaleID	Holt den Sprachcode des Servers. Der OPC-Server für SIMATIC NET unterstützt Deutsch und Englisch.
QueryAvailableLocaleIDs	Liefert alle verfügbaren Sprachcodes des Servers. Der OPC-Server für SIMATIC NET unterstützt Deutsch und Englisch.
GetErrorString	Liefert den entsprechenden Fehlercode zu einem Fehlercode.
SetClientName	Übergibt einen Beschreibungstext für den Client an den Server.

## Schnittstelle IPersistFile

Die optionale Schnittstelle *IPersistFile* wird von OPCServer für SIMATIC NET nicht unterstützt.

### 4.2.2.2 Objekt OPCGroup

Ein Objekt der Klasse OPC-Group verwaltet die einzelnen Prozessvariablen, die OPC-Items. Mit Hilfe von Objekten OPCGroup kann ein Client semantisch sinnvolle Einheiten von OPC-Items bilden und mit diesen Operationen durchführen.

## Schnittstellen von OPCGroup

Das Objekt OPCGroup besitzt folgende Schnittstellen:

- IOPCGroupStateMgt
- IOPCGroupStateMgt2 (neu ab 3.00)
- IOPCASyncIO2
- IOPCAAsyncIO3 (neu ab 3.00)
- IOPCItemMgt

- IOPCItemDeadbandMgt (neu ab 3.00)
- IOPCItemSamplingMgt (neu 3.00, optional)
- IConnectionPointContainer
- IOPCSyncIO
- IOPCSyncIO2 (neu ab 3.00)

Die OPC-DA-Spezifikation, mit der die Schnittstelle unterstützt wird, steht in Klammern.

Das folgende Bild zeigt die Schnittstellen von OPCGroup.

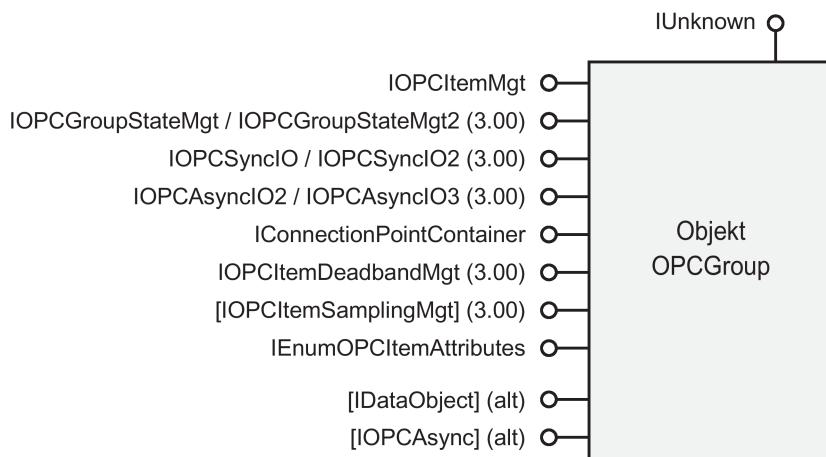


Bild 4-4 Das Objekt OPCGroup - [optionale Schnittstellen in Klammern]

### Schnittstelle IOPCGroupStateMgt

Die Schnittstelle IOPCGroupStateMgt bietet Methoden für die Verwaltung von Gruppen. Sie können gruppenspezifische Parameter bearbeiten und Gruppen kopieren.

Im Folgenden werden die Methoden für die Schnittstelle IOPCGroupStateMgt aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Custom Interface Standard

Version 2.05

December 17, 2001

Die Methode	bedeutet
CloneGroup	Erzeugt eine Kopie einer Gruppe. Die Gruppenattribute werden bis auf folgende Ausnahmen kopiert:  Der Aktiv-Status wird auf False gesetzt. Ein neues Server-Handle wird vergeben <i>szName</i> kann leer sein. In diesem Fall wird ein eindeutiger Name generiert.
GetState	Holt den Status der Gruppe.  Die Parameter <i>pTimeBias</i> und <i>pLCID</i> haben für OPC-Server für SIMATIC NET keine Bedeutung.

Die Methode	bedeutet
SetName	Ändert den Namen einer Gruppe
SetState	Ändert die Eigenschaften der Gruppe. Die Parameter <i>pTimeBias</i> und <i>pLCID</i> haben für OPC-Server für SIMATIC NET keine Bedeutung. Die vom OPC-Server für SIMATIC NET verwendeten Aktualisierungsraten (Update Rate) sind Vielfache der bei der Projektierung festgelegten Zykluszeit. Die kleinste zulässige Aktualisierungsr率e (Minimum Update Rate) entspricht der Zykluszeit.

## Schnittstelle IOPCGroupStateMgt2

Mit dieser Erweiterung von IOPCGroupStateMgt ab OPC DA 3.00 kann für jede OPC-Gruppe eine Überwachungszeit eingerichtet werden. Der OPC-Server schickt dann zyklisch ein Lebenszeichen. Dies können Daten sein oder ein leeres Callback (keep-alive), wenn sich die aktiven Items nicht geändert haben.

Die Methode	bedeutet
SetKeepAlive	Mit dieser Methode kann die Lebenszeichen-Überwachung des OPC-Server in Form eines zyklischen Rückrufs (Callback) eingestellt werden. Damit muss im OPC-Client keine GetStatus()-Überwachung verwendet werden.
GetKeepAlive	Liest den aktuellen Wert aus

Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Custom Interface Standard

Version 3.00

March 4, 2003

## Schnittstelle IOPCPublicGroupStateMgt

OPC-Server für SIMATIC NET unterstützt keine Public Groups. Aus diesem Grund hat die optionale Schnittstelle IOPCPublicGroupStateMgt keine Bedeutung.

## Schnittstelle IOPCAsyncIO2

Die Schnittstelle IOPCAsyncIO2 enthält Methoden für asynchrones Lesen und Schreiben von Items.

Asynchron bedeutet, dass der Client eine Lese- oder Schreiboperation anstößt und seine Ausführung fortsetzt. Die Schnittstelle verwendet Connection Points. Damit wird die Verarbeitung der übergebenen Daten vereinfacht.

Zu jedem gelesenen Wert liefert OPC einen Zeitstempel. Da die SIMATIC-Systeme keine Zeitstempel verwalten, wird der Zeitpunkt des Empfangs im Server als Zeitstempel verwendet.

Im Folgenden werden die Methoden für die Schnittstelle IOPCAsyncIO2 aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Custom Interface Standard  
Version 2.05  
December 17, 2001

<b>Die Methode</b>	<b>bedeutet</b>
Read	Setzt einen asynchronen Lesebefehl ab. Der Aufruf wird im Server zeitüberwacht. Bei Überschreiten der in der Projektierung eingestellten Fehlerwartezeit erfolgt eine Benachrichtigung mit dem Status E_ABORT.
Write	Setzt einen asynchronen Schreibbefehl ab. Der Aufruf wird im Server zeitüberwacht. Bei Überschreiten der in der Projektierung eingestellten Fehlerwartezeit erfolgt eine Benachrichtigung mit dem Status E_ABORT.
Cancel2	Bricht einen anstehenden Auftrag ab
Refresh2	Fordert für jedes aktive Item den aktuellen Wert aus dem Cache an
SetEnable	Ermöglicht die Deaktivierung der Benachrichtigung über <i>OnDataChange</i>
GetEnable	Liefert den aktuellen Wert für die Benachrichtigung über <i>OnDataChange</i>

### Schnittstelle IOPCAsyncIO3

Die Schnittstelle IOPCAsyncIO3 ergänzt IOPCAsyncIO2 ab OPC DA 3.00 um die Möglichkeit, mit MaxAge, Quality und Timestamp asynchron zu lesen oder zu schreiben. Der OPC-Server lehnt allerdings das Schreiben von Quality und Timestamp mit OPC\_E\_NOTSUPPORTED ab.

---

#### Hinweis

Beachten Sie, dass die beiden Schnittstellen *IOPCAsyncIO2* und *IOPCAsyncIO3* den gleichen *Connection Point*, gleiche Cookies und Callback-Funktionen benutzen. Richten Sie daher bei Verwendung beider Schnittstellen den Callback nur einmal ein.

---

<b>Die Methode</b>	<b>bedeutet</b>
ReadMaxAge	Liest asynchron mit dem Parameter MaxAge einen oder mehrere Werte, Qualitäten und Zeitstempel
WriteVQT	Schreibt asynchron einen oder mehrere Werte. Das Schreiben von Quality und Timestamp wird nicht unterstützt, es wird mit der Meldung OPC_E_NOTSUPPORTED abgelehnt.
RefreshMaxAge	Erzwingt die Rückruf-Funktion IOPCDataCallback::OnDataChange für alle aktiven Items unter Berücksichtigung von MaxAge

Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Custom Interface Standard  
Version 3.00  
March 4, 2003

## Schnittstelle IOPCAsyncIO

Diese Schnittstelle enthält Methoden für asynchrones Lesen und Schreiben von Items. Asynchron bedeutet, dass der Client eine Lese- oder Schreiboperation anstößt und seine Ausführung fortsetzt.

Asynchrone Operationen liefern eine Transaktions-ID. Wenn der Server die Lese- oder Schreiboperation abgeschlossen hat, wird der Client durch eine Meldung an seine IAdviseSink-Schnittstelle benachrichtigt (zur IAdviseSink-Schnittstelle des Client siehe Beschreibung der Schnittstelle *IDataObject*).

---

### Hinweis

In der Version 2 wurde die Schnittstelle *IOPCAsyncIO* durch *IOPCAsyncIO2* ersetzt.

*IOPCAsyncIO2* und die neuere *IOPCAsyncIO3* verwenden *Connection Points* und sind einfacher einzusetzen. Verwenden Sie in zukünftigen Projekten *IOPCAsyncIO2* oder *IOPCAsyncIO3*.

---

Zu jedem gelesenen Wert liefert OPC einen Zeitstempel. Da die SIMATIC-Systeme keine Zeitstempel verwalten, wird der Zeitpunkt des Empfangs im Server als Zeitstempel verwendet.

Im Folgenden werden die Methoden für die Schnittstelle IOPCAsyncIO aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Custom Interface Standard  
Version 2.05  
December 17, 2001

Die Methode	bedeutet
Cancel	Bricht einen anstehenden Auftrag ab.
Read	Setzt einen asynchronen Lesebefehl ab. Der Aufruf wird im Server zeitüberwacht. Der zugehörige Konfigurationsparameter ist <i>Read/Write Timeout</i> . Bei Abbruch der Zeitüberwachung erfolgt ein Callback mit dem Status <i>E_ABORT</i> .
Refresh	Fordert für jedes aktive OPC-Item einen aktuellen Wert.
Write	Setzt einen asynchronen Schreibbefehl ab. Der Aufruf wird im Server zeitüberwacht. Der zugehörige Konfigurationsparameter ist <i>Read/Write Timeout</i> . Bei Abbruch der Zeitüberwachung erfolgt ein Callback mit <i>hrStatus=E_ABORT</i> .

## Schnittstelle IOPCItemMgt

Die Schnittstelle IOPCItemMgt enthält Methoden, um mehrere Items in einer Gruppe zu verwalten.

Im Folgenden werden die Methoden für die Schnittstelle IOPCItemMgt aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Custom Interface Standard  
Version 2.05  
December 17, 2001

<b>Die Methode</b>	<b>bedeutet</b>
AddItems	Fügt zu einer Gruppe ein oder mehrere Items hinzu.
CreateEnumerator	Erzeugt einen Enumerator. Dabei handelt es sich um das Objekt EnumOPCItemAttributes.
RemoveItems	Löscht ein oder mehrere Items aus der Gruppe.
SetActiveState	Setzt den Aktivstatus eines oder mehrerer Items der Gruppe.
SetClientHandles	Setzt das Client-Handle eines oder mehrerer Items der Gruppe.
SetDataTypes	Setzt den geforderten Datentyp eines oder mehrerer Items der Gruppe.
ValidateItems	Prüft ein OPC-Item auf Gültigkeit.

### Schnittstelle IOPCItemDeadbandMgt

Mit dieser neuen Schnittstelle für OPC DA 3.00 ist es möglich, den Parameter PercentDeadband innerhalb einer Gruppe Item-spezifisch zu verändern. Der Gruppen-Parameter PercentDeadband wird für alle Items mit dem eingestellten Wert der Gruppe verwendet, oder die Items werden nun, falls mit dieser Methode eingestellt, mit Item-spezifischen PercentDeadband-Werten ausgewertet (Defaultwert = 0).

<b>Die Methode</b>	<b>bedeutet</b>
SetItemDeadband	Überschreibt den Gruppenwert Item-spezifisch
GetItemDeadband	Liest den aktuellen Item-spezifischen Wert
ClearItemDeadband	Setzt den Item-spezifischen Wert auf den Gruppenwert zurück

Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Custom Interface Standard  
Version 3.00  
March 4, 2003

### Schnittstelle IOPCItemSamplingMgt

Diese optionale Schnittstelle für OPC DA 3.00 wird unterstützt.

Der OPC-Client kann mit dieser Schnittstelle Item-spezifische Aktualisierungszeiten (RevisedSamplingRate) einstellen. Diese können größer oder kleiner der Gruppen-Aktualisierungszeit (RevisedUpdateRate) sein. Bei Gleichheit gibt es keine Änderung zum bisherigen Verhalten. Damit kann innerhalb einer OPC-Gruppe die Aktualisierungszeit einzelner Items feiner auf die tatsächliche Änderungsgeschwindigkeit der Items eingestellt werden.

Die Gruppen-Aktualisierungszeit zum Client wird durch Item-spezifische Werte nicht verändert.

Wenn die Item-spezifische Aktualisierungszeit kleiner als die Gruppen-Aktualisierungszeit ist, kann der OPC-Server mehrere Werte speichern. Auf Anforderung des OPC-Client können mehrere Item-Änderungen im Server gespeichert werden. Dieser Mechanismus wird

in OPC Data Access 3.00 als Pufferung (item buffering) bezeichnet. Der vorbelegte Default-Zustand für die Pufferung ist 0 = "AUS".

Die Methode	bedeutet
SetItemSamplingRate	Überschreibt die Gruppen-Aktualisierungszeit Item-spezifisch
GetItemSamplingRate	Liest den aktuellen Item-spezifischen Wert
ClearItemSamplingRate	Setzt den Item-spezifischen Wert auf den Gruppenwert zurück
SetItemBufferEnable	Aktiviert oder deaktiviert Item-spezifische Daten-Zwischenspeicherung für Items, die eine schnellere Änderungsgeschwindigkeit (sampling rate) als die Gruppen-Aktualisierungszeit haben.
GetItemBufferEnable	Liest den Zustand Item-spezifisch aus

Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Custom Interface Standard

Version 3.00

March 4, 2003

## Schnittstelle IOPCSyncIO

Die Schnittstelle IOPCSyncIO enthält Methoden für synchrones Lesen und Schreiben. Synchron bedeutet, dass der Client wartet, bis eine Lese- oder Schreiboperation abgeschlossen ist, bevor er weiterarbeitet. Der Client kann dann das Ergebnis der Lese- oder Schreiboperation für die weitere Bearbeitung verwenden.

Da OPC-Server für SIMATIC NET für jeden Client einen eigenen Thread startet, werden andere Clients nicht blockiert, während der Client wartet.

Im Folgenden werden die Methoden für die Schnittstelle IOPCSyncIO aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Custom Interface Standard

Version 2.05

December 17, 2001

Die Methode	bedeutet
Read	Liest die Werte, Statusinformationen und die Zeitstempel eines oder mehrerer Items einer Gruppe. Der Aufruf wird im Server zeitüberwacht. Die entsprechende Fehlerwartezzeit wird protokollspezifisch in der Projektierung eingestellt.
Write	Schreibt den Wert für ein oder mehrere Items einer Gruppe. Der Aufruf wird im Server zeitüberwacht. Die entsprechende Fehlerwartezzeit wird protokollspezifisch in der Projektierung eingestellt.

## Schnittstelle IOPCSyncIO2

Die Schnittstelle IOPCSyncIO2 ergänzt IOPCSyncIO ab OPC DA 3.00 um die Möglichkeit, mit MaxAge, Quality und Timestamp synchron zu lesen oder zu schreiben. Der OPC-Server lehnt allerdings das Schreiben von Quality und Timestamp mit OPC\_E\_NOTSUPPORTED ab.

Die Methode	bedeutet
ReadMaxAge	Liest synchron mit dem Parameter MaxAge einen oder mehrere Werte, Qualitäten und Zeitstempel
WriteVQT	Schreibt synchron einen oder mehrere Werte. Das Schreiben von Qualität und Zeitstempel wird nicht unterstützt, es wird mit der Meldung OPC_E_NOTSUPPORTED abgelehnt.

Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Custom Interface Standard  
Version 3.00  
March 4, 2003

## Schnittstelle IDataObject

Die Schnittstelle IDataObject ist eine Standardschnittstelle von COM zur Datenübertragung. Sie enthält Methoden zum Aufbau einer Benachrichtigungsverbindung zwischen dem Client und einer Server-Gruppe.

Wenn der Server einem Client eine Benachrichtigung schickt, spricht er den Client über die Client-Schnittstelle IAdviseSink an. Der Server ruft dazu die Methode *OnDataChange* von IAdviseSink auf.

Die folgende Abbildung zeigt die Zusammenarbeit der Schnittstellen IAdviseSink im Client und IDataObject im Server.

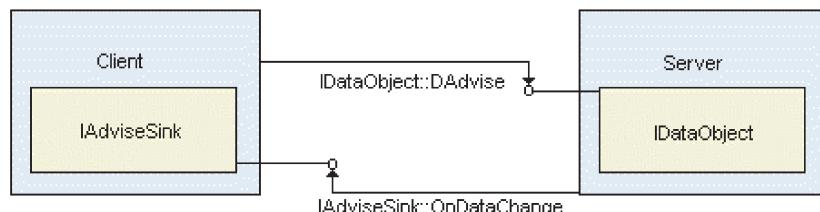


Bild 4-5      Zusammenarbeit von *IAdviseSink* und *IDataObject*

---

### Hinweis

*IDataObject* wurde in Version 1 der Data Access Schnittstelle für die asynchrone Kommunikation verwendet. Ab Version 2 kommen *Connection Points* zum Einsatz, was einfacher und flexibler ist.

Verwenden Sie in zukünftigen Projekten Version 2.

Im Folgenden werden die Methoden für die Schnittstelle IDataObject aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Custom Interface Standard

Version 2.05

December 17, 2001

Die Methode	bedeutet
DAdvise	Baut eine Verbindung zwischen Server und Client auf
DUnadvise	Baut eine Verbindung zwischen Server und Client ab

### Schnittstelle IEnumOPCItemAttributes

Die Schnittstelle IEnumOPCItemAttributes basiert auf der Standardschnittstelle IEnum. Sie liefert die Items einer Gruppe zurück. Die Schnittstelle wird durch die Methode *CreateEnumerator* der Schnittstelle *IOPCItemMgt* geliefert. Sie ist nicht durch *QueryInterface* erreichbar.

Im Folgenden werden die Methoden für die Schnittstelle IEnumOPCItemAttributes aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Data Access Custom Interface Standard

Version 2.05

December 17, 2001

Die Methode	bedeutet
Clone	Erzeugt eine identische Kopie des Objekts IEnumOPCItemAttributes
Next	Holt das nächste OPC-Item der Gruppe
Reset	Setzt die Auflistung auf das erste Item der Gruppe zurück
Skip	Überspringt eine Anzahl von Items in der Aufzählung

### 4.2.3

### Objekte der Custom-Schnittstelle für Alarms & Events

Im Folgenden werden die Schnittstellen mit Ihren Methoden aufgelistet. Es werden nur die SIMATIC NET-spezifischen Besonderheiten beschrieben. Eine detaillierte Beschreibung der Schnittstellen finden Sie in den entsprechenden OPC-Spezifikationen.

#### Rückgabewerte der Methoden der Schnittstellen

Alle Methoden liefern ein Ergebnis vom Typ *HResult*.

##### 4.2.3.1

### Objekt OPCEventServer

Ein Objekt der Klasse OPC-Event-Server wird vom Client erzeugt. Der Client wendet sich dann an dieses Objekt, um die Dienste von Alarms & Events in Anspruch zu nehmen.

Über das Objekt OPCEventServer können folgende Aufgaben durchgeführt werden:

- Meldeobjekt erzeugen
- Abfragen durchführen

- Ereignisse aktivieren
- Landessprache von Ausgabetexten setzen
- Empfang von serverspezifischen Ereignissen anmelden

### Schnittstellen von OPCEventServer

Das folgende Bild zeigt die Schnittstellen von OPCEventServer.

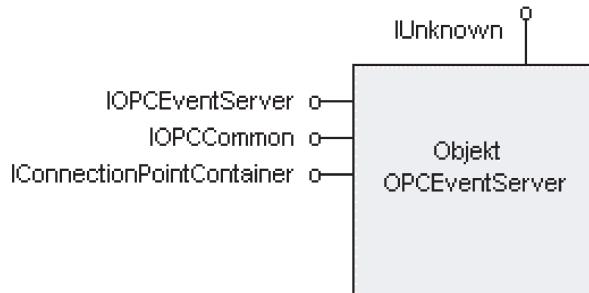


Bild 4-6 Objekt OPCEventServer

### Schnittstelle IOPCCommon

Die Schnittstelle IOPCCommon enthält Methoden, um dem Server Spracheinstellungen und den Namen des Client bekannt zu machen.

Im Folgenden werden die Methoden für die Schnittstelle IOPCCommon aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Alarms & Events Custom Interface Standard

Version 1.10

October 2, 2002

Die Methode	bedeutet
SetLocaleID	Setzt die Sprachcodes des Servers. OPC-Server für SIMATIC NET unterstützt Deutsch und Englisch.
GetLocaleID	Holt die Sprachcodes des Servers. OPC-Server für SIMATIC NET unterstützt Deutsch und Englisch.
QueryAvailableLocaleIDs	Liefert alle verfügbaren Sprachcodes des Servers. OPC-Server für SIMATIC NET unterstützt Deutsch und Englisch.
GetErrorString	Liefert für einen Fehlercode die entsprechende Fehlermeldung
SetClientName	Übergibt einen Beschreibungstext des Client an den Server

## IOPCEventServer

Die Schnittstelle IOPCEventServer ist die zentrale Schnittstelle für Alarms & Events. Sie hat folgende Aufgaben:

- Subscription-Objekte erzeugen
- Area-Browser erzeugen
- Ereigniskategorien untersuchen
- Bedingungen verwalten

### Hinweis

SIMATIC NET unterstützt nicht die optionalen Areas, deshalb wird auch kein Area-Browser erzeugt.

Im Folgenden werden die Methoden für die Schnittstelle IOPCEventServer aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Alarms & Events Custom Interface Standard  
Version 1.10  
October 2, 2002

Die Methode	bedeutet
GetStatus	Liefert aktuelle Statusinformationen zu OPC-Server.
CreateEventSubscription	Erstellt ein Meldeobjekt zur Benachrichtigung eines Client. Das Meldeobjekt heißt Subscription und entspricht einem Abonnement auf eine Menge von Ereignissen.  Es wird die verlangte Schnittstelle zum Zugriff auf das Meldeobjekt zurückgeliefert.
QueryAvailableFilters	Liefert Informationen über vom Event-Server unterstützte Filtermöglichkeiten.  OPC Event Server für SIMATIC NET unterstützt folgende Filter: OPC_FILTER_BY_EVENTS: 0x01 OPC_FILTER_BY_CATEGORY: 0x02 OPC_FILTER_BY_SEVERITY: 0x04 OPC_FILTER_BY_SOURCE: 0x16
QueryEventCategories	Liefert die vom Event-Server angebotenen Ereigniskategorien.  OPC-Event-Server für SIMATIC NET liefert nur Ereigniskategorien, wenn der Parameter <i>dwEventType</i> den Wert OPC_SIMPLE_EVENT oder OPC_CONDITION_EVENT hat.
QueryConditionNames	Liefert die vom Event-Server für eine bestimmte Ereigniskategorie angebotenen Bedingungen.
QuerySubConditionNames	Liefert die vom Event-Server für eine bestimmte Ereigniskategorie angebotenen Unterbedingungen.
QuerySourceConditions	Liefert die vom OPC-Event-Server für eine bestimmte Ereignisquelle (Source) angebotenen Bedingungen (Conditions).

<b>Die Methode</b>	<b>bedeutet</b>
QueryEventAttributes	Liefert die vom Event-Server für eine bestimmte Ereigniskategorie angebotenen Attribute. Der OPC-Event-Server für SIMATIC NET stellt besondere Attribute bereit. Die Attribute können nicht als ItemIDs bei Data Access verwendet werden.
TranslateToItemIDs	Ermittelt die einem Ereignisattribut entsprechenden OPCItems zur Verwendung mit einem assoziierten OPC-Data-Access-Server. OPC-Event-Server für SIMATIC NET unterstützt <i>TranslateToItemIDs</i> nicht.
GetConditionState	Liefert Informationen über den Zustand einer Bedingung einer Ereignisquelle.
EnableConditionByArea	Aktiviert alle Bedingungen für alle Ereignisquellen innerhalb des angegebenen Bereichs (Area).
EnableConditionBySource	Aktiviert alle Bedingungen für alle angegebenen Ereignisquellen.
DisableConditionByArea	Deaktiviert alle Bedingungen für alle Ereignisquellen innerhalb des angegebenen Bereichs (Area). OPC-Event-Server für SIMATIC NET unterstützt keine Bedingungen. Es wird E_NOTIMPL zurückgeliefert.
DisableConditionBySource	Deaktiviert alle Bedingungen für alle angegebenen Ereignisquellen.
AckCondition	Übergibt eine Ereignisbestätigung an den Client. Es können nur bedingte Ereignisse bestätigt werden.
CreateAreaBrowser	Erstellt ein Objekt OPCEventAreaBrowser zur Untersuchung des Prozessraums. OPC-Event-Server für SIMATIC NET unterstützt keine Areas. Es wird E_NOTIMPL zurückgeliefert.

## IConnectionPointContainer

Die Schnittstelle IConnectionPointContainer ist eine COM-Standardschnittstelle zur Meldung asynchroner Ereignisse über Connection Points. Detaillierte Informationen über die Verwendung von Connection Points entnehmen Sie bitte der Literatur zu COM.

### 4.2.3.2 Objekt OPCEventSubscription

Ein Objekt der Klasse OPC-Event-Subscription liefert Ereignismeldungen an den Client, der die Schnittstelle IConnectionPointContainer dieses Objekts verwendet.

Ein Client kann mehrere Objekte OPCEventSubscription benutzen. Er kann für die verschiedenen Objekte unterschiedliche Filter definieren.

Ein Objekt OPCEventSubscription entspricht einem Abonnement für definierte Ereignisse.

## Schnittstellen von OPCEventSubscription

Das folgende Bild zeigt die Schnittstellen von OPCEventSubscription.

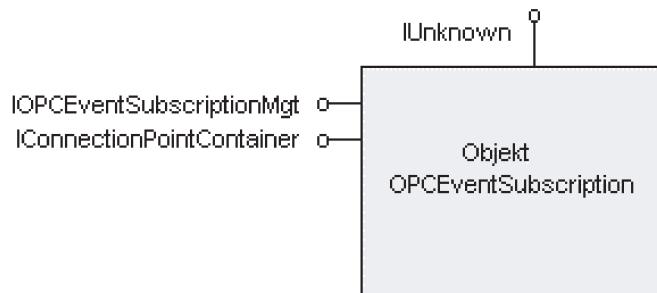


Bild 4-7 Objekt OPCEventSubscription

### Schnittstelle IOPCEventSubscriptionMgt

Die Schnittstelle IOPCEventSubscriptionMgt ist die zentrale Schnittstelle zur Verwaltung der Informationen über ein bestimmtes Ereignis-Abonnement. Über die Schnittstelle können beispielsweise die für den Client relevanten Ereignisse ausgewählt werden.

Im Folgenden werden die Methoden für die Schnittstelle IOPCEventSubscriptionMgt aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Alarms & Events Custom Interface Standard  
Version 1.10  
October 2, 2002

Die Methode	bedeutet
SetFilter	<p>Stellt den Filter zur Auswahl bestimmter Ereignisse für dieses Ereignis-Abonnement ein. Die Filterparameter haben für OPC-Event-Server für SIMATIC NET folgende Bedeutung:</p> <p><i>Event Typ</i> OPC-Event-Server unterstützt die Event Typen <b>OPC_SIMPLE_EVENT</b> und <b>OPC_CONDITION_EVENT</b>.</p> <p><i>Event Categories</i> Eine Beschreibung der Ereigniskategorien finden Sie in Kapitel "Event-Server für S7-Kommunikation (Seite 391)".</p> <p><i>Severity</i> Die Severity kann über die STEP 7 oder SIMATIC NCM PC/S7 konfiguriert werden. Der Standardwert z. B. für <b>S7_PROCESS_ALARM</b> ist 500.</p> <p><i>Areas</i> OPC-Event-Server für SIMATIC NET unterstützt keine Areas.</p> <p><i>Source</i> Sie können einen Verbindungsnamen angeben.</p>
GetFilter	Liefert die aktuell verwendeten Filter des Ereignis-Abonnements zurück. Siehe <i>SetFilter</i> .

Die Methode	bedeutet
SelectReturnedAttributes	Legt die Attribute fest, die mit einer Ereignismeldung für eine Ereigniskategorie geliefert werden.
GetReturnedAttributes	Gibt die Liste der Attribute zurück, die mit einer Ereignismeldung für eine Ereigniskategorie geliefert werden.
Refresh	Sendet alle aktiven und alle inaktiven, nicht bestätigten Bedingungsmeldungen, die der aktuellen Filtereinstellung entsprechen, an den Client.
CancelRefresh	Unterbricht einen laufenden Refresh-Vorgang. Da bei <i>Refresh</i> keine Ereignismeldungen gesendet werden, hat auch <i>Cancel/Refresh</i> keine Auswirkungen.
GetState	Liefert den aktuellen Zustand des Ereignis-Abonnements zurück.
SetState	Stellt verschiedene Eigenschaften eines Ereignis-Abonnements ein.

### Schnittstelle IConnectionPointContainer

Die Schnittstelle IConnectionPointContainer ist eine COM-Standardschnittstelle zur Meldung asynchroner Ereignisse über ConnectionPoints. Detaillierte Informationen über die Verwendung von Connection Points entnehmen Sie bitte der Literatur zu COM.

### Siehe auch

Objekte der Automation-Schnittstelle für Data Access (Seite 433)

#### 4.2.3.3 Objekt OPCEventAreaBrowser

Da OPC-Event-Server keine Bereiche (Areas) unterstützt, kann auch das Objekt OPCEventAreaBrowser nicht verwendet werden.

#### 4.2.4 Schnittstellen des Client für Alarms und Events

##### Übermittlung von Ereignissen

Ereignisse werden über Connection Points an einen Client übermittelt. Dazu muss der Client ein COM-Objekt mit den Schnittstellen IUnknown und der aufrufspezifischen Schnittstelle IOPCEventSink zum Empfang der Aufrufe bereitstellen. Bei der Anmeldung an einen Connection Point übergibt der Client einen Zeiger auf die IUnknown Schnittstelle an den Server.

### Schnittstellen der Client-Objekts

Das Objekt, das der Client zum Empfang von Nachrichten bereitstellt, muss wie folgt aufgebaut sein:

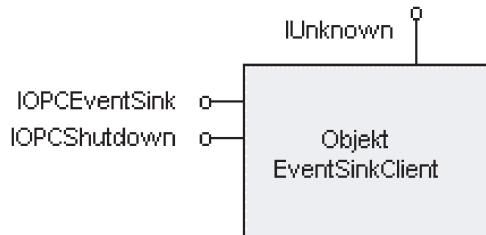


Bild 4-8 Schnittstellen des Client

#### 4.2.4.1 Schnittstelle IOPCEventSink

Die Schnittstelle IOPCEventSink ist die zentrale Schnittstelle des Client zum Empfang von Nachrichten. Sie enthält eine Methode, die vom Server zur Übergabe von Ereignissen aufgerufen wird.

Im Folgenden werden die Methoden für die Schnittstelle IOPCEventSink aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Alarms & Events Custom Interface Standard  
Version 1.10  
October 2, 2002

<b>Die Methode</b>	<b>bedeutet</b>
OnEvent	<p>Übergibt eine oder mehrere Ereignismeldungen an den Client.          Die Struktur <i>*pEvents</i> enthält ein oder mehrere Ereignisse. In folgende Strukturelemente trägt der OPC-Event-Server spezifische Werte ein:</p> <p><i>SzSource</i>  <i>Verbindungsinformation.</i>          Die Verbindungsinformation kann nicht über TranslateToItemID in eine ItemID umgewandelt werden und dann von Data Access verwendet werden.</p> <p><i>ftTime</i>          Entstehungszeitpunkt des Ereignisses im Partnergerät. Der Empfangszeitpunkt der Nachricht im OPC-Event-Server ist im Attribut EVENT_ATTR_S7_PCTIME abgelegt.</p> <p><i>szMessage</i>  <i>ALARM-Meldungsnummer</i></p> <p><i>dwEventType</i>          OPC_SIMPLE_EVENT</p> <p><i>dwEventCategory</i>          S7_PROCESS_ALARM</p> <p><i>pEventAttributes</i>          Diese Struktur enthält die mit dem Ereignis gelieferten Attribute. Die Attribute enthalten auch die vom Partnergerät gelieferten Begleitwerte der Meldung.</p> <p>Alle weiteren Strukturelemente sind nicht relevant.</p>

#### 4.2.4.2 Schnittstelle IOPCShutdown

Über diese Connection Point basierte Aufrufsschnittstelle kann der Server die Clients informieren, bevor er herunterfahren wird oder sich selbst abschaltet. So können dann gegebenenfalls die Clients reagieren und sich beenden.

Im Folgenden werden die Methoden für die Schnittstelle IOPCShutdown aufgelistet. Es werden nur die Besonderheiten für SIMATIC NET aufgeführt. Eine Beschreibung der Methoden finden Sie in der folgenden OPC-Spezifikation:

Alarms & Events Custom Interface Standard  
Version 1.10  
October 2, 2002

<b>Die Methode</b>	<b>bedeutet</b>
ShutdownRequest	<p>Der Server benachrichtigt seine Clients, dass der Server beendet wird.          Den Text für die Begründung für das Herunterfahren (<i>szReason</i>) können Sie über das Konfigurationsprogramm "Kommunikations-Einstellungen" setzen.</p>

#### 4.2.5 Fehlermeldungen für OPC-DA-Prozessvariablen

##### Protokollunabhängige Fehlercodes

Fehlercode	Bedeutung
OPC_E_INVALIDHANDLE (0xC0040001L)	Der Wert des Handles ist ungültig.
IDS_DUPLICATE (0xC0040002L)	Es wurde derselbe Wert mehrfach übergeben.
IDS_UNKNOWNLCID (0xC0040003L)	Die spezifizierte <i>locale ID</i> wird vom Server nicht unterstützt.
OPC_E_BADTYPE (0xC0040004L)	Eine Konvertierung zwischen dem canonicalDatatype und dem requestedDatatype wird vom Server nicht unterstützt.
OPC_E_PUBLIC (0xC0040005L)	Die gewünschte Operation kann für eine <i>public group</i> nicht durchgeführt werden.
OPC_E_BADRIGHTS (0xC0040006L)	Die gewünschte Operation (Lesen bzw. Schreiben) ist durch die Zugriffsrechte des Items untersagt.
OPC_E_UNKNOWNITEMID (0xC0040007L)	Der Name (item definition) ist im Namensraum des Servers nicht vorhanden.
OPC_E_INVALIDITEMID (0xC0040008L)	Der Name (item definition) hat eine unzulässige Syntax.
OPC_E_INVALIDFILTER (0xC0040009L)	Die Zeichenfolge des Filters ist nicht zulässig.
OPC_E_UNKNOWNPATH (0xC004000AL)	Der als AccessPath spezifizierte Verbindungsname ist nicht zulässig.
OPC_E_RANGE (0xC004000BL)	Der Wert ist außerhalb des zulässigen Wertebereichs.
OPC_S_UNSUPPORTEDRATE (0x0004000DL)	Die gewünschte Update-Frequenz (updateRate) wird vom Server nicht unterstützt. Es wird der nächste zulässige Wert benutzt.
OPC_S_CLAMP (0x0004000EL)	Der im Write übergebene Wert wurde akzeptiert, aber dessen Ausgabe wurde abgeschnitten.
OPC_S_INUSE (0x0004000FL)	Die Operation kann nicht vollständig ausgeführt werden, da noch Referenzen für das Objekt bestehen.
OPC_E_NOTFOUND (0xC0040011L)	Die Methode GetErrorString liefert einen String der Form dwError=%lx zurück, wobei %lx eine hexadezimale Fehlerkennung ist, die der OPC-Server nicht kennt.
0xC0048003L 0x00048003L	Es ist ein Timeout aufgetreten, z.B. auf Grund einer unterbrochenen Verbindung.
0xC0048004L 0x00048004L	Ein intern benutzerter Dienst ist beendet.
0xC0048005L 0x00048005L	Die gewünschte Operation (Lesen bzw Schreiben) ist durch die Zugriffsrechte des Item's untersagt.
0xC0048006L 0x00048006L	Es ist ein Fehler in der Kommunikation aufgetreten. Überprüfen Sie den Kommunikationspartner und die Kabelverbindung.
0xC0048007L 0x00048007L	Der Wert ist oberhalb des zulässigen Wertebereichs.

Fehlercode	Bedeutung
0xC0048008L 0x00048008L	Der Wert ist unterhalb des zulässigen Wertebereichs.
0xC0048009L 0x00048009L	Ein Fehler bei der Konvertierung ist aufgetreten.
0x85270181L 0x05270181L	Der Callback-Puffer ist übergelaufen.
CONNECT_E_NOCONNECTION (0x80040200)	Keine Verbindung. Die Meldung bezieht sich auf eine interne COM-Verbindung des OPC-Servers und nicht um eine S7- oder Transportverbindung.
CONNECT_E_CANNOTCONNECT (0x80040202)	Es kann keine Verbindung aufgebaut werden. Prüfen Sie die Anzahl der Async Connection Points (es ist nur 1 Async Connection Point möglich).

### Fehlercodes für das DP-Protokoll

Fehlercode	Bedeutung
0x85270101L 0x05270101L	Der DP-Master ist nicht betriebsbereit.
0x85270102L 0x05270102L	Der DP-Slave ist nicht betriebsbereit.
0x05270165L	Der DP-Master ist im Zustand CLEAR oder AUTOCLEAR.

### Fehlercodes für das S7-Protokoll

Fehlercode	Bedeutung
0x85270201L 0x05270201L	Blockdienste: Die r_id ist ungültig.
0x85270202L 0x05270202L	Domaindienste: Die Zugriffsrechte sind nicht gültig.
0x85270203L 0x05270203L	Domaindienste: Es ist ein Blockfehler aufgetreten.
0x85270204L 0x05270204L	Domaindienste: Es ist ein Dateifehler aufgetreten.
0x85270205L 0x05270205L	Die Konsistenzprüfung für das SIMOTION-Projekt ist fehlgeschlagen.
0x85270206L 0x05270206L	Es besteht keine Verbindung zum S7-Kommunikationspartner, die sofortige Fehlerrückgabe wurde projektiert.

## Fehlercodes für das PROFINET-Protokoll

Fehlercode	Bedeutung
0x85270601L 0x05270601L	Der angeforderte Wert ist nicht persistent.
0x85270602L 0x05270602L	Die Verbindung ist nicht aufgebaut.
0x85270603L 0x05270603L	Keine Verbindung
0x85270605L 0x05270605L	Der Wert ist unsicher.
0x85270606L 0x05270606L	Ungültige Beschreibung (zu lang, unerlaubte Zeichen, keine Trennzeichen, Syntaxfehler).
0x8527060aL 0x0527060aL	Ungültige Aufzählung.
0x8527060bL 0x0527060bL	Ungültige ID.
0x8527060cL 0x0527060cL	Ungültiger Epsilon-Typ oder Wert.
0x8527060dL 0x0527060dL	Der Ersatzwert ist ungültig.
0x8527060eL 0x0527060eL	Unerlaubte Verbindung mit sich selbst.
0x8527060fL 0x0527060fL	Ungültiger Cookie-Wert.
0x85270610L 0x05270610L	Nicht unterstützte Zeit.
0x85270611L 0x05270611L	Nicht unterstützter QoS-Typ.
0x85270614L 0x05270614L	Ziel bereits verbunden.
0x85270604L 0x05270604L	Der Wert wurde nur gepuffert.
0x85270607L 0x05270607L	Unbekanntes Objekt.
0x85270608L 0x05270608L	Unbekannte Eigenschaft.
0x85270609L 0x05270609L	Der zurückgegebene Typ stimmt nicht mit dem erwarteten Typ überein.
0x85270612L 0x05270612L	Der QoS-Wert (Quality of Service) wird nicht unterstützt.
0x85270613L 0x05270613L	Es erfolgt gerade ein Speichervorgang, deshalb ist eine Änderung der Konfiguration nicht erlaubt.
0x85270615L 0x05270615L	Aktion ist momentan nicht anwendbar.
0x85270616L 0x05270616L	Zugriff verweigert.
0x85270617L 0x05270617L	Es wurde ein Hardware-Defekt festgestellt.

Fehlercode	Bedeutung
0x85270A01L 0x05270A01L	Die Adressangabe kann nicht aufgelöst werden.
0x85270A02L 0x05270A02L	Die Datenlänge ist ungültig.
0x05270A64L	Der Provider- bzw. Consumerstatus wurde gepuffert, aber noch nicht zum Device übertragen.
0x05270A65L	Der Ausgangswert vom Controller wurde gepuffert, jedoch noch nicht zum Device übertragen (z.B. wegen einer Störung des Device).

### Fehlercodes für das SEND/RECEIVE-Protokoll

Fehlercode	Bedeutung
0x85270301L 0x05270301L	Es besteht keine Verbindung zum SR-Kommunikationspartner, die sofortige Fehlerrückgabe wurde projektiert.

### Windows-Fehlercodes

Fehlercode	Bedeutung
0x80070005	Zugriff verweigert.
0x80070057	Falscher Parameter.

---

#### Hinweis

Rückmeldungen, deren erstes Byte gleich Null ist (0x0....), sind eingeschränkte Erfolgsmeldungen. Der Anwender kann dann annehmen, dass der Variablenwert, die Qualität und der Zeitstempel wohl definiert sind, obwohl die Qualität "Schlecht" oder "Unsicher" ist.

---

## 4.3 XML-Schnittstelle programmieren

---

#### Hinweis

Die OPC XML-Schnittstelle ist in der SIMATIC NET Software ab Version 6.1 verfügbar.

---

### Einleitung

Dieses Kapitel beschreibt die Methoden der XML-DA-Schnittstelle. Es stellt die Tagstruktur der Methoden graphisch dar und beschreibt die einzelnen Elemente und Attribute.

Zuerst wird die Syntax der Methodenbeschreibung dargestellt. Anschließend werden die sogenannten Basis-Schemas behandelt. Das sind Elemente, die in der Schema-Definition von OPC XML-DA mehrfach vorkommen. Dann wird auf die einzelnen Methoden eingegangen.

### **XML-Schema-Definition der Datentypen**

In der Schema Section der WSDL von OPC XML-DA sind - konform zur XMLSchema2001 Definition - die komplexen Datentypen beschrieben, die als Parameter der einzelnen Methoden verwendet werden. Die genaue Beschreibung ist in der OPC XML-DA Spezifikation nachzulesen.

Im Code des Web-Dienstes wird für jeden komplexen Datentyp aus dem XML Schema eine C# Klasse angelegt, die Methoden des Web-Dienstes sind in einem Interface beschrieben.

### **Hinweise zur Nutzung der XML-DA-Schnittstelle**

---

#### **Hinweis**

Zu produktspezifischen Fehlermeldungen wird bei OPC XML-DA kein Fehlertext, sondern nur die Fehlernummer zurückgegeben. Lesen Sie den Fehlertext dazu im Kapitel "Fehlermeldungen für OPC-DA-Prozessvariablen (Seite 474)" nach.

---

---

#### **Hinweis**

Schalten Sie für den Betrieb von OPC-XML-DA die remote OPC-Kommunikation im Konfigurationsprogramm "Kommunikations-Einstellungen" bei "Sicherheit" frei.

---

---

#### **Hinweis**

Der OPC XML-DA-Web-Dienst verwendet weitere Dienste für die Nutzung der Symbolik. Diese Dienste benötigen ausreichende Rechte für den Zugriff auf die Verzeichnisse, in denen die STI- oder ATI-Symboldateien abgelegt werden sowie für die Dateien selbst. Bitte stellen Sie deshalb sicher, dass die folgenden Benutzer die Berechtigung "Vollzugriff" auf diese Dateien und Verzeichnisse haben:

**Betriebssystem:** Windows 7 oder Windows 8

**Benutzer im deutschen Betriebssystem:** Benutzer "NETZWERKDIENST"

**Benutzer im englischen Betriebssystem:** Benutzer "NETWORK SERVICE"

---

---

### Hinweis

Der OPC XML-DA-Web-Dienst beauftragt folgende weitere Dienste für die Erstellung der Trace-Dateien:

- unter Windows 7 und Windows 8 die Netzwerkdienste

Bitte geben Sie diesen Diensten Lese- und Schreibrechte für die Verzeichnisse, in denen die Trace-Dateien abgelegt werden.

---

---

### Hinweis

Der Internet-Information-Server kann so konfiguriert werden, dass er bei Eintreten bestimmter Ereignisse automatisch zurückgesetzt wird.

Unter Windows 7 und Windows 8 ist das automatische Zurücksetzen nach jeweils 29 Stunden Betriebszeit voreingestellt. Bitte beachten Sie:

- Durch diese Einstellungen können Antwortverzögerungen entstehen.
  - Nach jedem Zurücksetzen sind aktive Subscriptions verloren und müssen neu abgeschickt werden.
- 

## 4.3.1 Beschreibung der Elemente

### Systematik der Elementbeschreibung

Den Beschreibungen von OPC-XML-Elementen ist eine Darstellung der Tagstruktur vorangestellt:

- **Elemente** sind in der Tagstruktur in **fetter Schrift** dargestellt
- *Attribute* sind in der Tagstruktur in *kursiver Schrift* dargestellt

Die Verschachtelung der Tags wird durch entsprechende Einrückung wiedergegeben. Auch die Attribute eines Tags sind gegenüber dem entsprechenden Tag eingerückt.

### Beispiel

Gegeben sei folgende Tagstruktur:

```
<tag1>
    <tag2 att1="yes" att2="no">Inhalt von tag2
        </tag2>
        <tag3>Inhalt von tag3
        </tag3>
    </tag1>
```

In der Beschreibung von `<tag1>` würden Sie dann die folgende Darstellung finden:

```
tag1
  tag2
    att1
    att2
  tag3
```

Die Elemente **tag2** und **tag3** sind gegenüber dem Element **tag1** eingerückt, weil sie Unterelemente von **tag1** sind. Außerdem sind unmittelbar nach **tag2** die Attribute **att1** und **att2** dieses Elements aufgeführt.

### 4.3.2 Basis-Schemas

#### Schema-Definition für OPC XML-DA

Für alle Elemente von OPC XML-DA gibt es eine Schema-Definition. Häufig wird in diesen Beschreibungen auf die Definition grundlegender, mehrfach vorkommender Elemente verwiesen. Der folgende Abschnitt beschreibt einige dieser Elemente:

##### **ItemProperty**

Ein Element für die Definition von Eigenschaften eines Items.

##### **ItemValue**

Ein Element, mit dem der Wert eines Items sowie relevante Zusatzinformationen (z. B. der Zeitstempel) festgelegt werden.

##### **OPCError**

Enthält den Fehlercode und die Beschreibung eines OPC-Fehlers.

##### **ReplyBase**

Ein Element mit grundlegenden Informationen zu einer Serverantwort (z. B. Versendezeitpunkt der Antwort).

##### **RequestOptions**

Enthält Informationen, die eine Anfrage des Client genauer spezifizieren.

#### 4.3.2.1 ItemProperty

##### Tagstruktur

###### **ItemProperty**

*Name*

*Description*

*ItemPath*

*ItemName*

*ResultID*

**Value**

## Elemente und Attribute

Name	Beschreibung
Name	Enthält den Namen der Eigenschaft.
Description	Die Beschreibung der Eigenschaft.
ItemPath	Die Pfadangabe eines Elements. Dieses Attribut wird von SIMATIC NET nicht unterstützt.
ItemName	Mit dem Attribut ItemName ist ein Element im Namensraum des Servers eindeutig identifizierbar.
ResultID	Wenn ein Fehler oder eine unkritische Ausnahme auftritt, enthält dieses Attribut den Namen des OPC-Fehlers.
Value	Der aktuelle Wert der Eigenschaft.

### 4.3.2.2 ItemValue

#### Tagstruktur

```

ItemValue
  ValueTypeQualifier
  ItemPath
  ItemName
  ClientItemHandle
  Timestamp
  ResultID
  DiagnosticInfo
  Value
  Quality
    QualityField
    LimitField
    VendorField

```

## Elemente und Attribute

Name	Beschreibung
ValueTypeQualifier	Dieses Attribut wird nur bei Werten des Typs time, date und duration verwendet, um den Datentyp anzugeben. Bei allen anderen Datentypen ist dieses Attribut entweder nicht enthalten oder es wird ignoriert, wenn es vorhanden ist.
ItemPath	Die Pfadangabe eines Elements. Dieses Attribut wird von SIMATIC NET nicht unterstützt.
ItemName	Mit dem Attribut ItemName ist ein Element im Namensraum des Servers eindeutig identifizierbar.
ClientItemHandle	Eine vom Client im Request vorgegebene Zeichenkette, die der Server in seiner Antwort zurückliefert. Sie dient dem Client in komplexeren Systemen dazu, die Antworten den zugehörigen Anfragen zuzuordnen.

Name	Beschreibung
Timestamp	Der Zeitpunkt, zu dem die Werte der Daten zuletzt vom Server ermittelt wurden.
ResultID	Wenn ein Fehler oder eine unkritische Ausnahme auftritt, enthält dieses Attribut den Namen des OPC-Fehlers.
DiagnosticInfo	Ausführliche serverspezifische Diagnoseinformationen.
Value	Der Wert des Items. Da es sich bei diesem Attribut um eine polymorphe Angabe handelt, ist das zusätzliche Attribut xsi:type notwendig (zum Beispiel: xsi:type=xsd:float).
Quality	Eine Information über die Qualität der Daten. Wenn der Server dieses Attribut nicht zurückgibt, ist die Qualität <i>Good</i> . Wenn die Qualität <i>Bad</i> oder <i>Uncertain</i> ist, wird das Attribut zurückgegeben.
QualityField	Wenn die Qualität den Wert <i>Good</i> hat, wird dieses Attribut nicht zurückgegeben. Wenn die Qualität <i>Bad</i> ist und ein vorangegangener Wert für das Item bekannt ist, hat dieses Attribut den Wert <i>badLastKnownValue</i> .
LimitField	Enthält einen Namen, über den das OPC-Limit-Bitfeld angesprochen werden kann. Dieses Attribut wird immer übertragen, außer wenn der Limit-Status <i>none</i> ist.
VendorField	Ein numerischer Wert, der dem OPC-Vendor-Bitfeld entspricht. Ob dieses Attribut übertragen wird, hängt vom Hersteller ab.

#### 4.3.2.3 OPCError

##### Tagstruktur

```

OPCError
  ID
  Text

```

##### Elemente und Attribute

Name	Beschreibung
ID	Enthält den Namen des OPC-Fehlers.
Text	Eine Beschreibung des Fehlers in Textform. Der Inhalt der Zeichenkette ist abhängig vom Attribut <i>LocaleID</i> .

#### 4.3.2.4 ReplyBase

##### Tagstruktur

**ReplyBase**  
*RcvTime*  
*ReplyTime*  
*ClientRequestHandle*  
*RevisedLocaleID*  
*ServerState*

##### Elemente und Attribute

Name	Beschreibung
RcvTime	Angabe des Zeitpunkts, zu dem der Server die Anfrage erhalten hat. RcvTime ist ein notwendiges Attribut.
ReplyTime	Angabe des Zeitpunkts, zu dem der Server die Antwort verschickt hat, ein notwendiges Attribut.
ClientRequestHandle	Falls dieses Attribut in der Anfrage des Client enthalten war, wird es vom Server mit der Antwort zurückgegeben.
RevisedLocaleID	Wenn der Client für das Attribut <i>LocaleID</i> einen Wert angibt, der vom Server nicht unterstützt wird, gibt der Server mit dem Attribut <i>RevisedLocaleID</i> den voreingestellten Wert für das Attribut <i>LocaleID</i> zurück.
ServerState	Gibt den Status des Servers an und wird immer zurückgegeben, ein notwendiges Attribut.

#### 4.3.2.5 RequestOptions

##### Tagstruktur

**RequestOptions**  
*ReturnErrorText*  
*ReturnDiagnosticInfo*  
*ReturnItemTime*  
*ReturnItemName*  
*ReturnItemPath*  
*RequestDeadline*  
*ClientRequestHandle*  
*LocaleID*

## Elemente und Attribute

Name	Beschreibung
ReturnErrorText	Der voreingestellte Wert für dieses Attribut ist TRUE. In diesem Fall gibt der Server ausführliche Fehlerbeschreibungen zurück.
ReturnDiagnosticInfo	Der Server gibt Diagnoseinformationen zurück, wenn der Wert dieses Attributs TRUE ist.
ReturnItemTime	Gibt an, ob für jedes Item ein Zeitstempel zurückgegeben wird. Default-Wert ist FALSE, es wird kein Wert zurückgegeben.
ReturnItemName	Gibt an, ob für jedes Item das Attribut ItemName zurückgegeben wird. Default-Wert ist FALSE, es wird kein Wert zurückgegeben.
ReturnItemPath	Gibt an, ob für jedes Item das Attribut ItemPath zurückgegeben wird. Für ReturnItemPath=true oder false wird immer der Pfad zurückgeliefert (nicht Null). Default-Wert ist FALSE.
RequestDeadline	Gibt den spätesten Zeitpunkt an, bis zu dem der Client auf eine Antwort vom Server wartet. Daten für Items, die nicht bis zu diesem Zeitpunkt verfügbar sind, werden als Fehler zurückgegeben. Wenn der angegebene Zeitpunkt vor der aktuellen Zeit des Servers liegt, schlägt die gesamte Anfrage fehl. Der Client muss die Zeitangabe als UTC-Zeit liefern. Um den Wert benutzen zu können, muss bei der Parameterübergabe RequestDeadlineSpecified=true gesetzt werden. Ausnahme: OPC-XML-DA-Clients, die mit dem Microsoft Development Environment 2003 entwickelt werden, müssen die lokale Zeit (localtime), nicht die UTC Zeit, als RequestOptions.RequestDeadline übergeben. Der Standard SOAP Writer des Visual Studios 2003 wandelt die übergebenen Zeiten immer in eine UTC Zeit im Format localtime + Zeitversatz.
ClientRequestHandle	Falls dieses Attribut in der Anfrage des Client enthalten war, wird es vom Server mit der Antwort zurückgegeben. In großen und komplexen Systemen hilft diese Angabe dem Client bei der Zuordnung von Anfragen und Antworten. Die Angabe dieses Attributs ist optional.
LocaleID	Ein optionales Attribut, mit dem der Client die Sprache für bestimmte Rückgabedaten festlegt.

### 4.3.3 Read- und Write-Aufträge

#### 4.3.3.1 Read

##### Tagstruktur

```
Read
  Options
    ReturnErrorText
    ReturnDiagnosticInfo
    ReturnItemTime
    ReturnItemName
    ReturnItemPath
    RequestDeadline
    ClientRequestHandle
    LocaleID
  ItemList
    ItemPath
    ReqType
    MaxAge
  Items
    ItemPath
    ReqType
    ItemName
    ClientItemHandle
    MaxAge
```

##### Elemente und Attribute

###### Read

Das Element <Read> enthält alle Informationen für einen Leseauftrag.

###### Options

Das Element <Options> enthält Optionen, die für die XML-DA-Anfragen verfügbar sind. Dieses Element hat den Typ RequestOptions. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

###### ItemList

Das Container-Element für die einzelnen Items. Eine Anfrage kann mehrere Items enthalten.

Die Attribute *ItemPath*, *ReqType* und *MaxAge* können für die Elemente *ItemList* und *Items* verwendet werden. Ein Attribut für ein *Item* überschreibt das gleichnamige Attribut für *ItemList*.

###### ItemPath

Die Pfadangabe eines Elements. Dieses Attribut wird von SIMATIC NET nicht unterstützt.

**MaxAge**

Angabe einer Zeitdauer des Typs xsd:int, die die geforderte Aktualität der Daten festlegt. Die Daten sollen nicht älter als angegeben sein.

**ReqType**

Gibt den vom Client angeforderten Datentyp für den Wert des Items an. Es werden alle in der Spezifikation genannten Datentypen unterstützt.

**Items**

Element, mit dem ein einzelnes Item spezifiziert wird.

**ItemName**

Der Name des Items.

**ClientItemHandle**

Eine vom Client vergebene Zeichenkette. Wenn dieses Attribut vom Client angegeben wird, muss der Server es zurückgeben.

**Beispiel**

```
<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>
```

**C#-Schnittstelle für die synchrone Web-Methode Read**

Für die Verwendung in C#-Anwendungsprogrammen gibt es die folgende Methode für das Lesen von Werten. Es muss nur ein Teil der oben aufgeführten Parameter angegeben werden:

```
ReplyBase Read(RequestOptions Options,
    ReadRequestItemList ItemList,
    out ReplyItemList ItemValues,
    out OPCError[] Error);
```

#### 4.3.3.2 ReadResponse

##### Tagstruktur

```
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
```

##### Elemente und Attribute

###### ReadResponse

Mit dem Element <ReadResponse> liefert der Server die Ergebnisse eines Leseauftrags zurück.

###### ReadResults

Das Element <ReadResults> enthält grundlegende Informationen über die Antwort des Servers.

Dieses Element hat den Typ ReplyBase. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

###### RItemList

RItemList ist das Container-Element für die einzelnen Items einer Antwort. Eine Antwort kann mehrere Items enthalten.

###### Reserved

Dieses Attribut verhindert, dass WSDL-basierte Programme zur Code-Erzeugung die Rückgabekette als Array von Items darstellen.

###### Items

Element, mit dem ein einzelnes Item spezifiziert wird.

Dieses Element hat den Typ ItemValue. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

#### Errors

Ein Array von Fehlern, die bei dieser Antwort aufgetreten sind.

Dieses Element hat den Typ OPCError. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

## Beispiel

```
<soap:Body>
    <ReadResponse
        xmlns="http://opcfoundation.org/webservices/XMLDA/1.0/">
        <ReadResult
            RcvTime="2003-05-27T00:15:36.640000-07:00"
            ReplyTime="2003-05-27T00:15:36.750000-07:00"
            ServerState="running"
        />
        <RItemList>
            <Items
                <Item
                    ItemName="Simple Types/UInt"
                    Timestamp="2003-05-27T00:15:36.7343750-07:00">
                        <Value xsi:type="xsd:unsignedInt">
                            4294967295
                        </Value>
                    </Item>
            </Items>
            <Items
                <Item
                    ItemName="Simple Types/Int"
                    Timestamp="2003-05-27T00:15:36.7343750-07:00">
                        <Value xsi:type="xsd:int">
                            2147483647
                        </Value>
                    </Item>
            </Items>
            <Items
                <Item
                    ItemName="Simple Types/Float"
                    Timestamp="2003-05-27T00:15:36.7343750-07:00">
                        <Value xsi:type="xsd:float">
                            3.402823E+38
                        </Value>
                    </Item>
            </Items>
        </RItemList>
    </ReadResponse>
</soap:Body>
```

### 4.3.3.3 Write

#### Tagstruktur

```
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
```

#### Elemente und Attribute

##### Write

Das Element <Write> enthält alle Informationen für einen Schreibauftrag. Der Schreibauftrag kann für ein oder mehrere Items ausgeführt werden.

##### ReturnValuesOnReply

Mit diesem Attribut kann der Client angeben, ob für jedes Item ein Wert zurückgegeben werden soll. Es wird der Wert zurückgegeben, der vom Server für diesen Schreibauftrag verarbeitet wurde. Es ist der gleiche Wert, den der Server liefern würde, wenn direkt nach dem Schreibauftrag ein Leseauftrag erfolgen würde. Wenn der Schreibauftrag fehlschlägt, werden keine Werte zurückgegeben. Für nur schreibbare Werte liefert der Server E\_WRITEONLY.

##### Options

Das Element <Options> enthält Optionen, die für die XML-DA-Anfragen verfügbar sind. Dieses Element hat den Typ RequestOptions. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

**ItemList**

Das Container-Element für die einzelnen Items. Eine Anfrage kann mehrere Items enthalten.

**ItemName**

Mit dem Attribut ItemName ist ein Element im Namensraum des Servers eindeutig identifizierbar.

**Items**

Element, mit dem ein einzelnes Item spezifiziert wird. Dieses Element hat den Typ ItemValue. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

**Hinweis**

Beim Schreiben und Wiedereinlesen von *DateTime*-Items können Fehler bis zu 2 ms auftreten, die durch die .NET-Interop-Schicht verursacht werden.

**Beispiel**

```
<soap:Body>
    <Write
        xmlns="http://opcfoundation.org/webservices/XMLDA/1.0/">
        <Options
            ReturnErrorText="false"
            ReturnItemName="true"
            LocaleID="en"
        />
        <ItemList>
            <Items ItemName="Simple Types/UIInt">
                <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>
```

**C#-Schnittstelle für die synchrone Web-Methode Write**

Für die Verwendung in C#-Anwendungsprogrammen gibt es die folgende Methode für das Schreiben von Werten. Es muss nur ein Teil der oben aufgeführten Parameter angegeben werden:

```
ReplyBase Write(RequestOptions Options,
                 WriteRequestItemList ItemList,
                 bool ReturnItemVal,
```

```
        out ReplyItemList ItemValues,
        out OPCError[] Error);
```

#### 4.3.3.4 WriteResponse

##### Tagstruktur

```
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
```

##### Elemente und Attribute

###### WriteResponse

Mit dem Element <WriteResponse> liefert der Server die Ergebnisse eines Schreibauftrags zurück.

###### WriteResult

Das Element <WriteResult> enthält grundlegende Informationen über die Antwort des Servers.

Dieses Element hat den Typ ReplyBase. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

###### RItemList

Das Container-Element für die einzelnen Items. Eine Antwort kann mehrere Items enthalten.

**Reserved**

Dieses Attribut verhindert, dass WSDL-basierte Programme zur Code-Erzeugung die Rückgabekette als Array von Items darstellen.

**Items**

Element, mit dem ein einzelnes Item spezifiziert wird.

Dieses Element hat den Typ ItemValue. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

**Errors**

Ein Array von Fehlern, die bei dieser Antwort aufgetreten sind.

Dieses Element hat den Typ OPCError. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

**Beispiel**

```
<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 Verwendung von Subscriptions****Kontakt über mehrere Methodenaufrufe hinweg**

Im Unterschied zu den meisten anderen Web-Diensten unterstützt der OPC XML-DA-Webservice einen über mehrere Methodenaufrufe hinweg bestehenden losen Kontakt zu einem angemeldeten Web-Client. Dieser lose Kontakt wird im weiteren Text als polled Subscription bezeichnet. Mit einer polled Subscription können die Werte von Items zyklisch gelesen werden.

**An- und Abmeldung durch den Client**

Der Client meldet eine solche Subscription beim OPC XML-DA-Webservice an. Dieser merkt sich daraufhin, welche Items für den Client von Interesse sind und ermittelt deren Werte. Wenn dann ein sogenannter polled Request von einem Client kommt, werden die aktuellen Werte zurückgeliefert. Beendet wird die Subscription durch den Client.

## Identifikation durch Handles

Zur Identifikation der jeweiligen Subscription wird bei jedem weiteren Aufruf ein vom Server vergebene Handle als Parameter übergeben. Es kann auch eine Liste von Handles angegeben werden, wodurch pro SubscriptionPolledRefresh mehrere Subscriptions relevant sind. Dadurch erhöht sich die Effizienz des Pollens.

Als Subscription-Handle wird die Adresse des Subscription-Objekts an den Client zurückgeliefert, damit bei den folgendem SubscriptionPolledRefresh und SubscriptionCancel die Subscription eindeutig identifiziert werden kann.

Die folgende Grafik zeigt die Nutzung einer Subscription:

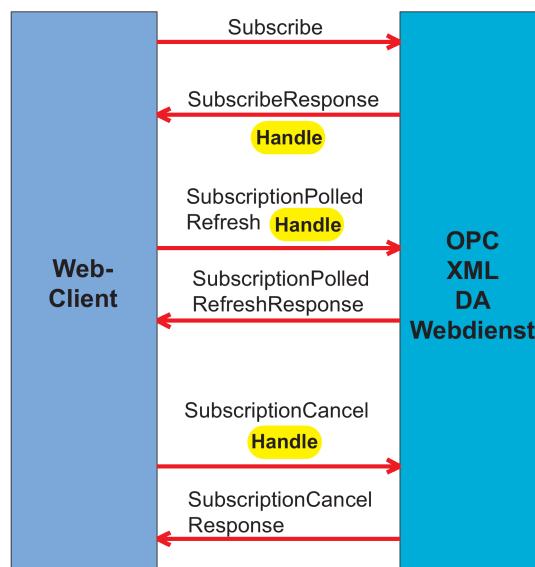


Bild 4-9 Nutzung einer Subscription

## Gültigkeitsdauer der zu beobachtenden Items

Die Methode SubscriptionPolledResponse liefert die Werte an den Client zurück, die mit der Methode SubscriptionPolledRefresh angefragt wurden. Beim SubscriptionCancel werden die Subscriptions mit ihren Items verworfen. Kommen nach einer gewissen Zeit (SubscriptionPingRate) keine weiteren SubscriptionPolledRefresh-Requests vom Client, beendet der OPC-DA-Web-Dienst die Subscription eigenständig und analog zur Methode SubscriptionCancel.

#### 4.3.4.1 Subscribe

##### Tagstruktur

```

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

```

##### Elemente und Attribute

###### **Subscribe**

Das Element <Subscribe> enthält alle Informationen für einen Subscribe-Auftrag.

###### **ReturnValuesOnReply**

Falls der Wert für dieses Attribut auf TRUE gesetzt wurde, gibt der Server Werte zurück, die für eine SubscribeResponse-Antwort verfügbar sind. Ist der Wert FALSE, liefert der Server mit einer SubscribeResponse-Antwort keine Werte zurück.

###### **SubscriptionPingRate**

Der Wert für dieses Attribut legt fest, innerhalb welcher Zeitspanne der Server die Existenz des Client überprüft. Falls der Client innerhalb der angegebenen Zeitspanne nicht mit dem Server kommuniziert hat, kann der Server alle Ressourcen freigeben, die für den Subscription-Vorgang des Client erforderlich waren.

###### **Options**

Das Element <Options> enthält Optionen, die für die XML-DA-Anfragen verfügbar sind. Dieses Element hat den Typ RequestOptions. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

### **ItemList**

Das Container-Element für die einzelnen Items. Eine Anfrage kann mehrere Items enthalten.

Die Attribute *ItemPath*, *ReqTyp*, *Deadband*, *RequestedSamplingRate* und *EnableBuffering* können für die Elemente *ItemList* und *Items* verwendet werden. Ein Attribut für ein *Item* überschreibt das gleichnamige Attribut für *ItemList*.

#### **ItemPath**

Die Pfadangabe eines Elements. Dieses Attribut wird von SIMATIC NET nicht unterstützt.

#### **ReqType**

Gibt den vom Client angeforderten Datentyp für den Wert des Items an.

#### **Deadband**

Legt einen Grenzwert fest, bis zu dem Wertänderungen eines Items keinen SubscriptionPolledRefresh auslösen. Die Angabe erfolgt in Prozent vom maximalen Wertebereich des Items. Der Wert für diesen Parameter sollte deshalb zwischen 0 und 100 Prozent liegen und kann nur bei Items des Typs integer und float verwendet werden.

#### **RequestedSamplingRate**

Vom Client festgelegte Zeit in Millisekunden, nach der der Server die Werte auf Änderung überprüfen soll.

#### **EnableBuffering**

Falls der Wert dieses Attributs auf TRUE gesetzt wird, berücksichtigt der Server den Parameter *RequestedSamplingRate* und speichert alle Wertänderungen in einem Puffer. Die gespeicherten Daten gibt der Server bei der nächsten Anfrage des Typs *PolledRequest* an den Client zurück.

#### **Item Name**

Der Name des Items.

#### **ClientItemHandle**

Eine vom Client vergebene Zeichenkette. Wenn dieses Attribut vom Client angegeben wird, muss der Server es zurückgeben.

#### 4.3.4.2 SubscribeResponse

##### Tagstruktur

```

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

```

##### Elemente und Attribute

###### **SubscribeResponse**

Das Element <SubscribeResponse> enthält alle Informationen für eine SubscribeResponse-Antwort.

###### **ServerSubHandle**

Der vom Server für dieses Attribut festgelegte Wert muss bei SubscriptionPolledRefresh-Aufrufen und bei SubscriptionCancel-Aufrufen verwendet werden. Dieses Attribut identifiziert den Client, der die Anfrage abschickt.

###### **SubscribeResult**

Das Element <SubscribeResult> enthält grundlegende Informationen über die Antwort des Servers.

Dieses Element hat den Typ *ReplyBase*. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

#### RItemList

Enthält Items-Elemente. Die verfügbaren Werte für die Items werden nur dann an den Client gesendet, wenn der Client sie durch einen entsprechenden Wert für das Attribut ReturnValuesOnReply (siehe Subscribe) angefordert hat.

#### RevisedSamplingRate

Der kürzeste Aktualisierungszyklus, der vom Server unterstützt wird. Dieser Wert wird vom Server an den Client zurückgemeldet.

#### Items

Element, mit dem ein einzelnes Item spezifiziert wird.

Dieses Element hat den Typ ItemValue. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

#### Errors

Ein Array von Fehlern, die bei dieser Antwort aufgetreten sind.

Dieses Element hat den Typ OPCError. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

### 4.3.4.3 SubscriptionPolledRefresh

#### Tagstruktur

##### SubscriptionPolledRefresh

*HoldTime*

*WaitTime*

*ReturnAllItems*

##### Options

*ReturnErrorText*

*ReturnDiagnosticInfo*

*ReturnItemTime*

*ReturnItemName*

*ReturnItemPath*

*RequestDeadline*

*ClientItemHandle*

*LocaleID*

##### ServerSubHandles

#### Elemente und Attribute

##### SubscriptionPolledRefresh

Mit diesem Element fordert der Client beim Server eine Aktualisierung der Items an, die in einer vorangegangenen Subscription-Anfrage definiert wurden.

##### HoldTime

Dieses Attribut veranlasst den Server, solange keine aktualisierten Werte an den Client zurückzuschicken, bis die interne Zeit des Servers gleich oder größer als der Wert dieses Attributs ist.

##### WaitTime

Nach Erreichen des im Attribut HoldTime festgelegten Zeitpunkts wartet der Server die mit

WaitTime definierte Zeitdauer, bevor er eine Antwort zurückgibt. Wenn sich innerhalb der WaitTime der Wert eines Items ändert wird allerdings sofort eine Antwort gesendet.

#### **ReturnAllItems**

Wenn der Wert dieses Attributs auf FALSE gesetzt ist, gibt der Server nur solche Items zurück, die sich zwischen der vorangegangenen SubscriptionPolledRefresh-Anfrage und der aktuellen SubscriptionPolledRefresh-Anfrage geändert haben.

Ist dieses Attribut auf TRUE gesetzt, gibt der Server alle Items zurück, die in der entsprechenden Subscription-Anfrage definiert wurden.

#### **Options**

Das Element <Options> enthält Optionen, die für die XML-DA-Anfragen verfügbar sind. Dieses Element hat den Typ RequestOptions. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

#### **ServerSubHandles**

Dieses Attribut wird vom Server zur Identifikation der zu pollenden Subscription verwendet.

### 4.3.4.4 SubscriptionPolledRefreshResponse

#### Tagstruktur

##### **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*

## Elemente und Attribute

### **SubscriptionPolledRefreshResponse**

Dieses Element enthält alle Informationen, die der Server als Antwort auf eine SubscriptionPolledRefresh-Anfrage an den Client sendet.

### **DataBufferOverflow**

Ist der Wert dieser Attributs TRUE, haben sich Änderungen für die Items ergeben, die aber wegen beschränkter Ressourcen nicht gespeichert werden konnten. Die einzelnen Items zeigen an, ob sie von der Ressourcenbeschränkung betroffen sind oder nicht.

### **SubscriptionPolledRefreshResult**

Dieses Element enthält grundlegende Informationen über die Antwort des Servers. Es hat den Typ ReplyBase. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

### **InvalidServerSubHandles**

Vom Server als ungültig erkannte ServerSubHandles.

### **RItemList**

Das Element <RItemList> enthält Werte für alle Items, wenn der Client bei seiner SubscriptionPolledRefresh-Anfrage das Attribut ReturnAllItems mit dem Wert TRUE übergeben hat. Andernfalls gibt der Server nur Werte zurück, die sich geändert haben.

### **SubscriptionHandle**

Handle der beim Subscription-Aufruf vom Client angegebenen RItemList.

### **Items**

Element, mit dem ein einzelnes Item spezifiziert wird.

Dieses Element hat den Typ ItemValue. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

### **Errors**

Ein Array von Fehlern, die bei dieser Antwort aufgetreten sind.

Dieses Element hat den Typ OPCError. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

## 4.3.4.5 SubscriptionCancel

### Tagstruktur

#### **SubscriptionCancel**

*ServerSubHandle*

*ClientRequestHandle*

## Elemente und Attribute

### **SubscriptionCancel**

Mit dem Element <SubscriptionCancel> beendet der Client die Subscription und der Server kann die entsprechenden Ressourcen freigeben.

### **ServerSubHandle**

Der vom Server für dieses Element vergebene Wert identifiziert die Subscription, der die Anfrage abschickt.

#### **ClientRequestHandle**

Falls dieses Attribut in der Anfrage des Client enthalten ist, wird es vom Server mit der SubscriptionCancelResponse-Antwort zurückgegeben. In großen und komplexen Systemen hilft diese Angabe dem Client bei der Zuordnung von Anfragen und Antworten. Die Angabe dieses Attributs ist optional.

### 4.3.4.6      **SubscriptionCancelResponse**

#### Tagstruktur

##### **SubscriptionCancelResponse**

*ClientRequestHandle*

#### Elemente und Attribute

##### **SubscriptionCancelResponse**

Mit diesem Element bestätigt der Server eine SubscriptionCancel-Anfrage des Client.

##### **ClientRequestHandle**

Falls der Client bei seiner Anfrage dieses Attribut verwendet hat, gibt es der Server bei seiner Antwort zurück.

### 4.3.5      **Weitere Abfragen**

#### 4.3.5.1    **Browse**

#### Tagstruktur

##### **Browse**

*LocaleID*

*ClientRequestHandle*

*ItemPath*

*ItemName*

*ContinuationPoint*

*MaxElementsReturned*

*BrowseFilter*

*ElementNameFilter*

*VendorFilter*

*ReturnAllProperties*

*ReturnPropertyValues*

*ReturnErrorText*

*PropertyNames*

## Elemente und Attribute

### **Browse**

Das Element <Browse> enthält alle Informationen, die für das Navigieren durch einen hierarchischen Adressraum notwendig sind.

### **LocaleID**

Ein optionales Attribut, mit dem der Client die Sprache für bestimmte Rückgabedaten festlegt.

### **ClientRequestHandle**

Falls dieses Attribut in der Anfrage des Client enthalten war, wird es vom Server mit der Antwort zurückgegeben. In großen und komplexen Systemen hilft diese Angabe dem Client bei der Zuordnung von Anfragen und Antworten. Die Angabe dieses Attributs ist optional.

### **ItemPath**

Der Pfad zu dem Item, bei dem das Navigieren durch den Adressraum startet. Bei einer zweiten Browse-Anfrage muss dieses Attribut den gleichen Wert haben wie bei der vorangegangenen Anfrage.

### **ItemName**

Der Name des Items, bei dem das Navigieren durch den Adressraum startet. Bei einer zweiten Browse-Anfrage muss dieses Attribut den gleichen Wert haben wie bei der vorangegangenen Anfrage.

### **ContinuationPoint**

Falls es sich nicht um eine initiale Browse-Anfrage handelt, kann hier ein Punkt angegeben werden, von dem aus die Browse-Anfrage neu startet.

### **MaxElementsReturned**

Maximale Anzahl der Rückgabewerte.

Wird hier ein negativer Wert übergeben, tritt eine Exception auf.

### **BrowseFilter**

Dieses Attribut mit dem Aufzählungsdatentyp browseFilter (all, branch, item) legt fest, welche Teilmenge von Elementen zurückgegeben wird.

### **ElementNameFilter**

Ein regulärer Ausdruck, der zum Auswählen von Elementen verwendet wird.

### **VendorFilter**

Ein herstellerspezifischer Ausdruck, der zum Auswählen von herstellerspezifischen Informationen verwendet wird. Die Auswirkungen auf das Element ElementNameFilter sind nicht definiert.

### **ReturnAllProperties**

Wenn für dieses Attribut der Wert TRUE gesetzt wird, gibt der Server für jedes Rückgabe-Element alle verfügbaren Eigenschaften zurück. Der Wert des Attributs PropertyNames wird in diesem Fall ignoriert.

### **ReturnPropertyValue**

Wenn für diese Attribut der Wert TRUE gesetzt wird, gibt der Server nicht nur die Namen der Eigenschaften sondern auch die Werte der Eigenschaften zurück.

### **ReturnErrorText**

Wenn dieses Attribut den Wert TRUE hat, liefert der Server eine ausführliche Fehlerbeschreibung.

**PropertyNames**

Namen von Eigenschaften, die mit jedem Element zurückgegeben werden. Wenn der Wert des Attributs ReturnAllProperties auf TRUE gesetzt ist, gibt der Server unabhängig von PropertyNames alle Eigenschaften zurück.

**4.3.5.2 BrowseResponse****Tagstruktur****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***Elemente und Attribute****BrowseResponse**

Enthält alle Informationen einer Antwort auf eine Browse-Anfrage.

**ContinuationPoint**

Der Server kann hier einen Punkt angeben, von dem aus eine folgende Browse-Anfrage gestartet werden kann. Dieses Attribut wird von SIMATIC NET nicht unterstützt.

**MoreElements**

Falls der Server das Attribut ContinuationPoint nicht unterstützt und die Anzahl der Rückgabewerte den Wert von MaxElementsReturned übersteigt, setzt der Server dieses Attribut auf TRUE.

### **BrowseResult**

Das Element <BrowseResponse> enthält grundlegende Informationen über die Antwort des Servers.

Dieses Element hat den Typ ReplyBase. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

#### **Elements**

Enthält Informationen über die geparssten Elemente des Strukturaums.

#### **Name**

Name eines Elements im Namensraum, der dem Benutzer gezeigt wird (beispielsweise in einer Strukturansicht).

#### **ItemPath**

Die Pfadangabe eines Elements. Dieses Attribut wird von SIMATIC NET nicht unterstützt.

#### **ItemName**

Mit dem Attribut ItemName ist ein Element im Namensraum des Servers eindeutig identifizierbar.

#### **IsItem**

Wenn für dieses Attribut der Wert auf TRUE gesetzt ist, dann ist dieses Element ein Item, das in Read-, Write- und Subscribe-Anfragen verwendet werden kann.

#### **HasChildren**

Wenn für dieses Attribut der Wert auf TRUE gesetzt ist, hat das entsprechende Element Kindelemente.

#### **Properties**

Das Element <Properties> enthält Informationen zu einer Eigenschaft, die über einen Browse- oder GetProperties-Aufruf zugänglich gemacht werden.

Dieses Element hat den Typ ItemProperty. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

#### **Errors**

Ein Array von Fehlern, die bei dieser Anfrage aufgetreten sind.

Dieses Element hat den Typ OPCError. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

## 4.3.5.3 **GetProperties**

### Tagstruktur

#### **GetProperties**

*LocaleID*

*ClientRequestHandle*

*ItemPath*

*ReturnAllProperties*

*ReturnPropertyValues*

*ReturnErrorText*

**ItemIDs**

*ItemPath*

*ItemName*

**PropertyNames**

## Elemente und Attribute

### **GetProperties**

Mit dem Element <GetProperties> können Sie Eigenschaften abfragen.

### **LocaleID**

Ein optionales Attribut, mit dem der Client die Sprache für bestimmte Rückgabedaten festlegt.

### **ClientRequestHandle**

Falls dieses Attribut in der Anfrage des Client enthalten war, wird es vom Server mit der Antwort zurückgegeben. In großen und komplexen Systemen hilft diese Angabe dem Client bei der Zuordnung von Anfragen und Antworten. Die Angabe dieses Attributs ist optional.

### **ItemPath**

Die Pfadangabe eines Elements. Dieses Attribut wird von SIMATIC NET nicht unterstützt.

### **ReturnAllProperties**

Wenn für dieses Attribut der Wert auf TRUE gesetzt wird, gibt der Server für jedes Rückgabe-Element alle Eigenschaften zurück. Der Wert des Elements PropertyNames wird in diesem Fall ignoriert.

### **ReturnPropertyValues**

Wenn für dieses Attribut der Wert auf TRUE gesetzt wird, gibt der Server nicht nur die Namen der Eigenschaften sondern auch die Werte der Eigenschaften zurück.

### **ReturnErrorText**

Wenn dieses Attribut den Wert TRUE hat, liefert der Server eine ausführliche Fehlerbeschreibung.

### **ItemIDs**

Enthält eine Liste mit Items, für die die Eigenschaften ermittelt werden sollen.

### **ItemPath**

Die Pfadangabe eines Elements. Dieses Attribut wird von SIMATIC NET nicht unterstützt.

### **ItemName**

Mit dem Attribut ItemName ist ein Element im Namensraum des Servers eindeutig identifizierbar.

### **PropertyNames**

Namen von Eigenschaften, die mit jedem Element zurückgegeben werden. Wenn der Wert des Attributs ReturnAllProperties auf TRUE gesetzt ist, gibt der Server unabhängig von PropertyNames alle Eigenschaften zurück.

#### 4.3.5.4 GetPropertiesResponse

##### Tagstruktur

```
GetPropertiesResponse
  GetPropertiesResult
    RcvTime
    ReplyTime
    ClientRequestHandle
    RevisedLocaleID
    ServerState
    PropertyLists
      ItemPath
      ItemName
      ResultID
    Properties
      Name
      Description
      ItemPath
      ItemName
      ResultID
      Value
    Errors
      ID
      Text
```

##### Elemente und Attribute

###### GetPropertiesResponse

Mit diesem Element gibt der Server Antworten auf GetProperties-Anfragen zurück.

###### GetPropertiesResult

Das Element <GetPropertiesResult> enthält grundlegende Informationen über die Antwort des Servers.

Dieses Element hat den Typ ReplyBase. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

###### PropertyLists

Für jedes Item wird ein Element dieses Typs zurückgegeben, das die angefragten Eigenschaften des Items enthält.

Die Attribute *Item Path*, *ItemName* und *ResultID* sind für die Elemente PropertyLists und Properties verwendbar. Werden diese Attribute in einem Element Properties angegeben, so werden für dieses Element die Werte der gleichnamigen Attribute bei PropertyLists überschrieben.

###### ItemPath

Die Pfadangabe eines Elements. Dieses Attribut wird von SIMATIC NET nicht unterstützt.

**ItemName**

Mit dem Attribut ItemName ist ein Element im Namensraum des Servers eindeutig identifizierbar.

**ResultID**

Wenn ein Fehler oder eine unkritische Ausnahme auftritt, enthält dieses Attribut den Namen des OPC-Fehlers.

**Properties**

Dieses Element hat den Typ ItemProperty. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

**Errors**

Ein Array von Fehlern, die bei dieser Antwort aufgetreten sind.

Dieses Element hat den Typ OPCError. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

#### 4.3.5.5 GetStatus

##### Tagstruktur

**GetStatus**

*LocaleID*  
*ClientRequestHandle*

##### Elemente und Attribute

**GetStatus**

Eine GetStatus-Anfrage überprüft den Web-Service und ermittelt herstellerspezifische Informationen, die anderen OPC-Methoden nicht zugänglich sind.

**LocaleID**

Ein optionales Attribut, mit dem der Client die Sprache für bestimmte Rückgabedaten festlegt.

**ClientRequestHandle**

Falls dieses Attribut in der Anfrage des Client enthalten war, wird es vom Server mit der Antwort zurückgegeben. In großen und komplexen Systemen hilft diese Angabe dem Client bei der Zuordnung von Anfragen und Antworten. Die Angabe dieses Attributs ist optional.

##### Beispiel

```
<soap:Body>
    <GetStatus
        LocaleID="de-AT"
        xmlns="http://opcfoundation.org/webservices/XMLDA/1.0/"
    />
</soap:Body>
```

## C#-Schnittstelle für die Web-Methode GetStatus

Für die Verwendung in C#-Anwenderprogrammen gibt es die folgende Methode für das Überprüfen des Web-Services:

```
ServerStatus GetStatus(string LocaleID,  
                        string ClientRequestHandle,  
                        out ServerStatus);
```

### 4.3.5.6 GetStatusResponse

#### Tagstruktur

```
GetStatusResponse  
  GetStatusResult  
    RcvTime  
    ReplyTime  
    ClientRequestHandle  
    RevisedLocaleID  
    ServerState  
  Status  
    StartTime  
    ProductVersion  
    StatusInfo  
    VendorInfo  
    SupportedLocaleIDs  
    SupportedInterfaceVersions
```

#### Elemente und Attribute

##### **GetStatusResponse**

Enthält alle Informationen einer Antwort auf eine GetStatus-Anfrage

##### **GetStatusResult**

Das Element <GetStatusResult> enthält grundlegende Informationen über die Antwort des Servers.

Dieses Element hat den Typ ReplyBase. Informationen zu den Attributen dieses Elements finden Sie im entsprechenden Abschnitt dieser Dokumentation.

##### **Status**

Das Element <Status> enthält die einzelnen Informationen über den Server-Status.

##### **StartTime**

Zeitpunkt im UTC-Format, zu dem der Server gestartet wurde.

##### **ProductVersion**

Versionsnummer des Servers, bestehend aus Major-, Minor- und Build-Nummer.

##### **StatusInfo**

Enthält zusätzliche Informationen über den Server-Status und kann sprachabhängig sein.

##### **VendorInfo**

Herstellerspezifische Zeichenkette mit zusätzlichen Informationen über den Server. Der Wert

dieses Attributs sollte den Firmennamen enthalten sowie Informationen darüber, welche Geräte unterstützt werden.

**SupportedLocaleIDs**

Die vom Server unterstützten Ländereinstellungen. Ein Server braucht nicht alle Ländereinstellungen für alle Items zu unterstützen.

**SupportedInterfaceVersions**

Ein Array von Zeichenketten, die die Versionen der XML-DA Spezifikation enthalten, die der Server unterstützt.

## **4.4 OPC-UA-Schnittstelle programmieren**

### **Was finden Sie in diesem Kapitel?**

In diesem Kapitel wird die Programmierschnittstelle des OPC-UA-Servers und die erweiterte Konfiguration und Zertifikatverwaltung beschrieben.

#### **Schnittstellen**

Es gibt sowohl eine C-Schnittstelle für einen einfachen performanten Zugriff aus C-Programmen als auch eine .NET-Schnittstelle mit komfortabler Verwendung in einer .NET-Anwendung. Die zugehörigen Bibliotheken und Assemblies sind nach Installation der SIMATIC NET PC-Software vorhanden.

#### **Verfügbarkeit der OPC-UA-Schnittstelle unter SIMATIC NET**

Die OPC-UA-Schnittstelle ist ab der SIMATIC NET CD mit Edition 2008 (V7.1) verfügbar.

#### **Auffinden eines OPC-UA-Servers**

Das Auffinden eines OPC-UA-Servers erfolgt über OPC-UA-Discovery-Dienste. Dabei wird nur der Rechnername bzw. die IP-Adresse und der Discovery-Port 4840 vorgegeben. Die vorhandenen angemeldeten OPC-UA-Server geben dann ihre konfigurierten OPC-UA-Endpunkte zurück.

#### **Sicherheit und Authentifizierung**

Für eine sichere Kommunikation (OPC-UA-CreateSecureChannel ) zwischen OPC-UA-Client und Server müssen beidseitig X509-Zertifikate vorhanden sein. Diese werden zu Beginn der Kommunikation ausgetauscht. Der OPC-UA-Server wie auch der Client entscheiden zu diesem Zeitpunkt, ob dem gelieferten Zertifikat vertraut wird oder nicht.

#### **Kommunikation**

Nach Akzeptieren der Zertifikate kann eine OPC-UA-Session aufgebaut und aktiviert werden.

OPC-UA-Knoten können beobachtet (OPC-UA-MonitoredItems and Subscriptions) oder einfach nur einmalig gelesen oder geschrieben werden. Wenn Knoten mehrmals gelesen oder geschrieben werden sollen, dann wird aus Gründen der Schnelligkeit dringend empfohlen, die Knoten vorher zu registrieren (OPC-UA-RegisterNodes). Beim Schreiben oder Lesen wird dann auf diese Referenz verwiesen.

## Transparente Redundanz

Höhere Verfügbarkeit von Automatisierungsdaten ohne zusätzliche Software für UA-Clients wird durch die transparenten redundanten S7-UA-Server erreicht. Für Redundanz und Lastausgleich ist nur der Standard "Industrial Ethernet" erforderlich.

## Zugriff auf Array-Feldelemente unter OPC UA mittels IndexRange

Es gibt beim Zugriff mit IndexRange Einschränkungen, abhängig von der Zielvariablen. Das Schreiben von Teilen von Bit-Arrays per IndexRange zu S7-Bausteinen wird aus Konsistenzgründen abgelehnt. Es werden nur eindimensionale IndexRanges unterstützt. Beim Lesen mit IndexRange wird immer die gesamte Variable gelesen, auch wenn möglicherweise mehr vom Kommunikationspartner angefordert wird.

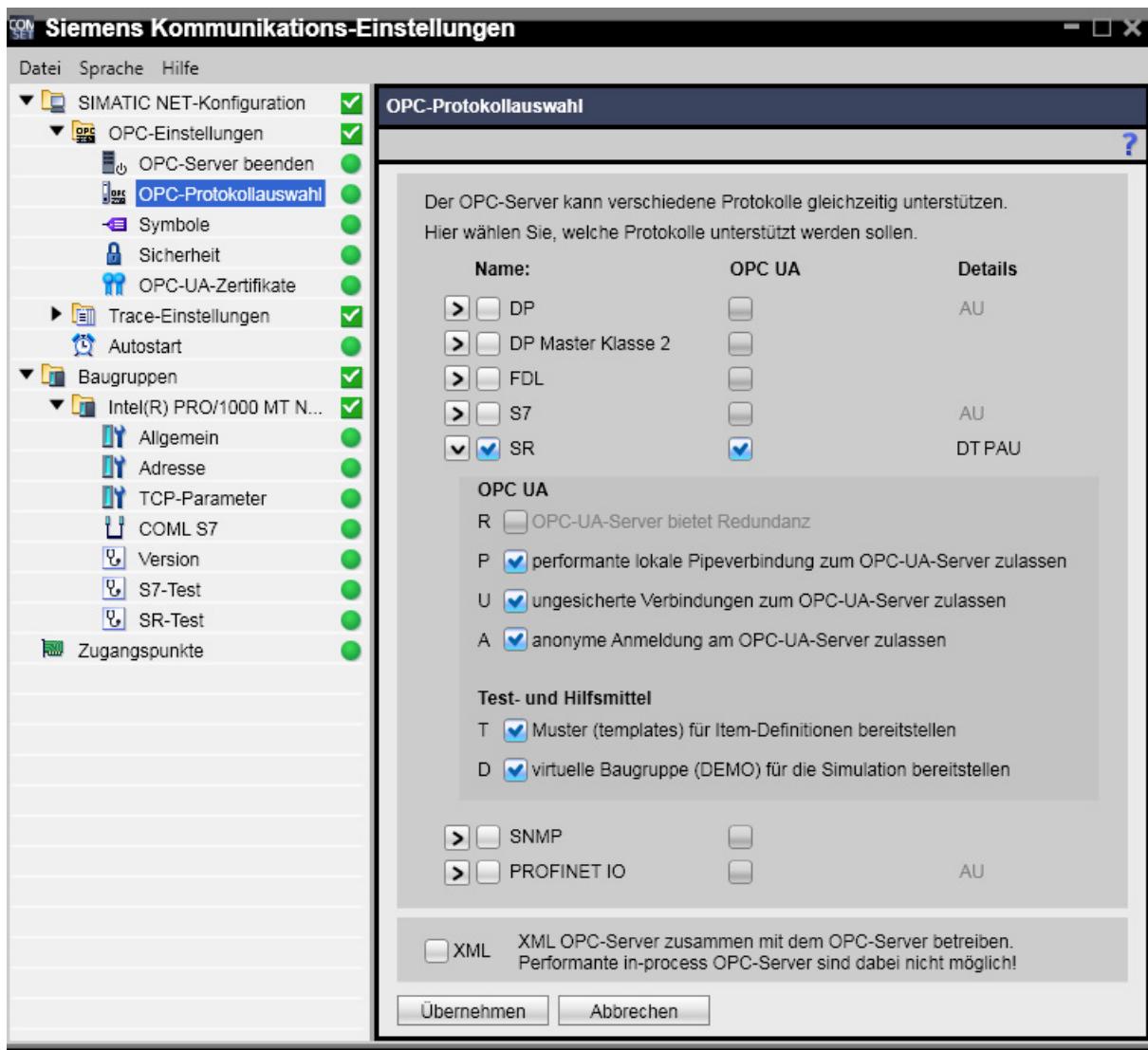
### 4.4.1 OPC-UA-Server konfigurieren

#### 4.4.1.1 Authentifizierung

##### Benutzer-Authentifizierung

Nach der Installation ist die Authentifizierung für Testzwecke und einfache Inbetriebnahme deaktiviert. Es wird jedoch empfohlen, die Benutzer-Authentifizierung zu aktivieren.

Die anonyme Anmeldung am OPC-UA-Server sollte im Konfigurationsprogramm "Kommunikations-Einstellungen" unter "OPC-Einstellungen" > "OPC-Protokollauswahl" > Klicken auf das Pfeilsymbol des entsprechenden Protokolls > "anonyme Anmeldung am OPC-UA-Server zulassen" deaktiviert werden. Sie kann hier auch wieder aktiviert werden.



Mit dieser Einstellung werden vorhandene Windows-Benutzer und deren Passwörter bei sicherem Verbindungsaufbau abgefragt. Benutzername und Passwort werden nach Client-spezifischer Vorgehensweise angegeben.

#### 4.4.1.2 Endpunkt-Sicherheit

##### Endpunkt-Sicherheit des OPC-UA-Servers

Die Endpunkt-Sicherheit "None" (keine Sicherheitsüberprüfung und Verschlüsselung) "ungesicherte Verbindungen zum OPC-UA-Server zulassen" sollte im Konfigurationsprogramm "Kommunikations-Einstellungen" unter "OPC-Protokollauswahl" deaktiviert werden. Damit werden dann nur sichere Endpunkte angeboten. Sie kann hier auch wieder aktiviert werden.

## 4.4.2 Zertifikatsverwaltung für den OPC-UA-Server

### Verzeichnisse der Zertifikatsverwaltung

Die Zertifikatsverwaltung für OPC-UA-Server basiert auf OpenSSL (<http://www.openssl.org/>) und befindet sich in folgendem Verzeichnis:

- Windows 7/Windows Server 2008 R2 und Windows 8/Windows Server 2012
  - Versteckter Pfad im Dateisystem:  
"<System-Laufwerk>:\ProgramData\Siemens\OPC\PKI\CA\"

### OPC-UA-Server-Zertifikate

Im oben genannten Verzeichnis befindet sich im Unterverzeichnis "\certs" der Ablageort für die rechnerspezifischen OPC-UA-Server-Zertifikate im X509-Format mit der Endung "der"

- für S7-UA: OPC.SimaticNET.S7.der inklusive des PublicKey,
- für S7OPT-UA: OPC.SimaticNET.S7OPT.der inklusive des PublicKey,
- für PROFINET IO-UA: OPC.SimaticNET.PNIO.der inklusive des PublicKey,
- für DP-UA: OPC.SimaticNET.DP.der inklusive des PublicKey,
- für SEND/RECEIVE-UA: OPC.SimaticNET.SR.der inklusive des PublicKey.

Diese Zertifikate werden bei Kommunikationsbeginn dem Client übermittelt. Abhängig vom OPC-UA-Client können oder müssen sie in die Zertifikatsverwaltung des OPC-UA-Client importiert werden.

---

#### Hinweis

Beachten Sie weiterhin, dass im Computernamen, und damit auch im Zertifikat, nur standardmäßige Zeichen verwendet werden dürfen.

Standardmäßige Namen bestehen aus Buchstaben (A-Z, a-z), Zahlen (0-9) und (-). Wenn Sie andere Zeichen im Computernamen verwenden, können keine Zertifikate erzeugt werden.

---

#### Hinweis

Wenn Sie nach der Installation der "SIMATIC NET PC Software" den Computernamen ändern, sind die für OPC UA installierten Zertifikate ungültig und OPC UA ist damit nicht mehr funktionsfähig. Abhilfe: Erstellen Sie eine neue UA-Konfiguration mit dem Programm "Kommunikations-Einstellungen" bei "OPC-UA-Zertifikate" > Kontextmenü "OPC-UA-Konfiguration neu erstellen".

---

### UA-Client-Zertifikate für die OPC-UA-Server

Auf dem oben genannten Verzeichnis befindet sich ebenfalls im Unterverzeichnis "\certs" der Ablageort für rechnerspezifische unterscheidbare OPC-UA-Client-Zertifikate. Für den lokal installierten OPC Scout V10 beispielsweise ist es das Zertifikat "OPC.ScoutV10.der" inklusive des PublicKey.

Dieses Zertifikat wird bei Kommunikationsbeginn dem OPC-UA-Server übermittelt. Es muss an dieser Stelle in der Zertifikatsverwaltung des OPC-UA-Servers vorhanden sein und wird bei Kommunikationsbeginn mit dem vom OPC-UA-Client übermittelten Zertifikat verglichen. Ohne dieses Zertifikat wird eine sichere Kommunikation abgelehnt.

### **UA-Revokationszertifikate für die OPC-UA-Server**

Im projektierbaren Unterverzeichnis "\crl" kann eine Liste nicht vertrauenswürdiger Zertifikate abgelegt werden. Verbindungsaufbauwünsche für gesicherte Verbindungen mit Zertifikaten aus dieser Liste werden abgelehnt.

Die Liste "examplecrl.crl" ist bereits vorinstalliert und kann angepasst werden.

Solche Listen können mit dem OpenSSL-Werkzeug erstellt werden  
(siehe "<http://www.openssl.org/docs/apps/x509.html>").

### **SIMATIC NET S7-UA-Server-Namen**

Der Server-Name und die Server-URI lauten:

```
<ServerUri>urn:Siemens.Automation.SimaticNET.S7:(%guid%)</ServerUri>
<ServerName>OPC.SimaticNET.S7</ServerName>
```

"%guid%" ist ein eindeutiger Bezeichner für den Server. Er besteht aus 28 hexadezimalen Ziffern und hat die Form ABCDEF01-2345-6789-0123456789AB.

Verändern Sie diesen Wert nicht.

### **SIMATIC NET S7OPT-UA-Server-Namen**

Der Server-Name und die Server-URI lauten:

```
<ServerUri>urn:Siemens.Automation.SimaticNET.S7OPT:(%guid%)</ServerUri>
<ServerName>OPC.SimaticNET.S7OPT</ServerName>
```

"%guid%" ist ein eindeutiger Bezeichner für den Server. Er besteht aus 28 hexadezimalen Ziffern und hat die Form ABCDEF01-2345-6789-0123456789AB.

Verändern Sie diesen Wert nicht.

### **SIMATIC NET PROFINET IO-UA-Server-Namen**

Der Server-Name und die Server-URI lauten:

```
<ServerUri>urn:Siemens.Automation.SimaticNET.PNIO:(%guid%)</ServerUri>
<ServerName>OPC.SimaticNET.PNIO</ServerName>
```

"%guid%" ist ein eindeutiger Bezeichner für den Server. Er besteht aus 28 hexadezimalen Ziffern und hat die Form ABCDEF01-2345-6789-0123456789AB.

Verändern Sie diesen Wert nicht.

## SIMATIC NET DP-UA-Server-Namen

Der Server-Name und die Server-URI lauten:

```
<ServerUri>urn:Siemens.Automation.SimaticNET.DP:(%guid%)</ServerUri>
<ServerName>OPC.SimaticNET.DP</ServerName>
```

"%guid%" ist ein eindeutiger Bezeichner für den Server. Er besteht aus 28 hexadezimalen Ziffern und hat die Form ABCDEF01-2345-6789-0123456789AB.

Verändern Sie diesen Wert nicht.

## SIMATIC NET SEND/RECEIVE-UA-Server-Namen

Der Server-Name und die Server-URI lauten:

```
<ServerUri>urn:Siemens.Automation.SimaticNET.SR:(%guid%)</ServerUri>
<ServerName>OPC.SimaticNET.SR</ServerName>
```

"%guid%" ist ein eindeutiger Bezeichner für den Server. Er besteht aus 28 hexadezimalen Ziffern und hat die Form ABCDEF01-2345-6789-0123456789AB.

Verändern Sie diesen Wert nicht.

## Siehe auch

Zertifikatsverwaltung mit grafischer Oberfläche (Seite 518)

### 4.4.3 Zertifikatsverwaltung im OPC-UA-Client "OPC Scout V10"

#### Zertifikatverwaltung für den OPC-UA-Client "OPC Scout V10" projektieren

Der mitgelieferte OPC-UA-Client "OPC Scout V10" verwendet die Windows-Zertifikatsverwaltung.

#### So verwalten Sie Zertifikate für einen Computer:

1. Melden Sie sich als Administrator am System an.
2. Klicken Sie auf "Start" > "Ausführen".  
Die Eingabebox öffnet sich.
3. Geben Sie "mmc" ein und klicken Sie auf "OK".  
Die Konsole öffnet sich.
4. Klicken Sie im Menü "Datei" auf "Snap-In hinzufügen/entfernen..." und anschließend auf "Hinzufügen".  
Das Dialogfeld zur Snap-In-Auswahl öffnet sich.

5. Doppelklicken Sie in der Liste "Snap-In" auf "Zertifikate".  
Markieren Sie im Dialogfeld "Zertifikats-Snap-In" die Option "Computerkonto" und klicken Sie auf "Weiter".
6. Führen Sie im Dialogfeld "Computer auswählen" eine der folgenden Aktionen aus:
  - Um Zertifikate für den lokalen Computer zu verwalten, markieren Sie die Option "Lokalen Computer" und klicken auf "Fertig stellen".
  - Um Zertifikate für einen entfernten Computer zu verwalten, markieren Sie die Option "Anderen Computer", geben den Computernamen ein (oder wählen ihn durch Klicken auf "Durchsuchen" aus) und klicken dann auf "Fertig stellen".
7. Klicken Sie auf "Schließen".  
Der Eintrag "Zertifikate (Computername)" wird in der Liste der ausgewählten Snap-Ins für die neue Konsole angezeigt.
8. Klicken Sie auf "OK", wenn Sie keine weiteren Snap-Ins zur Konsole hinzufügen möchten.
9. Klicken Sie zum Speichern dieser Konsole in der Konsole im Menü "Datei" auf "Speichern" und geben Sie ggf. einen Namen ein.
10. Schließen Sie die Konsole.

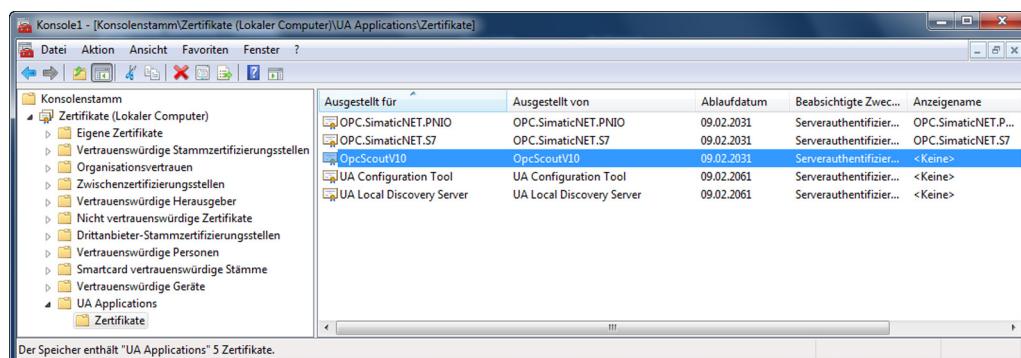


Bild 4-10 Fenster der Konsole: Zertifikatsverwaltung

Der OPC Scout V10 wird im Bereich "Lokaler Computer" unter "UA Applications" unter "OpcScoutV10" geführt und gesucht.

**So exportieren Sie die Zertifikate aus der Zertifikatsverwaltung des Client-Rechners:**

Für die notwendige sichere Kommunikation mit entfernten SIMATIC NET OPC-UA-Servern muss dieses Zertifikat zunächst aus der Zertifikatsverwaltung des Client-Rechners exportiert werden.

1. Öffnen Sie die Konsole.  
Menü "Aktion" > "Alle Aufgaben" > "Exportieren..."
2. Exportieren Sie die Zertifikate ohne privaten Schlüssel.

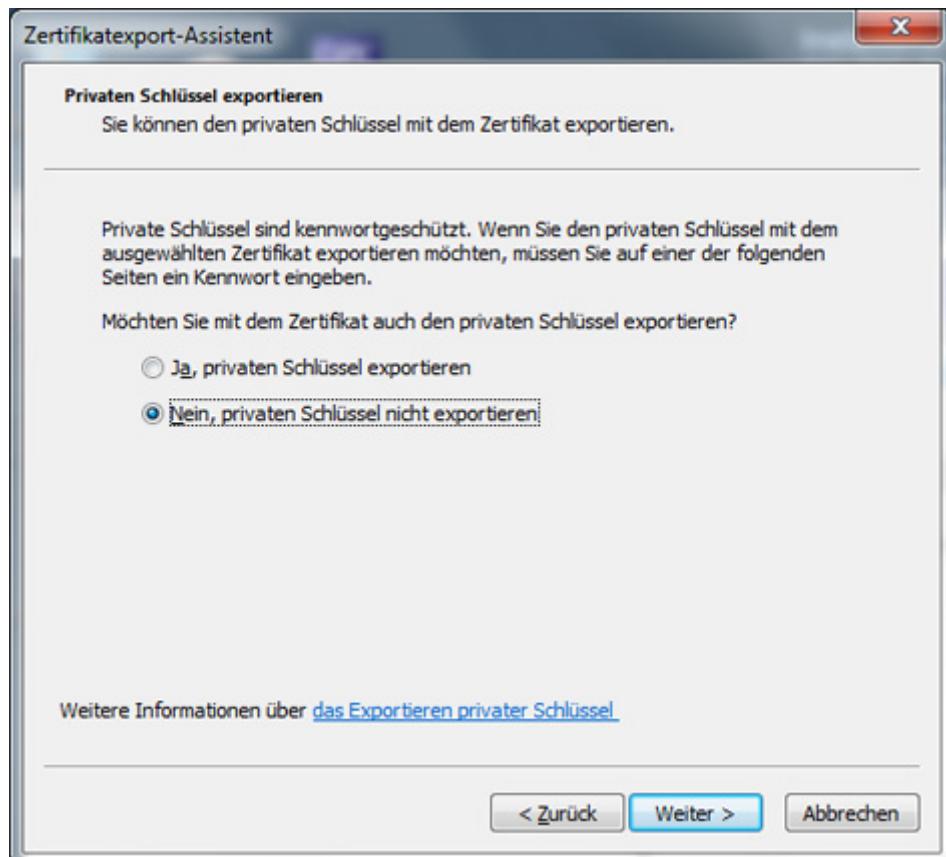


Bild 4-11 Export des Zertifikats ohne privaten Schlüssel

3. Exportieren Sie die Zertifikate im Format "DER-codiert-binär X.509".

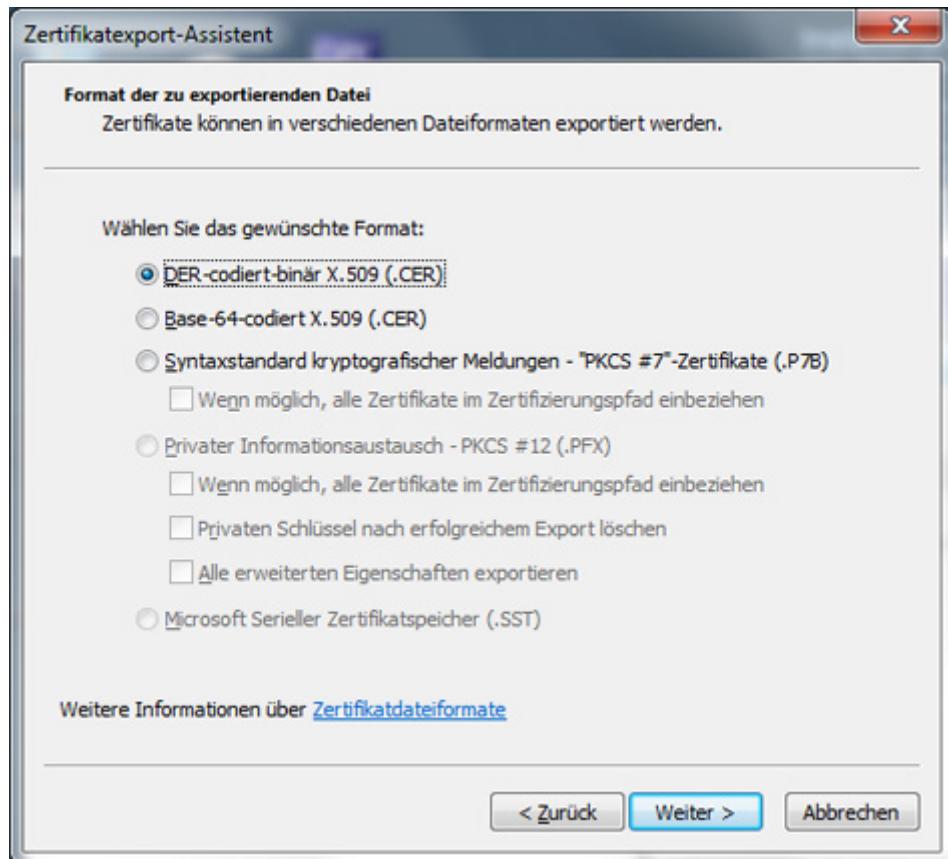


Bild 4-12 Export des Zertifikats im "DER"-Format

4. Klicken Sie nach Auswahl des gewünschten Formats auf "Weiter".

Achten Sie darauf, dass die Datei-Endung im Folgedialog ebenfalls "DER" lauten muss.

Ändern Sie hierzu die Datei-Endung von "\*.cer" in "\*.der".

5. Klicken Sie auf "Weiter" und "Fertigstellen", um den Export abzuschließen.

Die Datei liegt in dem gleichnamigen Verzeichnis, in dem auch die Zertifikate des Server-Rechners liegen.

#### Hinweis

##### Zertifikat-Datei im Format "\*.der"

Falls Sie die Zertifikat-Datei zwar DER-codiert aber versehentlich mit der Endung "\*.cer" erstellt haben, dann müssen Sie die Datei-Endung im Dateiverzeichnis des Client-Rechners von "\*.cer" in "\*.der" umbenennen.

**So kopieren Sie die Zertifikate in einen entfernten Server:**

Für die notwendige sichere Kommunikation mit entfernten SIMATIC NET OPC-UA-Servern muss dieses Zertifikat jetzt in die Zertifikatsverwaltung des entfernten OPC-UA-Servers importiert werden:

1. Kopieren Sie hierzu die soeben exportierte Zertifikat-Datei des Client auf einen geeigneten Datenträger.
2. Kopieren Sie die Zertifikat-Datei des Client von diesem Datenträger in das entsprechende Verzeichnis auf dem Rechner des OPC-UA-Servers (siehe Kapitel "Zertifikatsverwaltung für den OPC-UA-Server (Seite 511)").

Jetzt können Sie eine sichere Verbindung vom Client-Rechner zum OPC-UA-Server aufbauen.

---

**Hinweis**

Im Zertifikat ist der Rechnername (Hostname) kodiert. Beachten Sie, dass bei einer Änderung des Rechnernamens nach der Installation diese Zertifikate ungültig werden. OPC-UA-Kommunikation ist dann nicht mehr möglich. Die OPC-UA-Konfiguration kann jedoch neu erstellt werden, siehe im Kapitel "Zertifikatsverwaltung mit grafischer Oberfläche (Seite 518)".

---

**Hinweis**

Beachten Sie weiterhin, dass im Rechnernamen und damit auch im Zertifikat, nur standardmäßige Zeichen verwendet werden dürfen.

Standardmäßige Namen bestehen aus Buchstaben (A-Z, a-z), Zahlen (0-9) und (-). Wenn Sie andere Zeichen im Rechnernamen verwenden, können keine Zertifikate erzeugt werden.

---

Mit der SIMATIC NET PC Software ab V8.0 gibt es für die OPC-UA-Server eine Zertifikatsverwaltung auch für ankommende UA-Client-Zertifikate. Diese können über eine grafische Oberfläche im Konfigurationsprogramm "Kommunikations-Einstellungen" akzeptiert werden bzw. abgelehnt bleiben. Das Client-Zertifikat eines entfernen Server muss nicht mehr zwingend durch Export und Kopieren manuell vorab übertragen werden. Dies ist nur erforderlich, wenn bereits bei der ersten Verbindungsauftnahme der Client akzeptiert werden soll.

#### 4.4.4 Zertifikatsverwaltung mit grafischer Oberfläche

Ab der SIMATIC NET PC Software V8.0 gibt es für die OPC-UA-Server eine Zertifikatsverwaltung über eine grafische Oberfläche im Konfigurationsprogramm "Kommunikations-Einstellungen".

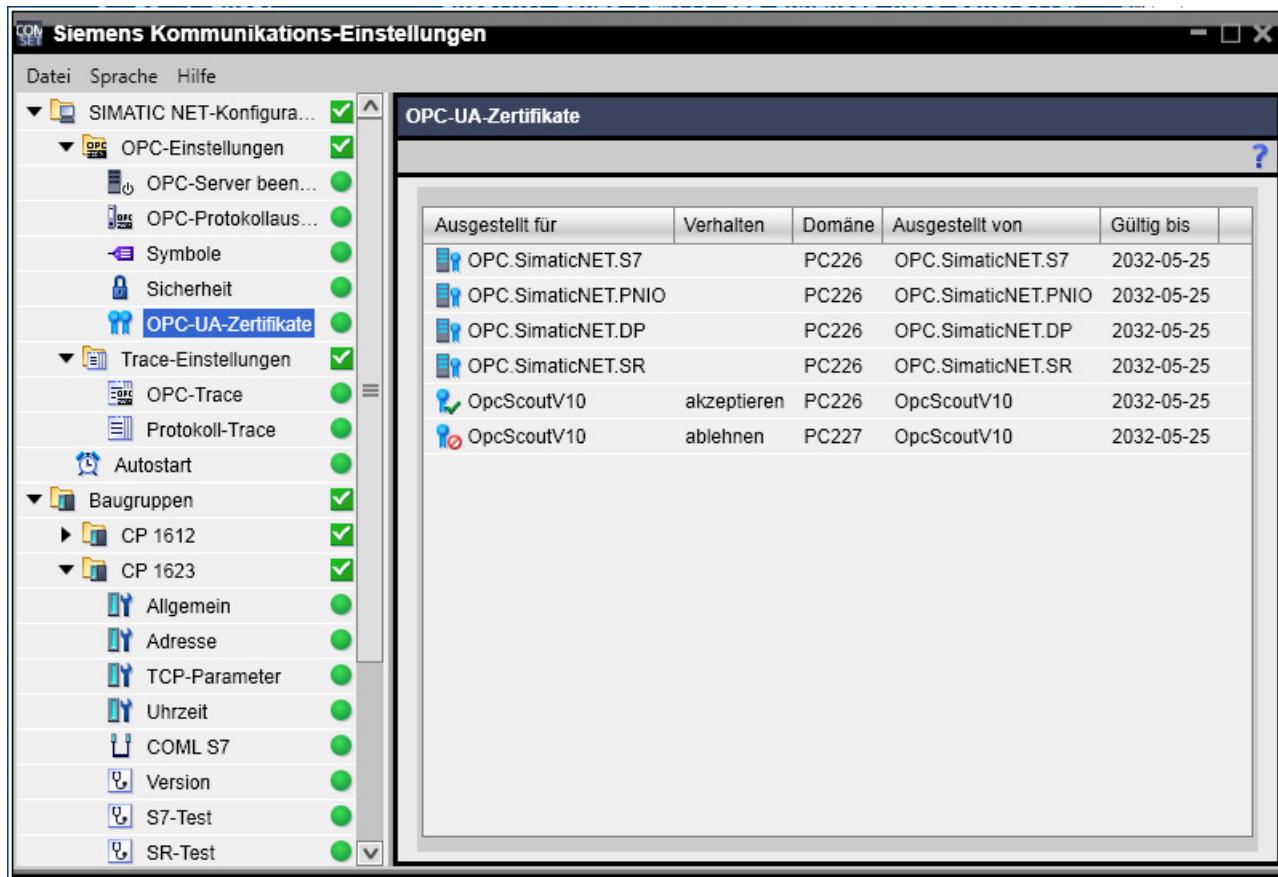


Bild 4-13 Zertifikatsverwaltung im Konfigurationsprogramm "Kommunikations-Einstellungen"

Hier werden die Zertifikate der lokalen OPC-UA-Server und die Zertifikate, mit denen sich OPC-UA-Clients bei den Servern ausgewiesen haben, angezeigt und verwaltet. Es werden auch die Namen des Rechners angegeben, auf dem die Software-Komponente für das Zertifikat ausgegeben wurde. Weiterhin kann die Gültigkeitsdauer und das Zertifikat selbst inhaltlich angezeigt werden. Es können Client-Zertifikate importiert und Server-Zertifikate als DER-codierte Binärdatei (X.509) exportiert werden.

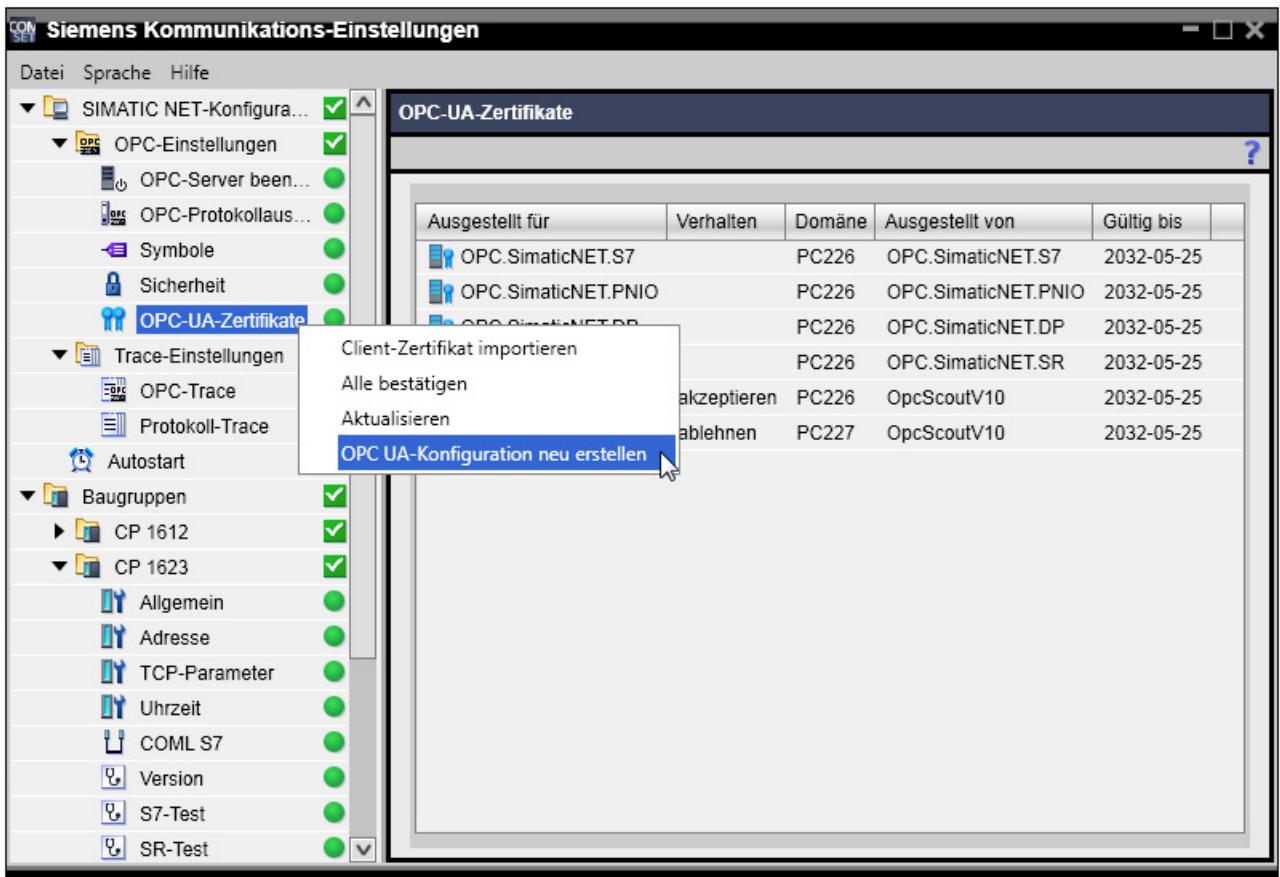


Bild 4-14 Zertifikate konfigurieren im Konfigurationsprogramm "Kommunikations-Einstellungen"

Die lokale OPC-UA-Konfiguration kann z.B. im Falle einer Änderung des Rechnernamens neu erstellt werden. Hierbei werden

- die Konfigurationsdateien der lokalen OPC-UA-Server für S7, S7 Optimiert, PROFINET IO, DP und SEND/RECEIVE neu erstellt (Anpassung an den Rechnernamen).
- die OPC-UA-Server-Zertifikate gelöscht und neu erstellt.
- das OPC-UA-Client-Zertifikat für den lokalen OPC Scout V10 gelöscht und neu erstellt.
- das Zertifikat für den OPC-UA-Discovery-Service neu aus dem Windows-Zertifikatspeicher exportiert.
- die OPC-UA-Server beendet und neu gestartet.

#### Hinweis

Die laufende SIMATIC NET-Kommunikation sollte daher vor dem Ausführen dieser Funktion beendet werden.

## **4.4.5        OPC-UA-Dienste**

### **UA-Dienste**

Die OPC-UA-Dienste sind in logische Funktiongruppen aufgeteilt. Die wichtigsten Dienste und Methoden werden in diesem Abschnitt beschrieben. Weitere Details entnehmen Sie den OPC-UA-Spezifikationen unter "<http://www.opcfoundation.org/UA>".

#### **Discovery-Dienste**

Mit dem Discovery-Diensten kann ein UA-Client die konfigurierten Endpunkte und Sicherheitsanforderungen eines OPC-UA-Servers auffinden:

- FindServers

Dieser Dienst gibt die OPC-UA-Server zurück, die dem Discovery-Server bekannt sind.

- GetEndpoints

Dieser Dienst gibt die Endpunkte eines Servers und deren benötigte Sicherheitsanforderungen zurück, um einen sicheren Kanal und eine Session zu verbinden.

#### **Sicherheitsdienste**

Die Sicherheitsdienste erlauben den Aufbau eines sicheren Kommunikationskanals, der die Integrität und Vertraulichkeit von ausgetauschten Nachrichten zwischen Server und Client gewährleistet.

- OpenSecureChannel

Dieser Dienst wird benutzt, um einen sicheren Kanal zu öffnen oder zu erneuern. Die Zertifikate von Client und Server werden dabei ausgetauscht. Details sind in "<http://www.opcfoundation.org/UA>" unter "Downloads" > "Specifications" > "Part 6" zu finden.

- CloseSecureChannel

Dieser Dienst wird benutzt, um einen sicheren Kanal zu schließen.

## Session-Dienste

Die Session-Dienste erlauben einem Client, den Benutzer zu authentifizieren, welcher die Session benutzen will. Weiterhin können Sessions verwaltet werden.

- CreateSession

Dieser Dienst wird von einem OPC-UA-Client benutzt, um eine Session zu erzeugen. Der Server liefert dabei zwei Werte, welche die Session eindeutig identifizieren:

- Der erste Wert ist die "SessionId", um die Session in Protokolldateien und im Namensraum eindeutig zu kennzeichnen.
- Der zweite Wert ist der "AuthenticationToken", welcher eingehende Anfragen einer Session zuordnet.

Vor "CreateSession" sollte "OpenSecureChannel" aufgerufen werden, um einen sicheren Kanal zu erzeugen. Der sichere Kanal sollte vom Server mit dem "AuthenticationToken" assoziiert werden, um nur Anfragen in dieser Session zu bedienen, die mit der von dem sicheren Kanal erzeugten Session in Verbindung gebracht werden können.

- ActivateSession

Dieser Dienst wird vom Client genutzt, um Zertifikate und die Identität des Benutzers der Session an den Server zu übermitteln. Die Zertifikate des sicheren Kanals sollten ebenfalls dieselben sein wie diejenigen, die in der Session verwendet werden. Client und Server sollten dies überprüfen und sicherstellen.

- CloseSession

Dieser Dienst wird benutzt, um eine Session zu schließen

- Cancel

Mit diesem Dienst können ausstehende Anfragen abgebrochen werden.

## Ansichts-Dienste

Mit diesen Diensten kann der Namensraum durchsucht werden. Knoten, die mehrfach geschrieben oder gelesen werden sollen, können registriert werden.

- Browse

Dieser Dienst dient zum Auslesen der Referenzen eines Knotens.

- BrowseNext

Dieser Dienst dient zum Anfordern eines nächsten Satzes von "Browse"- oder "BrowseNext"-Antworten, falls eine Antwort nicht in ein einfaches Antworttelegramm gepasst hat.

- TranslateBrowsePathsNodeID

Mit diesem Dienst werden Browse-Pfade in NodeIDs übersetzt. Jeder Browse-Pfad beginnt mit einem Start-Knoten und einem relativen Pfad. Der relative Pfad enthält eine Sequenz von Referenz-Typen und Browse-Namen.

- RegisterNodes

Der Dienst "RegisterNodes" sollte benutzt werden, wenn auf Knoten mehrfach oder wiederholt zugegriffen werden soll (z.B. Werte schreiben oder lesen). Dieser Dienst erlaubt dem Server, sich auf weitere Zugriffe vorzubereiten und nachfolgende Zugriffe

wesentlich effizienter durchzuführen. Damit wird eine wesentliche Performance-Steigerung im OPC-UA-Server erreicht. Weitere Details zu der Verwendung von RegisterNodes entnehmen Sie der OPC-UA-Spezifikation unter "http://www.opcfoundation.org/UA".

- UnregisterNodes

Mit diesem Dienst können registrierte Knoten wieder deregistriert werden.

### **Attribut- (Variablen-) Dienste**

Die Attribut-Dienste dienen dem Lesen und Schreiben der Attribute von Knoten. Da der Wert einer Variablen als Attribut modelliert ist, dienen diese Dienste zum Lesen und Schreiben von Werten von Variablen.

- Read

Mit diesem Dienst können ein oder mehr Attribute von einem oder mehreren Knoten gelesen werden. Von Feldern können auch Feld-Elemente oder Bereiche separat gelesen werden. Strukturierte Attributelemente können indiziert oder bereichsweise gelesen werden. Der Parameter "maxAge" weist den Server an, vom Gerät oder von einer lokal gespeicherten Kopie der Daten, die nicht älter ist als maxAge, zu lesen.

- Write

Dieser Dienst dient dem Schreiben ein oder mehrerer Attribute von einem oder mehreren Knoten. Von Feldern können auch Feld-Elemente oder Bereiche separat geschrieben werden. Strukturierte Attributelemente können indiziert oder bereichsweise geschrieben werden.

### **Methoden-Dienste**

Mit diesem Dienst können Methoden aufgerufen werden. Es können Eingangs-Parameter übergeben und Ausgabe-Parameter erhalten werden.

- Call

Dieser Dienst wird benutzt, um eine Liste von Methoden aufrufen.

### **Anmeldungs-Dienste**

Die Anmeldungs-Dienste erlauben dem Client, Subscriptions zu erzeugen, zu verändern und zu löschen. Subscriptions senden Notifications - generiert durch MonitoredItems - an den Client.

- CreateSubscription

Dieser Dienst wird benutzt, um eine Subscription zu erzeugen. Subscriptions beobachten eine Anzahl von MonitoredItems für Notifications und liefern diese an den Client als Antwort auf einen Publish Auftrag.

- ModifySubscription

Dieser Dienst wird benutzt, um eine Subscription zu verändern.

- SetPublishingMode

Dieser Dienst aktiviert das Senden von Notifications an eine oder mehrere Subscriptions.

- Publish

Dieser Dienst hat zwei Verwendungen:

- Erstens, den Empfang von Notifications für eine oder mehr Subscriptions zu bestätigen.
- Zweitens wird damit vom Server eine Notification oder eine "keep-alive"-Meldung angefordert.

- Republish

Dieser Dienst fragt bei der Subscription eine Wiederholung der Notification an.

- TransferSubscription

Mit diesem Dienst können Subscriptions und deren MonitoredItems von einer Session zu einer anderen transferiert werden. Dies geht sowohl innerhalb eines Client als auch zwischen Clients.

- DeleteSubscription

Dieser Dienst wird benutzt, um eine Subscription zu löschen.

## Beobachtungsdienste

Die Beobachtungsdienste werden zusammen mit den Anmeldungs-Diensten benutzt, um sich für die Beobachtung von Knoten im Namensraum anzumelden.

Die Beobachtungsdienste definieren Dienste zum Erzeugen, Verändern und Löschen von Item-Listen, die benötigt werden, um Attribute auf Datenänderungen oder Objekte auf Ereignisse zu überwachen.

- CreateMonitoredItems

Mit diesem Dienst können ein oder mehrere MonitoredItems einer Subscription zugeordnet werden.

- ModifyMonitoredItems

Mit diesem Dienst können ein oder mehrere MonitoredItems einer Subscription verändert werden.

- SetMonitoringMode

Dieser Dienst wird benötigt, um den Beobachtungs-Modus ein oder mehrerer MonitoredItems einer Subscription zu setzen.

- DeleteMonitoredItems

Mit diesem Dienst können ein oder mehrere MonitoredItems einer Subscription gelöscht werden.

## **4.4.6 OPC-UA-Clients erstellen**

### **4.4.6.1 Schnittstellen unter OPC UA**

#### **Die Programmierschnittstellen unter OPC UA**

Für die Erstellung von OPC-UA-Clients sollte die Programmierschnittstelle abhängig vom Anwendungsfall und der gewünschten Zertifikatverwaltung gewählt werden:

- Die C-Schnittstelle für einen einfachen performanten Zugriff und eine OpenSSL-Zertifikatverwaltung
- Die .NET-Schnittstelle mit komfortabler Verwendung und Zugriff auf die Windows-Zertifikatverwaltung

---

#### **Hinweis**

In Kapitel "OPC-UA-Schnittstelle in C (Seite 736)" finden Sie die entsprechenden Beispielprogramme.

---

### **4.4.6.2 Die C-Schnittstelle unter OPC UA**

#### **Die C-Schnittstelle**

Die OPC-UA-C-Schnittstelle eignet sich für OPC-UA-Client-Anwendungen, die auch bei Bedarf einfach auf andere Systeme portiert werden können.

Die benötigten Header-Dateien befinden sich nach Installation in folgendem Verzeichnis:

"<Installationsprogrammpfad>\SIMATIC.NET\opc2\inc\"

Die benötigte Import-Bibliothek "uastack.lib" und die ladbare Bibliothek "uastack.dll" befinden sich in folgenden Verzeichnissen:

"<Installationsprogrammpfad>\SIMATIC.NET\opc2\bins7\"  
"<Installationsprogrammpfad>\SIMATIC.NET\opc2\binpnio\"  
"<Installationsprogrammpfad>\SIMATIC.NET\opc2\bindp\"  
"<Installationsprogrammpfad>\SIMATIC.NET\opc2\binsr\"  
"<Installationsprogrammpfad>\SIMATIC.NET\opc2\bins7opt\"

Weiterhin wird für die Zertifikatbehandlung und Sicherheit die OpenSSL-Import-Bibliothek "libeay32.lib" und die ladbare Bibliothek "libeay32.dll" benötigt. Diese befinden sich ebenfalls in den oben genannten Verzeichnissen.

Damit lässt sich eine sichere OPC-UA-C-Client-Anwendung erstellen.

#### Hinweis

Um weitere Informationen zur Implementierung zu erhalten, verwenden Sie die Beispiel-Implementierung von der OPC Foundation.

### 4.4.6.3 Die .NET-Schnittstelle unter OPC UA

#### Die .NET-Schnittstelle

Die OPC-UA-.NET-Schnittstelle eignet sich für einfach programmierbare komfortable OPC-UA-Client-Anwendungen. Der OPC Scout V10 verwendet z.B. die UA-.NET-Schnittstelle.

Die benötigten Assembly-Dateien befinden sich nach Installation in folgendem Verzeichnis:

"<Installationsprogrammpfad>\SIMATIC.NET\opc2\bin\"

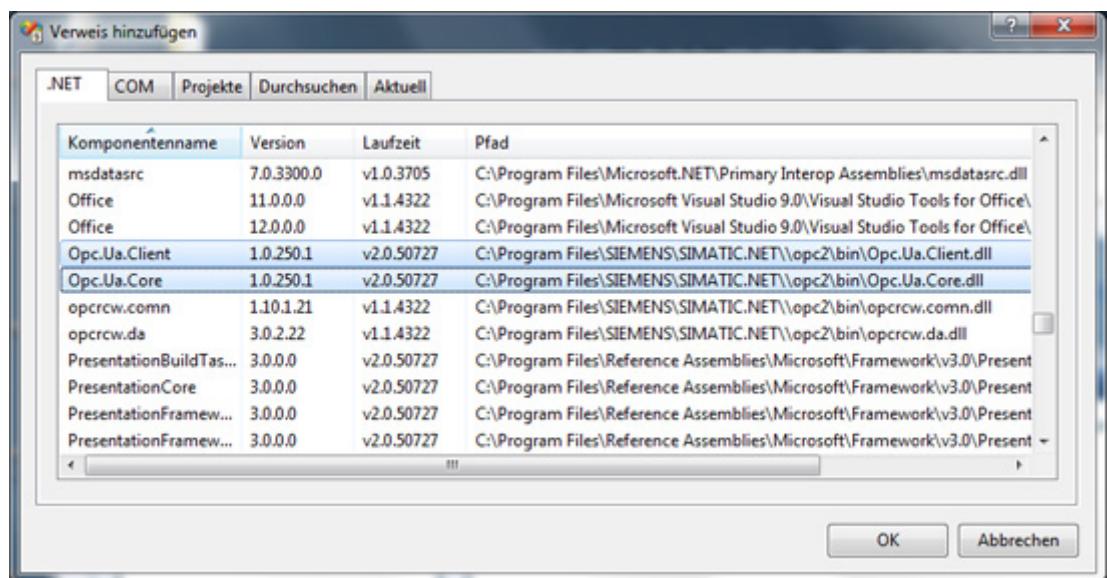


Bild 4-15 Assembly-Dateien der .NET-Schnittstelle

Folgende Komponenten werden benötigt:

- Die UA-Client-Komponente "Opc.Ua.Client.dll"
- Die UA-Core-Komponente "Opc.Ua.Core.dll"

Es können auch die entsprechenden OPC-UA-Komponenten im OPC-Foundation-Verzeichnis verwendet werden, das durch die SIMATIC NET-CD installiert wird. Sie finden es in folgendem Verzeichnis:

"<Installationprogrammpfad>\OPC Foundation\UA\V1.0\Bin"

#### 4.4.7 Meldungen der OPC-UA-Server

##### Systemmeldungen

Die SIMATIC NET OPC-UA-Server geben Systemmeldungen aus, die an der Schnittstelle mit dem Fehler-Code und im OPC Scout V10 mit der Symbolischen ID ausgegeben werden.

Die folgende Tabelle listet die wichtigsten Systemmeldungen auf.

Weitere Meldungen finden Sie in den Header-Dateien von OPC UA oder in der Spezifikation der OPC Foundation ("<http://opcfoundation.org/UA>") unter Part 4 und Part 6.

Tabelle 4- 1 Systemmeldungen des SIMATIC NET OPC-UA-Server

Symbolische ID	Fehler-Code (hex)	Bedeutung
Good	0x00000000	Gut - alles wie erwartet
Good_Overload	0x002F0000	Die laufende Aufzeichnung von Datenwerten hat sich auf Grund von Ressourcenengpässen verlangsamt.
Good_Clamped	0x00300000	Der Wert wurde angenommen und geschrieben. Dabei überschritt der Sensor (Daten-Ziel) den zulässigen Stellbereich.
Uncertain	0x40000000	Unsicher - ohne weitere Details
Bad	0x80000000	Ungültig - ohne weitere Details
Bad_NoCommunication	0x80310000	Der Kommunikationsendpunkt ist definiert. Die Kommunikation selbst ist nicht aufgebaut und es gibt keinen letzten bekannten Wert. Der Status / Substatus bezieht sich auf Cache-Werte vor Empfang des ersten Werts.
Bad_NodeIdInvalid	0x80330000	Die Syntax der Nodeld ist ungültig.
Bad_NodeIdUnknown	0x80340000	Die ID des Knotens referenziert einen Knoten, den es im Server-Adressraum nicht gibt.
Bad_NodeClassInvalid	0x805F0000	Die Klasse des Knotens ist ungültig.
Bad_SourceNodeIdInvalid	0x80640000	Die ID des Quellknotens referenziert keinen gültigen Knoten.
Bad_TargetNodeIdInvalid	0x80650000	Der Zielknoten gibt keinen gültigen Knoten an.
Bad_NoDeleteRights	0x80690000	Der Server erlaubt nicht das Löschen des Knotens.
Bad_HistoryOperationInvalid	0x80710000	Der Parameter "history details" (Details der Historie) ist ungültig.
Bad_HistoryOperationUnsupported	0x80720000	Der Server unterstützt nicht den angeforderten Vorgang.
Bad_IndexRangeInvalid	0x80360000	Die Syntax des Bereichsparameters des Index ist ungültig.
Bad_IndexRangeNoData	0x80370000	Innerhalb der spezifizierten Indizes gibt es keine Daten.
Bad_DataEncodingInvalid	0x80380000	Die Datenverschlüsselung ist ungültig.
Bad_DataEncodingUnsupported	0x80390000	Der Server unterstützt nicht die angeforderte Datenverschlüsselung für den Knoten.
Bad_NotReadable	0x803A0000	Die Sicherheitstufe erlaubt nicht das Lesen oder Anmelden am Knoten.
Bad_NotWritable	0x803B0000	Die Sicherheitstufe erlaubt nicht das Schreiben des Knotens.
Bad_OutOfRange	0x803C0000	Der Wert ist außerhalb des zulässigen Bereichs.
Bad_NotSupported	0x803D0000	Der angeforderte Vorgang wird nicht unterstützt.

Symbolische ID	Fehler-Code (hex)	Bedeutung
Bad_NotFound	0x803E0000	Ein angefordertes Element wurde nicht gefunden oder der Suchvorgang wurde ohne Erfolg beendet.
Bad_ObjectDeleted	0x803F0000	Das Objekt kann nicht verwendet werden, weil es gelöscht wurde.
Bad_NotImplemented	0x80400000	Der angeforderte Vorgang ist nicht implementiert.
Bad_MonitoringModelError	0x80410000	Der Beobachtungsmodus ist ungültig.
Bad_MonitoredItemModelError	0x80420000	Die ID des beobachtenden Items referenziert kein gültiges beobachtetes Item (MonitoredItem).
Bad_MonitoredItemFilterModelError	0x80430000	Der Parameter für die Filterbedingung des beobachteten Items (MonitoredItem) ist ungültig.
Bad_MonitoredItemFilterUnsupported	0x80440000	Die Filterbedingung des beobachteten Items (MonitoredItem) wird vom Server nicht unterstützt.
Bad_StructureMissing	0x80460000	Ein erforderlicher strukturierter Parameter fehlt oder ist gleich Null.
Bad_EventFilterModelError	0x80470000	Die Ereignis-Filterbedingung ist ungültig.
Bad_ContentFilterModelError	0x80480000	Die Inhalt-Filterbedingung ist ungültig.
Bad_FilterOperandModelError	0x80490000	Der Operand, der in einer Inhalt-Filterbedingung verwendet wurde, ist ungültig.
Bad_ContinuationPointModelError	0x804A0000	Der Rückgabewert des Fortsetzungspunkts (continuation point) ist weiterhin gültig.
Bad_NoContinuationPoints	0x804B0000	Der angeforderte Auftrag wurde auf Grund des Löschens aller Fortsetzungspunkte nicht ausgeführt.
Bad_ReferenceTypeIDModelError	0x804C0000	Der Referenztyp verweist nicht auf einen gültigen Referenz-Typ-Knoten.
Bad_BrowseDirectionModelError	0x804D0000	Die Browse-Richtung ist nicht mehr gültig.
Bad_NodeNotInView	0x804E0000	Der Knoten ist nicht Bestandteil der Ansicht.

#### 4.4.8 Migration von OPC Data Access / Alarms & Events nach OPC UA

Welche Dienste unterscheiden sich bei OPC Data Access / Alarms & Events und OPC Unified Architecture?

Für den Fall, dass Sie Ihre bestehende OPC DA / AE-Applikation nach OPC UA migrieren wollen, finden Sie hier eine Übersicht der korrespondierenden Dienste bei beiden Systemen.

OPC Data Access / OPC Alarms & Events	OPC Unified Architecture
CoCreateInstanceEx()	OpenSecureChannel() CreateSession() ActivateSession()
ChangeBrowsePosition() BrowseOPCItemIDs() GetItemID() QueryAvailableProperties() GetItemProperties()	Browse() Read()

OPC Data Access / OPC Alarms & Events	OPC Unified Architecture
<b>Zugriff über den Item-Namen:</b> Read() (DA 3.0) Write() (DA 3.0)	<b>Zugriff über die Knoten-ID:</b> Read() Write()
<b>Zugriff auf Items einer Gruppe über ein Handle:</b> AddItems() Read(...,handle,...) Write (... ,handle,...) RemoveItems()	<b>Zugriff auf registrierte Knoten über ein Handle:</b> RegisterNodes() Read(...,handle,...) Write (... ,handle,...) UnregisterNodes()
AddGroup() SetState() RemoveGroup()	CreateSubscription() ModifySubscription() DeleteSubscription()
AddItems() RemoveItems()	CreateMonitoredItems() DeleteMonitoredItems()
DataChange()	Publish()

Der SIMATIC NET OPC-Server unterstützt weiterhin die Syntax der bisherigen Spezifikationen wie beispielsweise "OPC Data Access" oder "OPC Alarms & Events". Dies vereinfacht die Migration nach OPC UA.

## 4.5 OPC-UA-Redundanz-Schnittstelle programmieren

### Transparente redundante OPC-UA-Server

Redundante OPC-UA-Server werden im Normalbetrieb zur Lastverteilung genutzt. Fällt einer der OPC-UA-Server aus, wird dessen Kommunikationsleistung vom redundanten OPC-UA-Server übernommen.

Transparent bedeutet, dass gegenüber dem OPC-UA-Client die in einer Redundanzgruppe konfigurierten OPC-UA-Server wie ein einziger, ausfallsicherer OPC-UA-Server erscheint. Der OPC-UA-Client verwendet eine einzige Cluster-IP-Adresse zum Ansprechen der OPC-UA-Server einer Redundanzgruppe. Der Datenabgleich zwischen den OPC-UA-Servern sorgt beim Ausfall eines OPC-UA-Servers für eine automatische Übernahme des aktiven OPC-UA-Server weitgehend ohne Aktion seitens des OPC-UA-Client.

### Funktionsweise transparenter redundanter OPC-UA-Server

In der Standardautomatisierungstechnik können Sie mithilfe der OPC-UA-Redundanz-Spezifikation einfache transparente Server-Redundanz einsetzen. Ein Standard-UA-Client benötigt dazu keine zusätzliche proprietäre Redundanz-Software.

Bei der Verwendung transparenter redundanter Server werden die Aufgaben der OPC-UA-Clients gleichmäßig auf die OPC-UA-Server verteilt, sofern kein Server ausgefallen ist. Dadurch wird ein Lastausgleich sichergestellt. Die gleichmäßige Aufgabenverteilung des Client auf die beiden OPC-UA-Server (siehe Abbildung) ermöglicht eine bessere Ausnutzung der Ressourcen.

Die OPC-UA-Client-Sessions und -Subscriptions werden regelmäßig zwischen den OPC-UA-Servern abgeglichen. Die Synchronisation der Server erfolgt dabei über Industrial-

Ethernet-Netzwerkzugänge mit 10/100/1000 Mbit/s. Aus diesem Grund ist keine Extraverkabelung für die Synchronisation erforderlich. Wird die Verbindung zwischen Client und einem der OPC-UA-Server unterbrochen, übernimmt der 2. OPC-UA-Server dessen Aufgabe. Dabei werden die Sessions und Subscriptions nur kurz unterbrochen, wodurch ein schnelles Weiterarbeiten ermöglicht wird.

Die OPC-UA-Redundanz-Server können Sie mit STEP 7 zentral projektieren. Dies ermöglicht Ihnen eine Projektierung mit wenigen Konfigurationsangaben durchzuführen. Zertifikate werden bei Bedarf erstellt und verteilt.

Die S7-Protokollanbindung zwischen den Servern ist nicht redundant ausgelegt. D. h. bei Ausfall eines Servers werden Daten vom 2. Server neu gelesen.

Die nachfolgende Abbildung zeigt grafisch eine Lastverteilung bei der Verwendung von zwei OPC-UA-Servern.

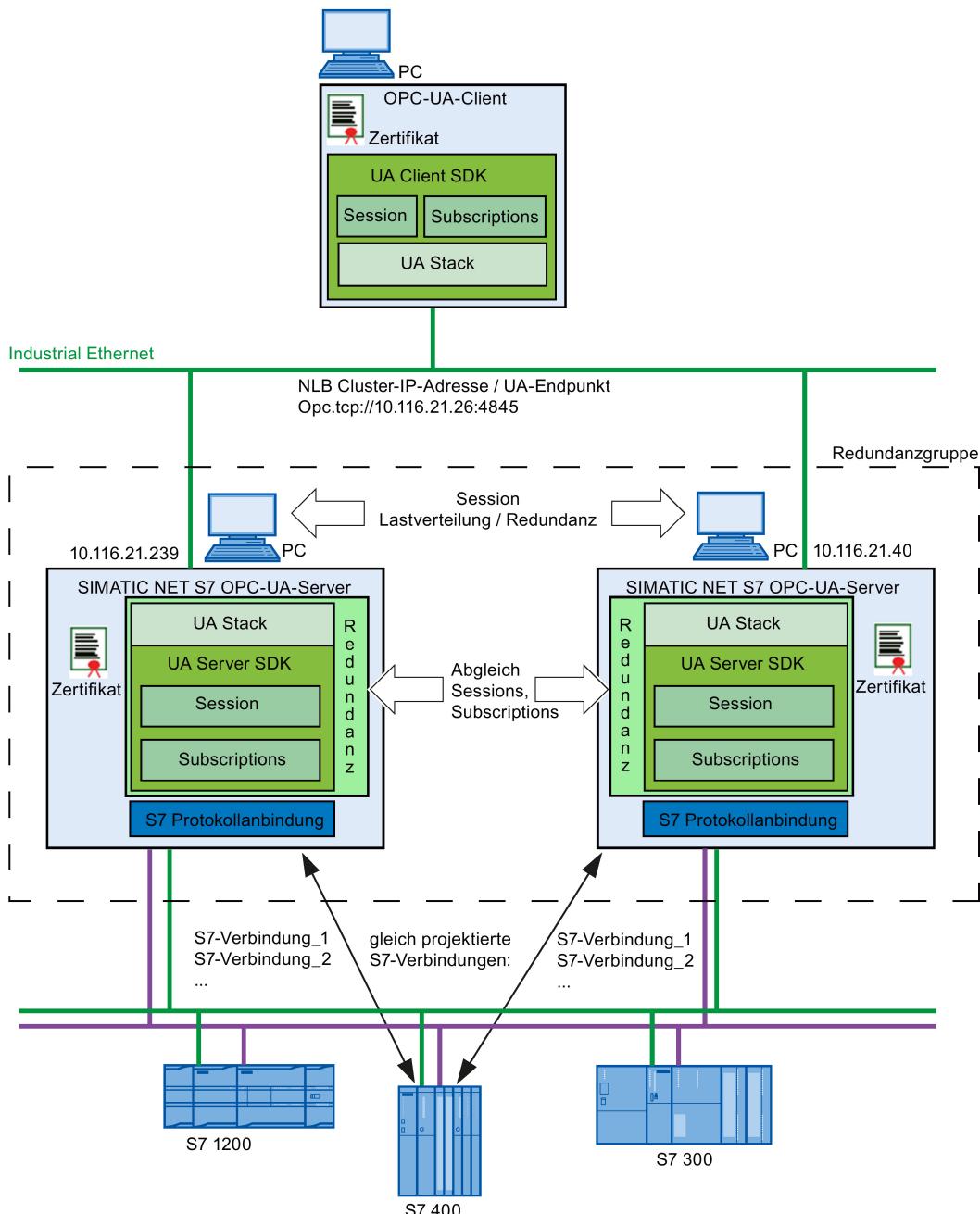


Bild 4-16 Schematischer Aufbau eines redundanten Netzwerks

Das Windows Feature "Netzwerklastenausgleich" (NLB) ist Voraussetzung für die transparente OPC-UA-Redundanz. Da der NLB nur mit Windows-Server-Betriebssystemen verfügbar ist, wird als Mindestanforderung ein Windows-Server-Betriebssystem benötigt. Ab der SIMATIC NET PC Software V8.1 wird hierfür "Windows Server 2008 R2 64 Bit" benötigt.

### Das Feature "Netzwerklastenausgleich" (NLB)

Das Windows Feature "Netzwerklastenausgleich" (NLB) gewährleistet eine Lastverteilung und Redundanz mehrerer Server-PCs.

Durch den NLB können Sie verschiedene OPC-UA-Server-PCs im Netzwerk unter einer Netzwerkadresse anmelden. Um dies zu ermöglichen, stellt der NLB für einen Client eine Cluster-IP-Adresse zur Verfügung, hinter der bis zu 32 physikalische Server-PCs zusammengefasst werden können. Beim Aufbau einer neuen TCP/IP-Verbindung werden diese gleichmäßig auf die verfügbaren Server verteilt. Dadurch werden Lastverteilung und eine hohe Verfügbarkeit gewährleistet. Eine spezielle Konfiguration des Client für die Redundanz ist nicht notwendig.

Fällt der OPC-UA-Server aus, zu dem eine Verbindung besteht, erkennt der Client, abhängig von der konfigurierten Überwachungszeit, die Verbindungsunterbrechung. Im von der OPC-UA-Spezifikation definierten Reconnect-Szenario wird automatisch mithilfe des NLB eine neue TCP/IP-Verbindung zum redundanten Server aufgebaut.

Der Abgleich von Session- und Subscription-Informationen zwischen den OPC-UA-Servern eines Redundanzsets erfolgt erst nach der Rückmeldung an den OPC-UA-Client und erfordert auch einige Zeit (siehe Abbildung oben). Der Abgleich erfolgt über Industrial Ethernet mithilfe einer OPC-UA-Verbindung mit separaten Ports. Dadurch erhält der Client, direkt nach dem TCP/IP-Reconnect, wieder Daten, ohne dass er neue Subscriptions anlegen muss. Der HMI-Bediener erhält während der Umschaltzeit keine Updates der Daten. Außerdem wird die Qualität der Daten bei den meisten OPC-UA-Clients zwischen Erkennen des Verbindungsfehlers und dem Reconnect kurz auf "BAD" gesetzt. Der Ausfall eines OPC-UA-Servers sieht für den OPC-UA-Client nur wie ein Abbruch der TCP/IP-Verbindung aus. Wenn die Verbindung zwischen den OPC-UA-Servern gestört ist, kann es vorkommen, dass bei einer Umschaltung der Verbindung des OPC-UA-Client zum redundanten OPC-UA-Server, bei diesem noch nicht alle Informationen über neu angelegte Sessions und Subscriptions vorliegen.

### Randbedingung

Um diese Funktion zu ermöglichen, müssen alle Server identisch projektiert sein. Genauere Informationen zur identischen Projektierung finden Sie im Kapitel "Projektierung der Redundanzgruppe (Seite 538)" unter "Konsistenz der OPC-UA-Server sicherstellen".

### Hinweise und Einschränkungen für den Umgang mit "Netzwerklastenausgleich" (NLB)

Die NLB-Portregeln sind maßgeblich für das Verhalten der Server beim Lastausgleich. In den Standardeinstellungen sind Protokolle auf "Beide" und Filterungsmodus auf "Mehrfachhost" mit Affinität "Einfach" eingestellt. Mit dem Ändern dieser Einstellungen kann das Verhalten des OPC-S7-Redundanzserver zum Lastausgleich verändert werden. Zum Beispiel wenn Protokolle auf "TCP" und Filterungsmodus gleichzeitig auf "Mehrfachhost" mit der Option "Keine" konfiguriert wird. Prüfen Sie dabei auch die Rückwirkungen auf andere Server-Applikationen, die NLB nutzen.

#### **4.5.1      Installieren und Konfigurieren des Feature "Netzwerklastenausgleich"**

Um einen redundanten Serverbetrieb zu ermöglichen, wird ab der SIMATIC NET PC Software V8.1 das Betriebssystem "Windows Server 2008 R2 64 Bit" mit dem Feature "Netzwerklastenausgleich" (NLB) benötigt.

##### **Netzwerklastenausgleich installieren**

Um das Feature zu installieren, gehen Sie folgendermaßen vor:

1. Starten Sie die Installation.

Klicken Sie auf "Start" > "Systemsteuerung" > "Verwaltung" > "Server Manager" > "Features", "Features hinzufügen"

Reaktion: Das Dialogfenster "Assistent zum Hinzufügen von Features" wird geöffnet.

2. Wählen Sie das Feature "Netzwerklastenausgleich" aus und klicken Sie auf "Installieren".
3. Folgen Sie den Installationsanweisungen.

##### **Netzwerklastenausgleich konfigurieren**

Um das Feature "Netzwerklastenausgleich" (NLB) zu konfigurieren, starten Sie den "Netzwerklastenausgleich-Manager". Gehen Sie dazu folgendermaßen vor:

1. Starten Sie den "Netzwerklastenausgleich-Manager".

Klicken Sie auf "Start" > "Systemsteuerung" > "Verwaltung" > "Server Manager" > "Netzwerklastenausgleich-Manager".

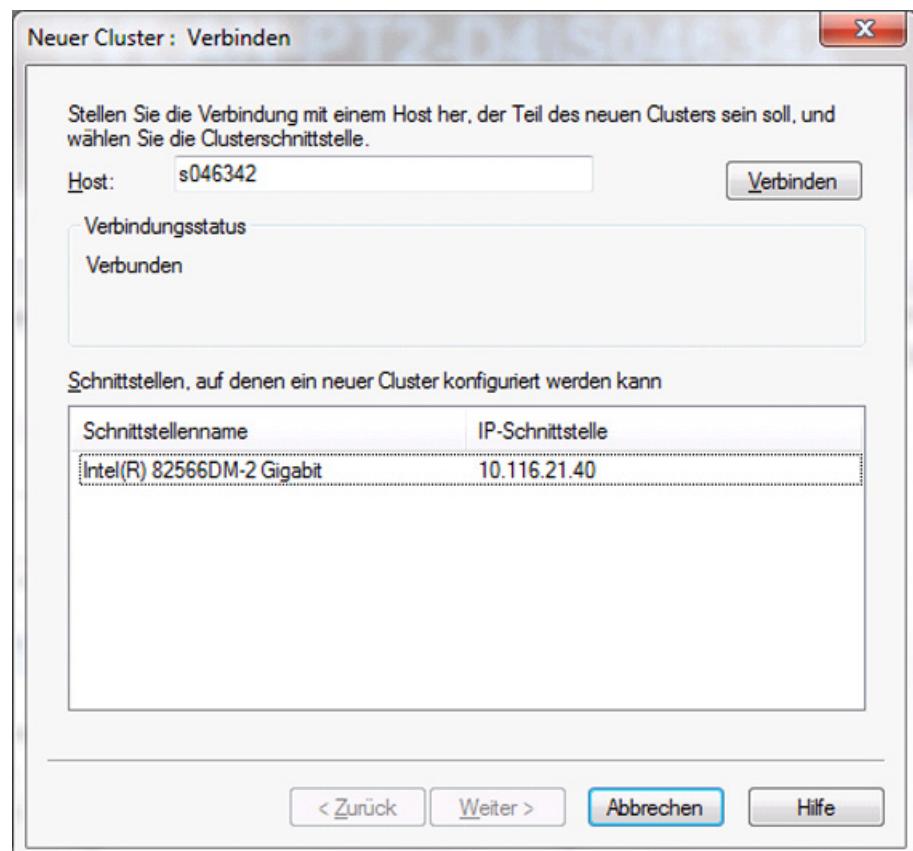
2. Erstellen Sie eine neue Redundanzgruppe (Cluster).

Klicken Sie im Menü auf "Neu" > "Cluster"

Reaktion: Das Dialogfenster "Neuer Cluster: Verbinden" wird geöffnet.

3. Verbinden Sie den Cluster zuerst mit dem lokalen Host und den lokalen Ethernet-Schnittstellen (siehe nachfolgende Abbildung).

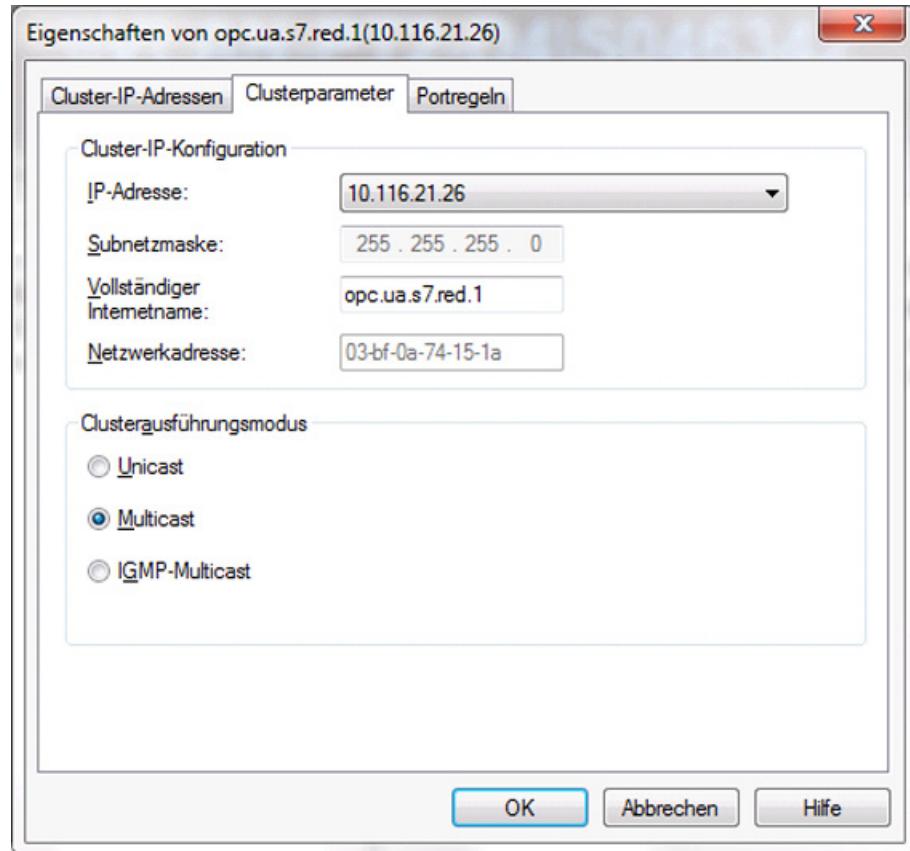
Geben Sie dazu in dem Eingabefeld "Host:" den Namen des Hosts an und klicken Sie auf "Verbinden".



4. Öffnen Sie das Dialogfenster "Eigenschaften".

Klicken Sie mit der rechten Maustaste auf den Schnittstellennamen und wählen Sie im Kontextmenü "Eigenschaften" aus.

Reaktion: Das Dialogfenster "Eigenschaften von..." wird geöffnet.



5. Konfigurieren Sie dann die IP-Adresse und den Internetnamen des Clusters (siehe Abbildung).

Es wird empfohlen den Clusterausführungsmodus "Multicast" oder "IGMP-Multicast" zu verwenden, um eine Mehrfachnutzung der Netzwerkkarte zu ermöglichen.

---

#### Hinweis

##### Wahl des Clusterausführungsmodus

- Mit den Parametern für den Clusterausführungsmodus wird angegeben, ob eine Multicast-MAC-Adresse (Media Access Control) für Clustervorgänge verwendet werden sollte. Wenn Multicast aktiviert ist, wird die Cluster-MAC-Adresse für den Clusteradapter vom Netzwerklastenausgleich in eine Multicast-Adresse konvertiert. Dadurch wird auch sichergestellt, dass die primäre IP-Adresse des Clusters als Teil des ARP (Address Resolution-Protokoll) in diese Multicast-Adresse aufgelöst wird. Vom Adapter kann nun dessen ursprüngliche integrierte MAC-Adresse verwendet werden, die im Unicast-Modus deaktiviert wurde.
- Sie müssen zuerst die Unterstützung für Multicast aktivieren, bevor Sie die Unterstützung für das Internet Group Management-Protokoll (IGMP) aktivieren. Die IGMP-Unterstützung kann auch auf dem Netzwerkadapter aktiviert werden.
- Mit dem Kontrollkästchen IGMP-Multicast wird die IGMP-Unterstützung zum Einschränken der Switchüberflutung aktiviert. Diese Einschränkung wird erreicht, indem der Datenverkehr nur auf "Netzwerklastenausgleich-Ports" eingeschränkt wird. Aktivieren Sie die IGMP-Unterstützung, um sicherzustellen, dass der für einen NLB-Cluster vorgesehene Datenverkehr nur durch die für die Clusterhosts bestimmten Ports geleitet wird.

- 
6. Bestätigen Sie mit "OK".

### Weitere Hosts dem Cluster hinzufügen

---

#### Hinweis

Damit Sie dem Cluster weitere Hosts (Server) hinzufügen können, werden Administratorrechte auf dem remoten Host benötigt.

---

Um Administratorrechte auf dem remoten Host zu erhalten, müssen Sie sich im "Netzwerklastenausgleich-Manager" mit Administratorrechten anmelden. Gehen Sie dazu folgendermaßen vor:

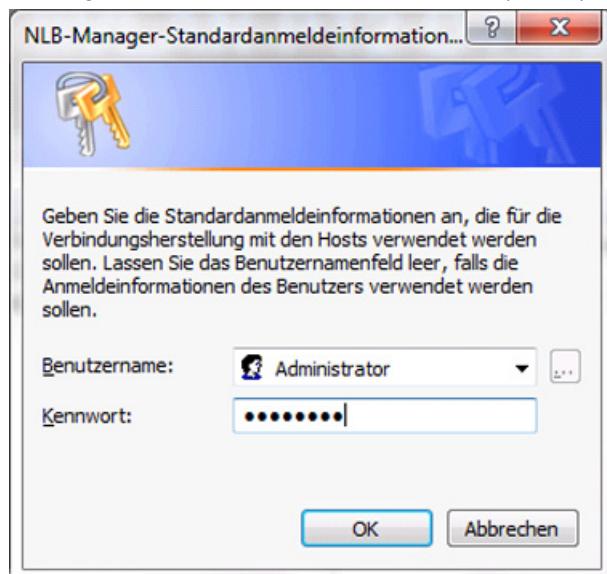
1. Öffnen Sie die im "Netzwerklastenausgleich-Manager" die "Anmeldeinformationen".

Klicken Sie auf "Optionen" > "Anmeldeinformationen".

Reaktion: Das Dialogfenster "NLB-Manager-Standardanmeldeinformationen" wird geöffnet.

2. Geben Sie einen Benutzernamen und das dazugehörige Kennwort ein.

Verwenden Sie bei einem Arbeitsgruppen-Betrieb der Server den von Windows voreingestellten Benutzer "Administrator" (builtin) mit dem dazugehörigen Kennwort.



3. Bestätigen Sie mit "OK".

#### Hinweis

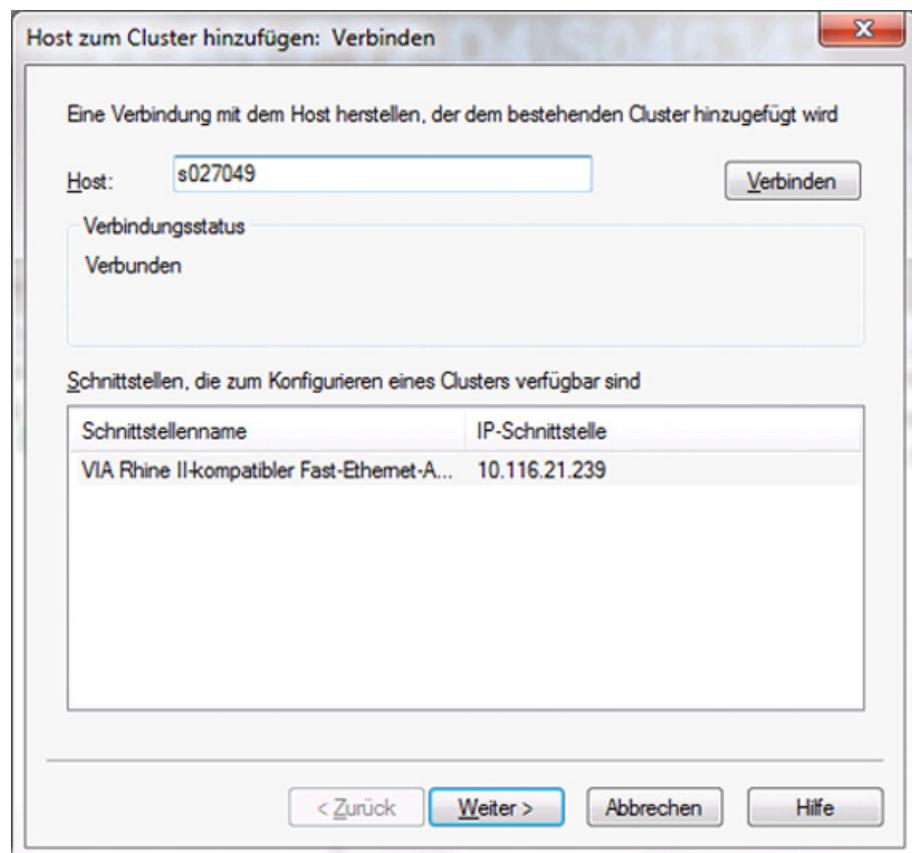
Eine höhere Sicherheit bietet der Domänenbetrieb der Server, da die Benutzerrechte durch die Windows-Domäne verwaltet werden. Dieser Betrieb wird empfohlen.

Um dem Cluster einen weiteren remoten Host hinzuzufügen, gehen Sie folgendermaßen vor:

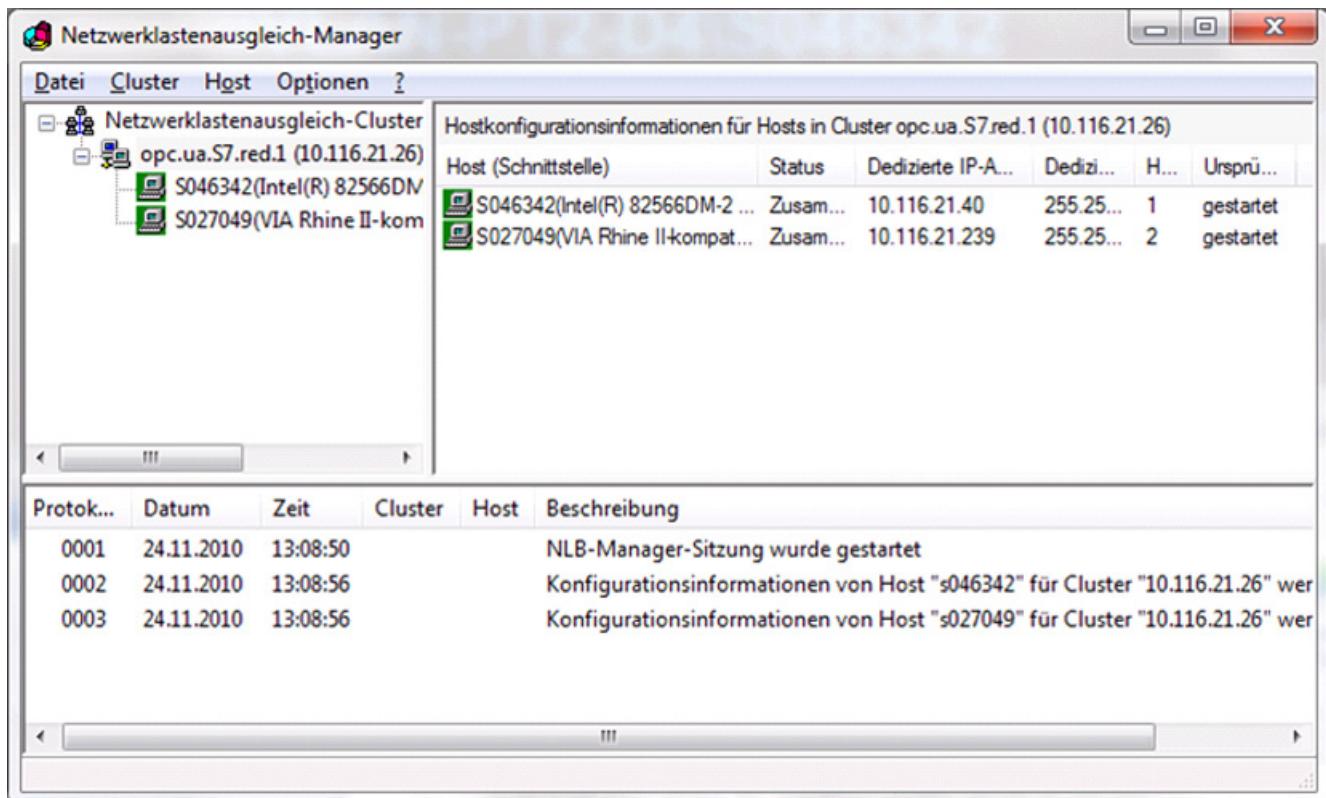
1. Klicken Sie auf den Menübefehl "Cluster" > "Host hinzufügen".

Reaktion: Das Dialogfenster "Host zum Cluster hinzufügen: Verbinden" wird geöffnet.

2. Geben Sie in dem Eingabefeld "Host:" den Namen des Hosts an, den Sie dem Cluster hinzufügen wollen und klicken Sie auf "Verbinden".



Nach der erfolgreichen Host-Verbindung ist ein neuer Cluster erstellt.



### Hinweis

Ein aktiver und passend konfigurierter "Netzwerklastenausgleich" (NLB) ist Voraussetzung für die transparente OPC-UA-Redundanz.

Ob der NLB aktiv ist, können Sie mit einem PING-Aufruf des Betriebssystems testen.

## 4.5.2 Projektierung der Redundanzgruppe

Die initiale Projektierung der Redundanzgruppe wird mit STEP 7 vorgenommen. Mit STEP 7 kann u. A. der Abgleich von Sessions und Subscriptions projektiert werden.

Mit der Projektierungsoberfläche können Sie:

- Die virtuelle Netzwerkadresse der Redundanzgruppe konfigurieren (z. B. Redundante IP-Adresse und Redundanter Port).
- Den Port für die Server-zu-Server-Kommunikation zwischen den redundanten OPC-UA-Servern projektiern.
- Die IP-Adressen der Mitglieder der anderen Redundanz-Gruppen angeben.
- Den lokalen Server in der Liste selektieren.

Das Zertifikat des redundanten Sets kann in der Projektierung bei Bedarf erstellt und angezeigt werden.

---

### Hinweis

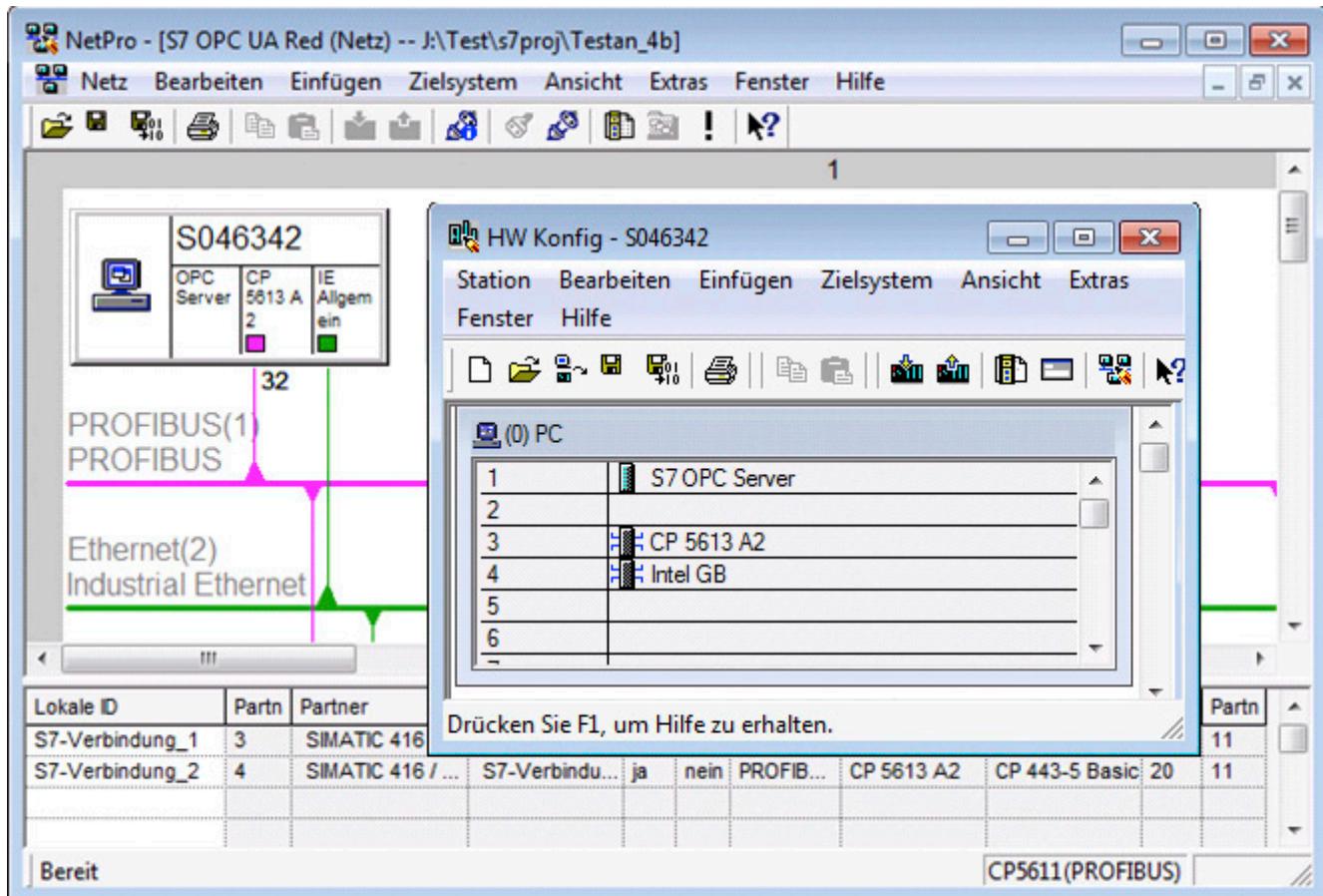
#### Projektierungsoberfläche

- Die Liste und die Reihenfolge müssen auf allen Servern der Redundanzgruppe identisch sein.
  - Es können nur IPv4 Adressen verwendet werden.
  - Die Konfiguration von "Gateway" oder "Subnetmask" ist nicht notwendig.
  - MAC-Adressen müssen nicht projektiert werden.
- 

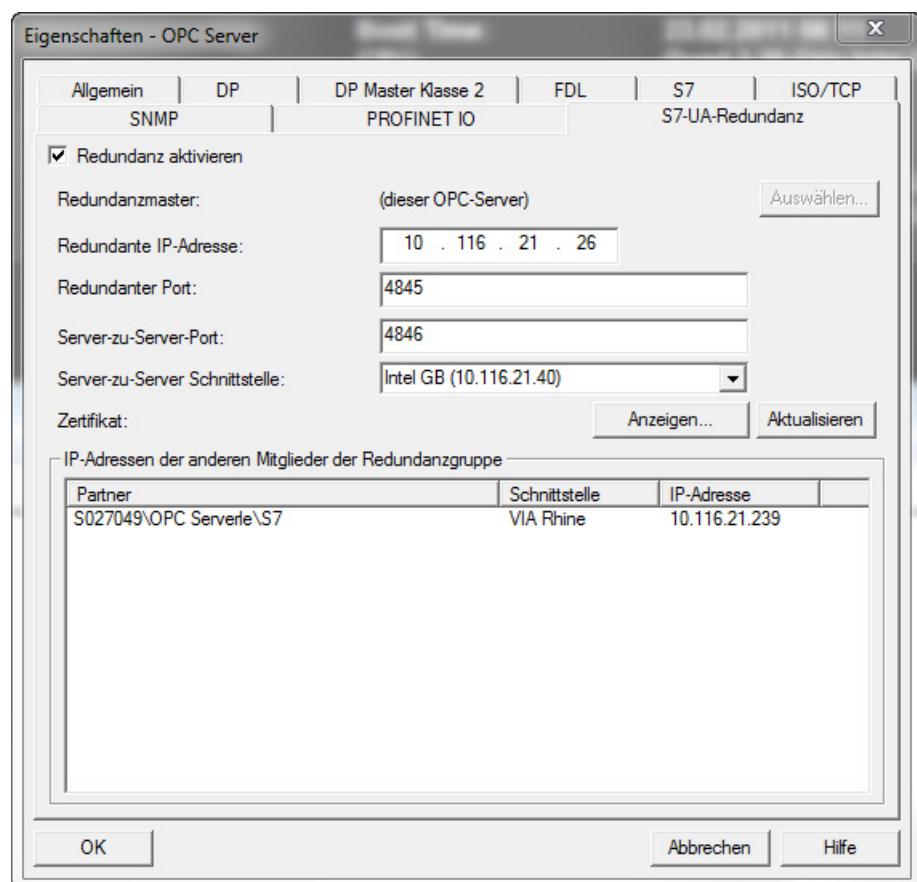
Der "Netzwerkklastenausgleich" (NLB) kann nicht über STEP 7/NCM projektiert werden. Auch Portsperillisten für den NLB werden hier nicht projektiert. Die Projektierung des NLB ist im Betriebssystem auf den redundanten Servern integriert und muss dort vorgenommen werden.

#### Projektierungsbeispiel

Im nachfolgenden Beispiel ist ein STEP 7-Projekt mit einem Anlagenbus z. B. "PROFIBUS(1)" und einem "Industrial Ethernet(2)" für die OPC-Client/Server-Kommunikation dargestellt. Die redundanten Server verfügen jeweils über zwei CPs, im Beispiel "CP 5613 A2" und "IE Allgemein". Als Zielstation kann eine S7-Station gewählt werden. Diese sollte über den Anlagenbus und entsprechende CPs angeschlossen sein.



Die folgende Abbildung zeigt ein Beispiel mit den notwendigen Konfigurationsoptionen. Diese Konfiguration ist in STEP 7 integriert. Die Parameter für die S7-UA-Redundanz können über das Eigenschaften-Fenster des OPC-Servers eingestellt werden (ab Version 8.1).



## Konsistenz der OPC-UA-Server sicherstellen

In der Konsistenzprüfung, z. B. beim Speichern und Übersetzen des Projekts, werden folgende Sachverhalte geprüft:

- Die Server-zu-Server Adressen innerhalb einer Redundanzgruppe müssen eindeutig sein. Ebenso darf innerhalb eines Projektes eine gemeinsame IP-Adresse einer Redundanzgruppe auch nur einmal verwendet werden.
- Jedes Mitglied einer Redundanzgruppe muss über dieselben S7-Verbindungen mit denselben Verbindungsnamen und denselben Kommunikationspartnern verfügen.
- Der Hardware-Ausbau der PC-Stationen in den Redundanzgruppen muss identisch sein.

- Die Parameter der S7-Verbindungen müssen identisch sein. Hierzu gehören: Verbindungsaufbau, Meldungen, Optimierungen, PDU-Größe, Anzahl paralleler Aufträge, Überwachungszeiten.
- Die S7-Einstellungen der OPC-UA-Server müssen identisch sein. Hierzu gehören: Zykluszeit, Zugriffsrechte und S7-Symbolik.

---

#### **Hinweis**

Alle Server können jedoch so konfiguriert werden, dass Sie im gleichen Netzwerk gegenseitig auch über eine individuelle IP-Adresse erreichbar sind.

---

#### **Hinweis**

Auf S7-Protokollebene kann zusätzlich das Produkt S7-Redconnect eingesetzt werden. Für OPC-Redundanz ist es jedoch keine Voraussetzung.

---

#### **Hinweis**

Es wird dringend empfohlen aktiv-permanent aufgebaute S7-Verbindungen auf den OPC-UA-Servern zu projektieren, um:

- geringere Umschaltzeiten bei einem Serverausfall zu erzielen.
  - die Verbindungszustände des passiven redundanten Server überprüfen zu können (S7-Verbindungsdiagnose).
- 

Um den gleichen Namensraum in beiden Server zu erhalten, müssen die beiden PC-Stationen identisch konfiguriert und projektiert werden. Um dies zu erreichen, empfehlen wir Ihnen zuerst eine SIMATIC PC-Station zu erstellen und diese in NetPro zu projektieren (z. B. gewünschte S7-Verbindungen erstellen).

Die erstellte SIMATIC PC-Station, im Bild "S046342", kann nun im SIMATIC Manager dupliziert werden. Mit den Menübefehlen "Kopieren" und "Einfügen" können Sie die PC-Station inklusive namengleicher S7-Verbindungen kopieren. Passen Sie nun noch den Stationsnamen, im Bild "S027049", die lokalen Adressen und das Ziel der S7-Verbindungen an. Falls weitere S7-Verbindungen hinzugefügt werden, müssen diese auf beiden Stationen identisch projektiert werden.

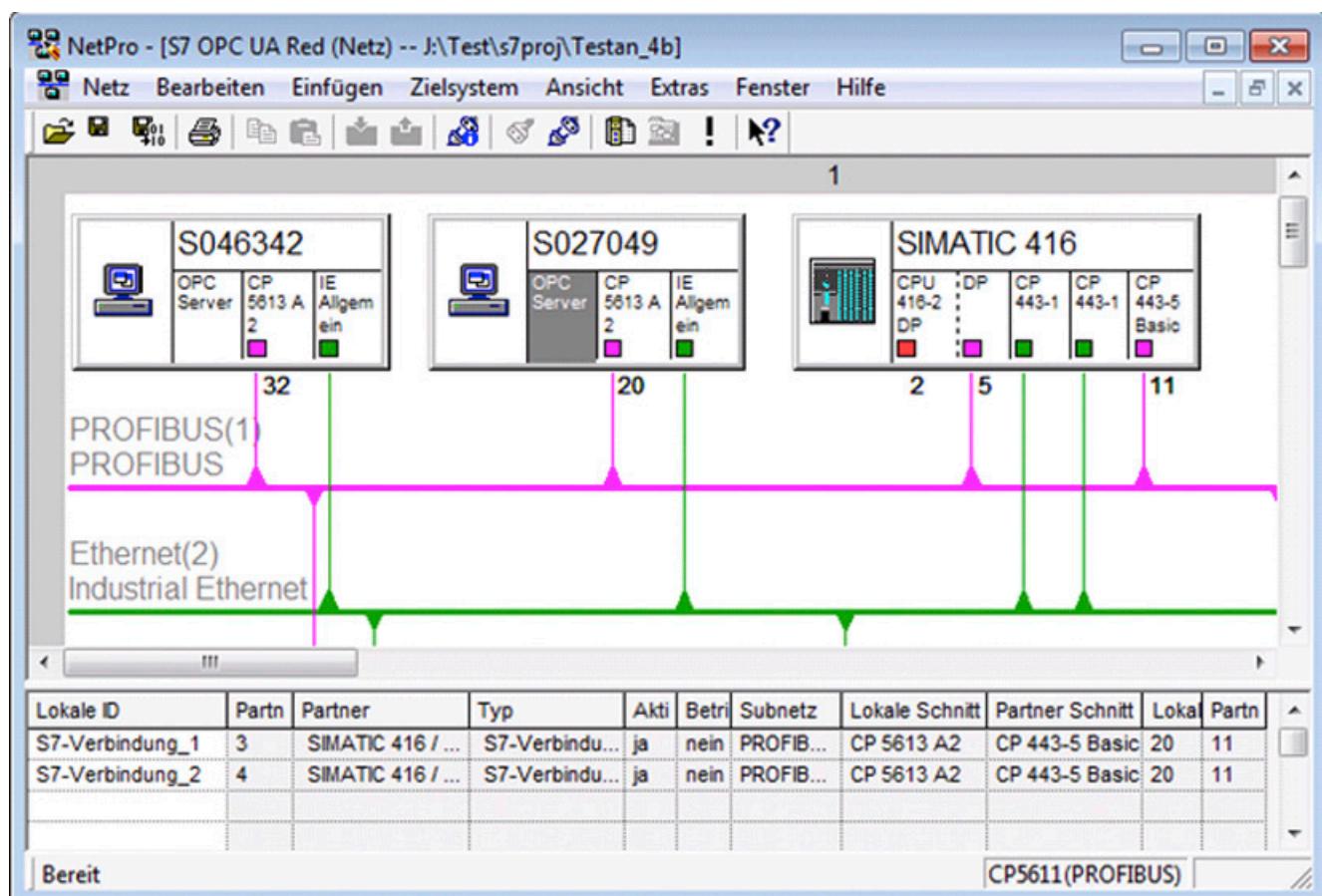


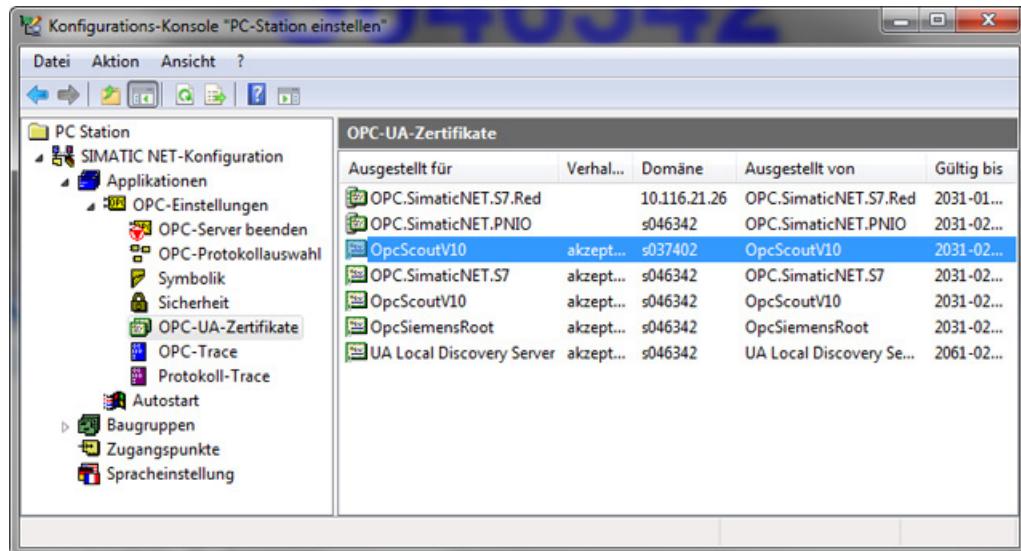
Bild 4-17 In NetPro identisch projektierte PC-Stationen mit S7-Verbindungen

Weiterhin ordnen Sie in der Redundanz-Konfiguration, diese Station der Redundanzgruppe der ersten Station zu. So entsteht am einfachsten ein redundantes Server-Set, mit gleichen S7-Verbindungen zur selben SIMATIC S7-Station.

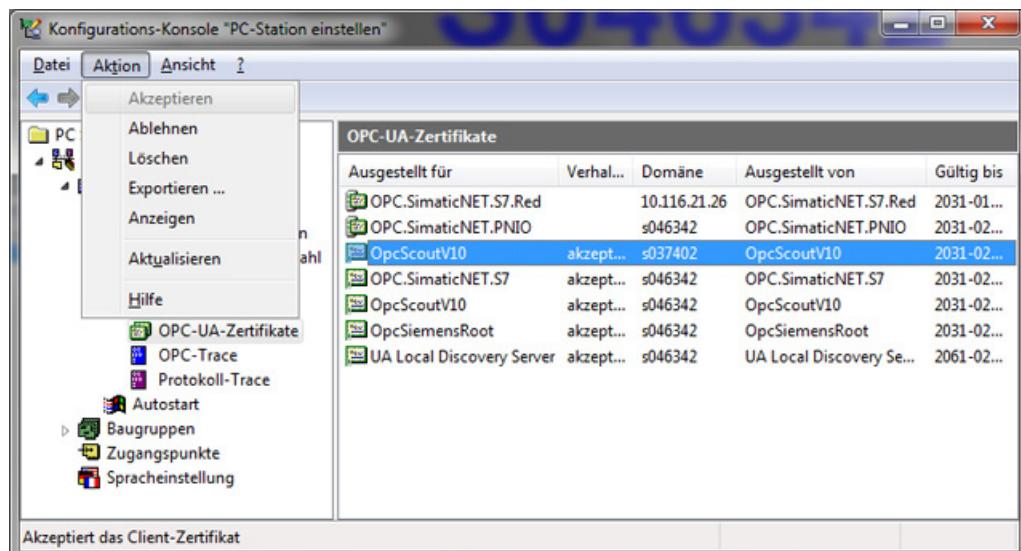
Achten Sie bei der Projektierung darauf, dass für die SIMATIC S7-Station die doppelte Anzahl von S7-Verbindungen erforderlich sind.

## Projektierungen herunterladen und UA-Zertifikate aktualisieren

Nachdem die Projektierung fertig gestellt ist und die Konsistenzprüfung erfolgreich war, müssen Sie die Projektierungen für alle Teilnehmer herunterladen. Auf die beiden SIMATIC PC-Stationen wird dabei auch das identische OPC-UA-Zertifikat geladen und aktiviert. Dies können Sie mit dem Konfigurationsprogramm "Kommunikations-Einstellungen" ("Applikationen" > "OPC-Einstellungen" > "OPC UA-Zertifikate") kontrollieren.



In der Zertifikatverwaltung müssen Sie auch die Zertifikate der OPC-UA-Clients akzeptieren. Verbinden Sie dazu den OPC-UA-Client mit dem redundanten Server, mithilfe der redundanten IP-Adresse (z. B. "10.116.21.26", vergleichen Sie dazu die Abbildung im Abschnitt "Netzwerklastenausgleich konfigurieren").



Bei aktivem NLB wird einer der beiden vorhandenen Server verbunden, akzeptieren Sie dort das Client-Zertifikat bei sicheren Verbindungen. Überprüfen Sie den projektierten OPC-UA-Namensraum und die Verwendbarkeit ihrer Items. Um auch auf dem zweiten Server das Client-Zertifikat zu akzeptieren, ziehen Sie das Netzwerkkabel zum ersten Server. Verbinden

Sie anschließend nochmals mit dem OPC-UA-Client, mithilfe der redundanten IP-Adresse. Nun muss der NLB auf den zweiten Server verbinden. Geben Sie dazu den Rechnername bzw. die lokale IP-Adresse des PC an. Der zweite redundante Server sollte sich genauso wie der erste redundante Server verhalten und gleiche Werte liefern.

Wenn die Verbindung nicht aufgebaut wird, überprüfen Sie ihren Anlagenaufbau und ihre Projektierung.

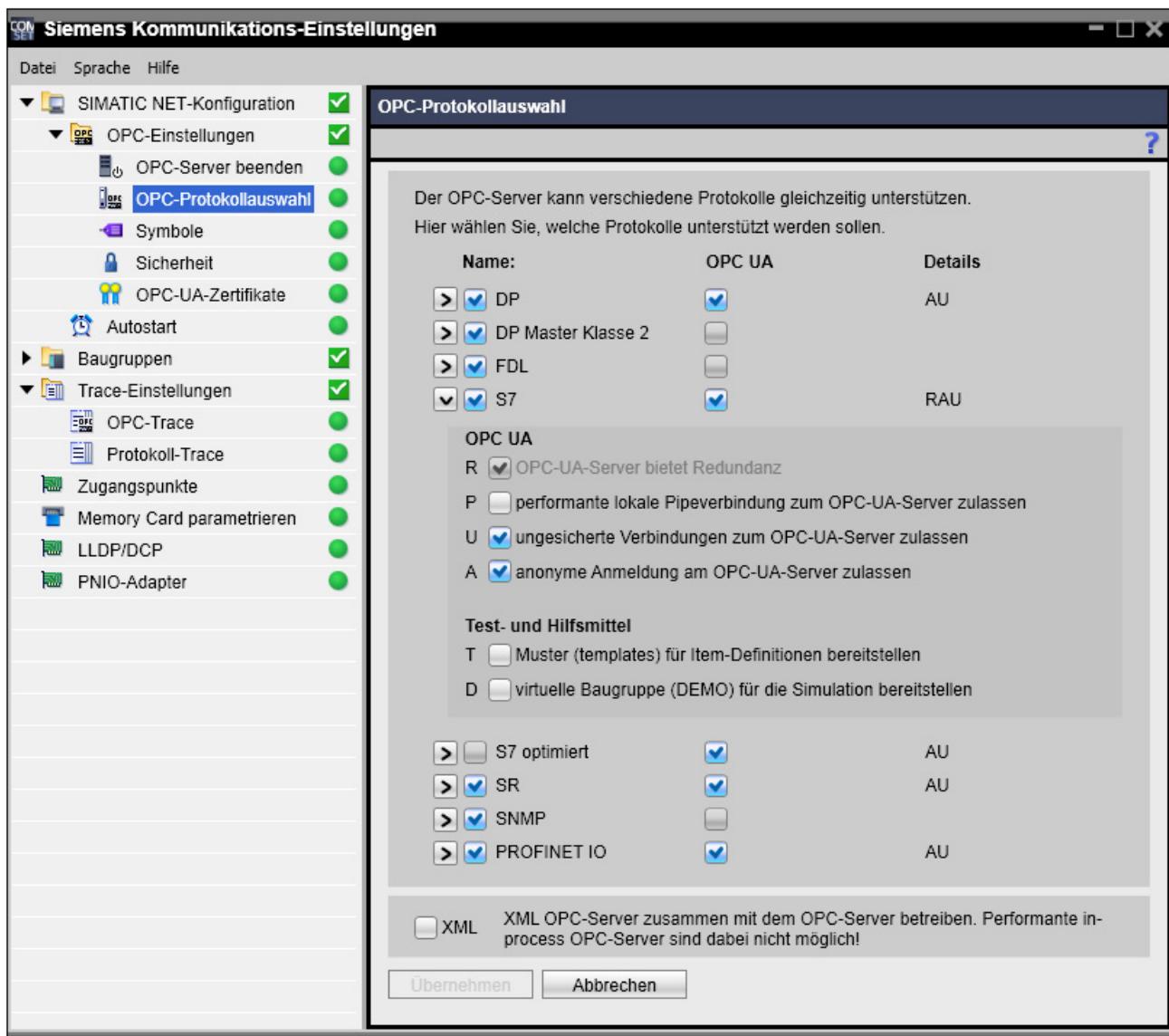


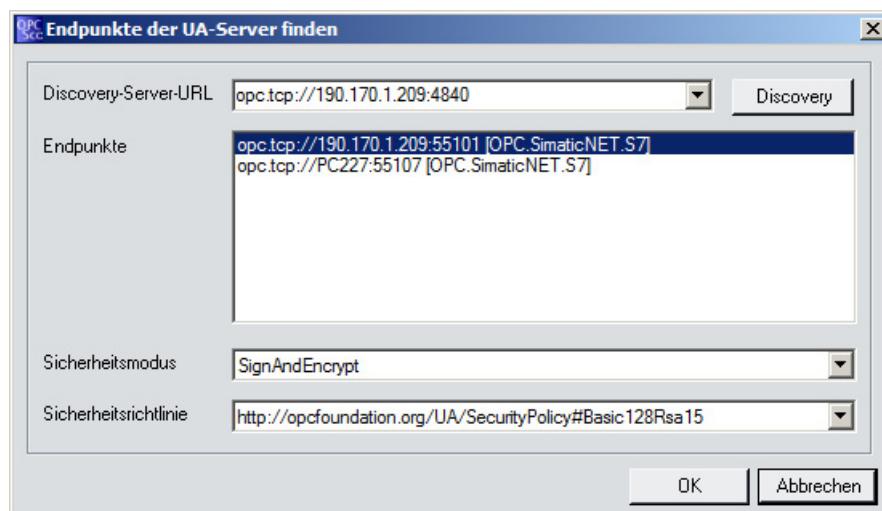
Bild 4-18 Ansicht der Protokolldetails bei funktionierender OPC-UA-Server-Redundanz

## Redundanzumschaltung testen

Die Redundanzumschaltung kann getestet werden, indem das Netzwerkkabel des aktiven Server gezogen wird. Nach erkannter Netzwerkunterbrechung im Client muss eine Wiederverbindung auf den vorhandenen anderen Server erfolgen. Das Umschaltverhalten kann durch die Diagnose-Items beobachtet werden. Nähere Informationen zu diesem Thema finden Sie im Kapitel "Auslesen des Zustands der Redundanzgruppe vom OPC-UA-Client (Seite 555)".

### 4.5.3 Auffinden der Server-Endpunkte mit Hilfe des Discovery-Servers

Mithilfe der OPC-UA-Discovery-Dienste können Endpunkte von redundanten S7-OPC-UA-Servern gefunden werden. Durch Eingeben der gemeinsamen Cluster-IP-Adresse und des Ports ("4840") kann über das binäre OPC-UA-TCP-Protokoll der Endpunkt gefunden werden.



Da die TCP-Verbindung über NLB transparent und in sich lastausgleichend ist, kann nicht vorhergesagt werden, auf welchen der redundanten Server verbunden wird. In dem Auswahlfeld werden alle am Server vorhandenen Endpunkte mit den jeweiligen Sicherheitsmodi angezeigt.

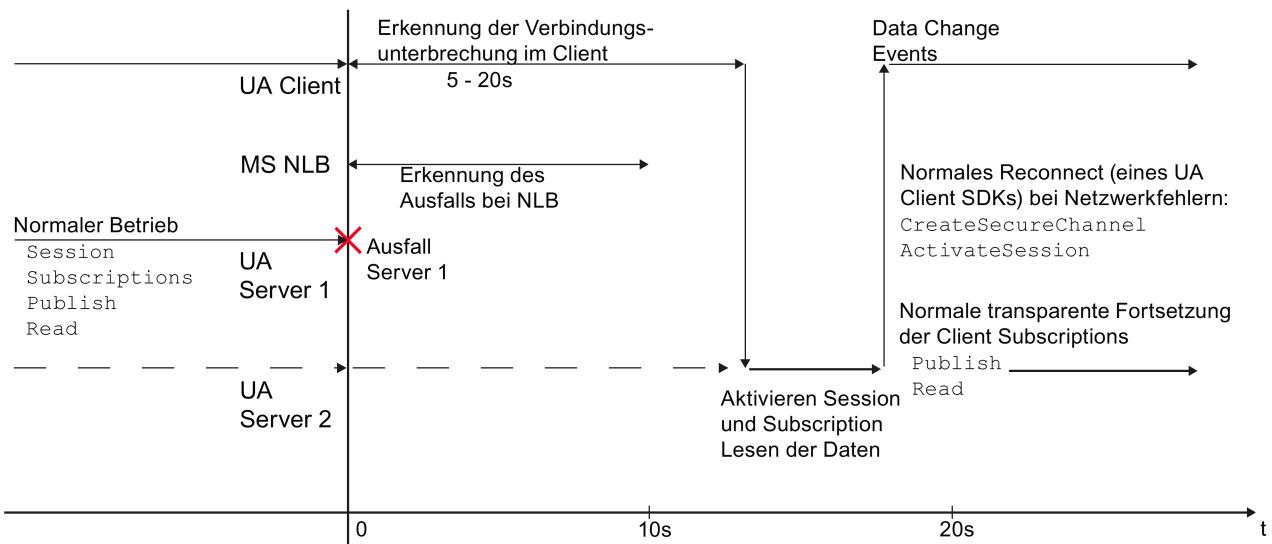
Für eine redundante, transparente OPC-UA-Verbindung wählen Sie den Endpunkt mit der projektierten gemeinsamen IP-Adresse und dem projektierten Server-Port. Im Beispiel "opc.tcp://190.170.1.209:55101".

Für eine gezielte nicht redundante OPC-UA-Verbindung kann auch der Endpunkt des spezifischen S7-OPC-UA-Server gewählt werden. Im Beispiel "opc.tcp://PC227:55107". Diese Verbindung kann genutzt werden, um Zertifikate mit spezifischen Servern auszutauschen oder die Qualität der Daten einzelner Server zu überprüfen. Beachten Sie, ein Abgleich der dabei benutzten OPC-UA-Sessions und -Subscriptions zu weiteren redundanten Servern erfolgt jedoch nicht für diesen Endpunkt.

#### 4.5.4 Der OPC-UA-Reconnect-Vorgang

Wenn ein redundanter OPC-UA-Server ausfällt, muss der Ausfall vom Windows Feature "Netzwerklastenausgleich" (NLB) des Server-Betriebssystems und vom OPC-UA-Client erkannt werden. Eine Unterbrechung der Verbindung wird anhand von auftretenden Netzwerkfehlern oder dem Eintreten eines Timeouts erkannt.

Der "Netzwerklastenausgleich" (NLB) benötigt etwa 10 Sekunden, um einen Ausfall des OPC-UA-Server 1 zu erkennen. Bei einem Ausfall leitet der NLB den Client, bei neuen Aufrufen, auf den verfügbaren, redundanten OPC-UA-Server 2 um.



Die Erkennungszeit beim OPC-UA-Client hängt von der Art der Verbindungsüberwachung und die eingestellten Überwachungszyklen ab, da sich diese direkt auf die Erkennungszeit auswirken können. OPC-UA-Clients können Verbindungen auf zwei Arten überwachen:

- Zyklisches Lesen des Server-Status mit einem kurzen Timeout.

Die Erkennungszeit hängt von folgenden Komponenten ab:

- Der Zykluszeit für den Leseauftrag (z. B. 5 Sekunden).
- Dem Timeout für den Leseauftrag (z. B. 2 Sekunden).
- Wie oft der Leseauftrag im Fehlerfall wiederholt wird, bis die Verbindung als gestört gekennzeichnet wird (z. B. 1x).

- Überwachung des KeepAlive-Wertes der Subscription.

Die Erkennungszeit hängt von folgenden Komponenten ab:

- Der KeepAlive Zeit (z. B. 5 Sekunden).
- Der zusätzlichen Wartezeit bis zur Fehlermeldung (z. B. 5 Sekunden).

Nachdem der Verbindungsfehler erkannt wurde (Netzwerkfehler, Timeout), startet der normale OPC-UA-Reconnect-Vorgang des Client. Diese Reconnect-Funktionalität ist in der Regel in einem Client UA SDK integriert und für den OPC-UA-Client transparent. Beim Reconnect-Vorgang wird:

- Ein neuer "SecureChannel" mit "CreateSecureChannel" erzeugt.

Ein TCP/IP Connect auf die redundante IP-Adresse des NLB erfolgt. Der NLB teilt die TCP-Verbindungen gleichmäßig auf. Da der OPC-UA-Server 1 ausgefallen und nicht erreichbar ist, wird die TCP-Verbindung zum OPC-UA-Server 2 erstellt.

- Die bereits vorhandene Session durch ActivateSession mit Zuweisung des SecureChannels aktiviert.

Der OPC-UA-Client erkennt nicht, dass er mit einem anderen Server verbunden ist, da die Session- und Subscription-IDs weiterhin gültig sind. Die Sessions und vorhandenen Subscriptions werden damit wieder aktiviert.

Der OPC-UA-Client kann nun transparent seine normale Funktionalität fortsetzen. Zum Beispiel beobachten durch wiedervorhandene Subscriptions oder auch einfaches Lesen und Schreiben.

Der OPC-UA-Client schickt ein PublishRequests an den Server, um neue Daten und Events zu erhalten. Dem OPC-UA-Server 2 ist die Session bekannt und alle zugehörigen Subscriptions und Monitored Items werden aktiviert. Die Daten werden über das SIMATIC NET-Kommunikationsprotokoll gelesen. Außerdem wird ein RefreshRequiredEvent an den Client ausgelöst, wenn mehrere Event Monitored Items angelegt sind. Sobald Daten bzw. Events vorhanden sind, werden diese über PublishResponse an den Client geschickt.

## **4.5.5 Redundanz Unterstützung durch OPC-UA-Clients**

### **Einleitung**

Die Eigenschaften, die ein OPC-UA-Client für die "optimale Redundanzumschaltung" benötigt, werden hinsichtlich der OPC-UA-Norm als konformes Verhalten bewertet.

Die wesentlichen Merkmale sind:

- Erkennen eines Verbindungsabbruch bzw. Verbindungsüberwachung
- Reconnect-Funktionen
- Refresh Required Event

### **Erkennen des Verbindungsabbruchs**

Um einen Verbindungsabbruch zu erkennen, muss der OPC-UA-Client über eine Verbindungsüberwachung verfügen.

Die Verbindung kann z. B. durch Folgendes überwacht werden:

- Statusüberwachung der Methode "Read"
- Überwachung des KeepAlive-Wertes der Subscription

Nachdem ein Verbindungsabbruch erkannt wurde, soll der OPC-UA-Client einer Reconnect-Logik folgen.

### Reconnect-Funktionen

Mit dem Reconnect-Vorgang nutzt der OPC-UA-Client die Möglichkeit von OPC UA noch bestehende Sessions und Subscriptions weiter zu verwenden.

Dafür muss die folgende Sequenz durchlaufen werden. Die Sequenz weicht vom initialisierenden Connect-Vorgang ab.

1. CreateSecureChannel

2. ActivateSession

Bei Fehler von AcitvateSession

- CreateSession
- ActivateSession
- Subscription neu anlegen

3. Auf Datenänderung / Events warten oder prüfen, ob die Subscription noch existiert.

Bei Fehler von Subscription

- Subscription neu anlegen

Wenn der OPC-UA-Client den Reconnect-Vorgang wie den Connect-Vorgang ausführt, besteht lediglich der Vorteil, dass der Client sich wieder auf den gleichen Server-Endpunkt verbinden kann.

### RefreshRequiredEvent

Der OPC-UA-Client muss auf das RefreshRequiredEvent des Servers reagieren. Dazu führt der OPC-UA-Client entsprechend ein Refresh aus. Dieser Ablauf ist in der Spezifikation "OPC UA Part 9 - Alarms and Condition" beschrieben.

---

#### Hinweis

Prüfen Sie den verwendeten OPC-UA-Client auf konforme Verhaltensweise. Oft sind Clients basierend auf SDKs implementiert, die solche Mechanismen bereits beinhalten.

---

### 4.5.6 Wann tritt eine Serverumschaltung ein?

Bei einem Server-Ausfall findet eine Umschaltung statt, wenn der OPC-UA-Client einen Ausfall von Channel, Session oder Subscription erkennt. Z. B. bei:

- Ausfall des Host-PC
- Absturz des OPC-UA-Servers
- Absturz des Betriebssystems

Eine Umschaltung findet nicht statt, wenn:

- S7-Verbindungsunterbrechungen oder andere Fehlerzustände auftreten, die verhindern, dass der OPC-UA-Server S7-Daten von der Steuerung erhält.
- Fehlerzustände im OPC-UA-Server auftreten, die keinen Ausfall von Channel, Session oder Subscription nach sich ziehen bzw. dieser Ausfall vom OPC-UA-Client nicht erkannt werden.

#### **4.5.7 Details zur Serverumschaltung**

##### **Mögliche Fehlermeldungen an den OPC-UA-Client**

Sind während des Ausfalls von Server 1 Schreib- oder Lese-Aufträge aktiv, werden für diese Aufträge vom Stack des OPC-UA-Client Fehler gemeldet. Diese Aufträge müssen dann von der Client-Applikation wiederholt werden.

##### **Mögliche Fehlermeldungen an den Benutzer des OPC-UA-Client**

OPC-UA-Clients setzen im Regelfall die Qualität der Variablen intern auf "BAD" wenn die Verbindung zum Server abbricht. Nach dem Reconnect wird die Qualität durch die neuen Daten wieder zurückgesetzt.

Bei einem Verbindungsabbruch kann dem Anwender die Qualität der Daten also kurz als "BAD" angezeigt werden. Diese Zeitspanne wird nur wenige Sekunden dauern oder, je nach Client, gar nicht erst eintreten. Ursache dafür ist, dass die Qualität der Daten erst nach der Erkennung des Verbindungsabbruchs auf "BAD" gesetzt wird, der Reconnect jedoch sofort automatisch gestartet wird.

##### **Kein Abgleich von Daten und Events**

Daten und Events, die zwischen dem Client und der über das S7-Kommunikationsprotokoll angeschlossenen SPS ausgetauscht werden, werden zwischen den redundanten OPC-UA-Servern nicht abgeglichen.

Jeder Server im redundanten Set verarbeitet diese Information nur über seine eigene S7-Verbindung. Nach der Redundanzumschaltung erhält der neue Server diese Informationen erneut von der SPS.

Wenn eine Subscription nach dem Umschalten aktiv wird, dann werden für:

- den Datentyp "Data Change Monitored Item" die aktuellen Werte über das SIMATIC NET-Kommunikationsprotokoll ermittelt und geliefert, sobald diese gelesen wurden.
- den Datentyp "Event Monitored Item" ein RefreshRequiredEvent an den Client geschickt, um einen Refresh durch den Client auszulösen. Dadurch wird dem Client für die aktiven Alarme der aktuelle Status, mit einer im neuen Server bekannten "EventId", zugeschickt. Ein UA-Alarm-Client kann die Events über die identische "ConditionId" den bekannten Conditions zuordnen. Er benötigt jedoch eine im Server bekannte "EventId", um Aktionen (z. B. Acknowledge von Alarmen) durchzuführen. Bei einer kurzzeitigen Unterbrechung und dem Wiederverbinden auf denselben Server wird kein "RefreshRequiredEvent" zum Client gesendet.

## Einschränkungen bei der Serverumschaltung

Folgende lokale, auf dem Server vorhandene, Daten und Zustände werden nicht abgeglichen:

- @Localserver Daten, die durch den Client oder über einen remoten Zugriff (S7-Station zur PC-Station, S7-Verbindung) durch PUT und GET-Funktionsbausteine geschrieben und gelesen werden.
- Lokale DEMO-Variablen.
- Lokale Puffer (auch Teilebereiche) von Blockdiensten "BSEND/BRCV", da die Blockdienste Daten gezielt zwischen der SPS und einem der redundanten OPC-Server ausgetauscht werden.
- Lokal spezifisch projektierte Verbindungsparameter (Baudaten, PDU Size, Timeouts), die durch unterschiedliche Projektierung entstanden sind.

Unterschiedliche Projektierung kann durch Folgendes entstehen:

- Auf beiden Servern wurden unterschiedliche Projekte bzw. Arbeitsstände geladen.
- Während der Laufzeit wurden zwischen den Kommunikationspartnern unterschiedliche Parameter ausgehandelt.
- Lokal spezifisch projektierte Symbolik (S7-Symbolnamen mit S7-Verbindungen, Deadbands,...), die durch unterschiedliche Konfiguration entstanden sind.
- Lokale Alarmzustände, -historie und -quittierungen. Dies gilt z. B. für lokal generierte Statepath-Alarne.
- Lokal akzeptierte oder abgelehnte OPC-UA-Client-Zertifikate des lokalen OPC-UA-Servers im redundanten Set.

Verwenden Sie die Zertifikatverwaltung in "Kommunikations-Einstellungen", um auf allen beteiligten OPC-UA-Servern das OPC-UA-Client-Zertifikat zu akzeptieren. Dadurch stellen Sie im Umschaltfall auch eine Akzeptanz beim Server sicher. Ab "SIMATIC NET PC Software V8.2" werden OPC-UA Client-Zertifikate auf redundanten Servern automatisch abgeglichen.

- Lokaler Ausfall von Teilkomponenten und lokale Fehlerzustände des SIMATIC NET-Kommunikationsprotokolls.

Diese Daten müssen vom Client verwaltet und bei erkannter Verbindungsunterbrechung wieder neu initialisiert bzw. angefordert werden. Dies funktioniert nur, wenn bei der Projektierung der Redundanzgruppe die Server-Namensräume konsistent und identisch projektiert wurden.

### Besonderheiten bei S7-Meldungen

- Diagnosemeldungen (Simple Events) werden während einer Verbindungsunterbrechung zwischen Server und Client nicht weitergereicht. Das gilt auch bei der Redundanzumschaltung. Die entsprechenden Events gehen verloren.
- S7-Meldungen. Wenn sich in der SPS eine S7-Meldung ändert, wird die Änderung separat an beide OPC-UA-Server gesendet. Die S7-Meldung kann an den OPC-UA-Server mit zeitlichem Versatz eintreffen. Der zeitliche Versatz kann z. B. durch viele Meldungen oder durch ein großes Mengengerüst auftreten und damit in der Größenordnung einer Redundanzumschaltung liegen. Wenn dies der Fall ist, kann Folgendes auftreten:
  - Am OPC-UA-Client wurde vor dem Serverausfall der Alarmzustand von Inaktiv nach Aktiv geändert. Nach der Redundanzumschaltung wird vom zweiten OPC-UA-Server ein Refresh ausgelöst. Der Alarmzustand wird zunächst mit dem Alarmzustand "Inaktiv" und dann mit dem Alarmzustand "Aktiv" gemeldet. Das gilt auch für den umgekehrten Fall.
  - Am OPC-UA-Client wurde vor dem Serverausfall eine Meldung quittiert. Nach der Redundanzumschaltung wird vom zweiten OPC-UA-Server ein Refresh ausgelöst. Die Meldung wird zunächst mit dem Status "Unquittiert" und dann mit dem Zustand "Quittiert" gemeldet.

### Besonderheiten beim Lesen/Beobachten von BSEND/BRECEIVE-Variablen

Wenn beim OPC-UA-Server bereits ältere Werte für ein Monitored BSEND/BRECEIVE Item (Variable) im Cache vorliegen, dann wird bei der Aktivierung einer Subscription zuerst der Wert aus dem Cache inklusive Qualität und Zeitstempel an den Client geliefert, bevor über das SIMATIC NET-Kommunikationsprotokoll ein aktueller Wert ermittelt und mit neuer Qualität und Zeitstempel geliefert wird. Der Wert aus dem Cache kann älter sein, als ein Wert, der vor der Redundanzumschaltung von einem anderen Server des redundanten Sets an den Client geliefert wurde. Auch wird in der Regel der Status "Good" gemeldet. Der Client muss daher beim Lesen/Beobachten von Werten die dazugehörigen Zeitstempel beachten.

## 4.5.8

### Verhalten des SIMATIC NET S7-OPC-UA-A&C bei Redundanzumschaltung

Vor der Redundanzumschaltung sendet der S7-OPC-UA-Server A an den Client Events, sofern bei einer Condition eine Änderung auftritt. Während der Umschaltzeit werden an den Client keine Events gesendet. Nach dem Umschalten sendet der S7-OPC-UA-Server B Events an den Client.

Wenn nach dem Umschalten der S7-OPC-UA-Server B aktiv ist, dann sendet der S7-OPC-UA-Server B für jedes Event-Monitored-Item mittels des Events RefreshRequired (Nodeld i=2789) die Aufforderung an den Client, einen Refresh durchzuführen. Durch die Durchführung dieser Refresh-Anforderung, wird sichergestellt, dass der Client über die aktuellen Zustände der Conditions informiert wird.

Die S7-Alarmbausteine werden auf OPC-UA-Alarmconditions, kurz Alarne, mit entsprechenden Alarmzuständen abgebildet.

Beide S7-OPC-UA-Server (Server A und Server B) sind mit den S7-PLC verbunden, so dass beide dieselben Meldungen empfangen. Unter Meldung wird die Signalisierung der

Zustandsänderung eines S7 Alarmbausteins verstanden. Obwohl beide S7-OPC-UA-Server die Meldungen empfangen, sind diese zwei Rechner nicht als synchronisiert anzusehen. Die S7-PLC meldet mit nicht vorhersehbarer Reihenfolge und Laufzeit nacheinander an alle angeschlossenen S7-OPC-UA-Server und an weitere Bedienen & Beobachten-Stationen. Bei einer Menge von 2000 Meldungen kann es passieren, dass dieselbe Meldung von der S7-PLC zunächst an S7-OPC-UA-Server A und an S7-OPC-UA-Server B erst 10 bis 20 Sekunden später gesendet wird.

Ein Änderungereignis (Event) für eine OPC-UA-Alarm hat 4 relevante Zeitstempel: ReceiveTimeClient (RTC-Zeitpunkt, zu dem das Event beim Client empfangen wird), ReceiveTime (RT-Zeitpunkt, zu dem die Meldung beim Server empfangen wird), Time (modifizierter Zeitstempel der Quelle – modifizierter Zeitpunkt der auslösenden Flanke auf dem S7 Alarmbaustein) und S7Time (Zeitstempel des Quelle-Zeitpunktes der auslösenden Flanke auf dem S7-Alarmbaustein). In folgendem wird nur auf ReceiveTime (RT) und S7Time Bezug genommen.

Die ReceiveTime RT ist sowohl wegen den nicht vorhersehbaren Laufzeiten als auch wegen der in der Regel nicht exakt gleich eingestellten Uhrzeiten auf den Rechnern A und B für ein und dieselbe Meldung bei beiden S7-OPC-UA-Servern A und B immer, zumindest leicht, unterschiedlich.

Die S7Time wird von der S7-PLC gesetzt und ist für ein und denselben Flankenwechsel bei beiden S7-OPC-UA-Servern immer identisch.

Durch ein Umschalten einer Event-Subscription von S7-OPC-UA-Server A nach B können folgende 4 Fälle auftreten:

- Eine Meldung erfolgt von der S7-PLC während des Umschaltvorgangs an die beiden S7-OPC-UA-Server und löst währenddessen beim Client keinen Event aus. Der Client wird durch den Refresh nach der Umschaltung von diesem neuen Alarmzustand informiert. RT zeigt den Zeitstempel des S7-OPC-UA-Server B an.
- Eine Meldung erfolgt von der S7-PLC kurz vor Umschaltvorgang an S7-OPC-UA-Server A und löst von diesem mit dessen Zeitstempel RT beim Client einen Event mit dem neuen Alarmzustand aus. S7-OPC-UA-Server B erhält dieselbe Meldung von der S7-PLC zu einem beliebigen anderen Zeitpunkt (möglicherweise auch vor A), jedoch vor Abschluss des Umschaltvorgangs, so dass S7-OPC-UA-Server B beim Client keinen Event auslöst. Der Refresh meldet denselben Alarmzustand mit identischer S7Time und anderen identischen Properties, jedoch mit unterschiedlicher ReceiveTime RT. Protokolliert man die Events (relevante Systemereignisse) für eine spätere Untersuchung, so ist der Alarmzustand zu verwerfen, der bei Rechner A oder Rechner B später empfangen wurde. Setzt man die Zeitsynchronität der zwei Rechner voraus, so ist das der Rechner mit dem jüngeren (größeren) RT.
- Eine Meldung erfolgt von der S7-PLC kurz vor Umschaltvorgang an S7-OPC-UA-Server A und löst von diesem mit dessen Zeitstempel RT beim Client einen Event mit dem neuen Alarmzustand aus. S7-OPC-UA-Server B erhält dieselbe Meldung von der S7-PLC zu einem späteren Zeitpunkt nach dem Umschaltvorgang, jedoch vor Beginn des Refresh. Der Refresh-Vorgang ist atomar, d.h. S7-OPC-UA-Server B kann den zugehörigen Event vor dem Refresh-Vorgang absetzen oder unmittelbar danach. Derselbe Alarmzustand mit identischer S7Time wird in diesem Falle insgesamt dreimal gemeldet. Die beiden Events mit den jüngeren (größeren) RT stammen vom S7-OPC-UA-Server B und sind für eine Protokollierung zu verwerfen.
- Eine Meldung erfolgt von der S7-PLC kurz vor Umschaltvorgang an S7-OPC-UA-Server A und löst noch von diesem mit dessen Zeitstempel RT beim Client einen Event mit dem

neuen Alarmzustand aus, S7-OPC-UA-Server B erhält dieselbe Meldung von der S7-PLC zu einem späteren Zeitpunkt nach Beginn des Refresh. Der Refresh meldet daher einen veralteten Alarmzustand mit veralteter S7Time. Das wird vom Client erkannt und deshalb verworfen. S7-OPC-UA-Server B löst später für den zu S7-OPC-UA-Server A identischen Alarmzustand einen weiteren Event aus, der für die Protokollierung verworfen werden kann.

Fällt bei einem der Rechner kurzzeitig die Meldungen-liefernde S7-Verbindung weg, so wird der S7-OPC-UA-Server diese Verbindung erneut aufbauen. Nachdem erfolgreichem Aufbau der Verbindung werden die S7-PLC über den Stand der S7-Alarmbausteine abgefragt. Die von den S7-PLC gelieferte Information ist unvollständig, d.h. dass sie die Zusatzwerte und S7Time nicht enthalten. S7Time nimmt bei unvollständigen Alarmen den Wert 01.01.1990 00:00:01 an. Da dieses Datum nur bei diesen unvollständigen Alarmen auftreten kann, empfiehlt es sich, wenn mehrere Events für denselben Alarmzustand vorhanden sind, die unvollständigen Alarne aus der Betrachtung auszuschließen.

#### 4.5.9 Generelle Empfehlungen und Hinweise für die redundante S7-OPC-UA-Kommunikation

Die PC-Stationen sind redundant vorhanden, die Partner-S7-Station im Allgemeinen nur einfach. Daraus ergibt sich eine höhere Verfügbarkeit, allerdings auch eine aufwändigere Projektierung. Beachten Sie folgende Empfehlungen und Hinweise:

- Blockorientierte Dienste "BSEND/BRCV" und die S7-Verbindung selbst, beinhalten Adressparameter zur speziellen PC-Station und zum OPC-UA-Server.

Projektieren Sie für die SIMATIC S7-Station für jede S7-Verbindungen einen eigenen Funktionsbausteinaufruf "BSEND/BRCV". Die Verwendung von permanenten S7-Verbindungen wird empfohlen. Das SPS-Programm muss die Funktionsbausteinaufrufe für beide S7-Verbindungen gleichzeitig ausführen. Das SPS-Programm wird dadurch aufwendiger.

In der Regel wird der Funktionsbausteinaufruf zu passiven Redundanzseite negativ quittiert oder muss abgebrochen werden.

Beachten Sie auch, dass die Daten auf den PC-Stationen nur lokal vorhanden sind. Ein Abgleich der gesendeten Daten zwischen den redundanten OPC-UA-Servern auf den PC-Stationen erfolgt nicht bei Redundanzumschaltung.

Auch wenn ein Funktionsbausteinaufruf zu einer Seite erfolgreich war, kann es bedingt durch die Redundanzumschaltung am OPC-UA-Client zu Datenverlusten kommen.

- Für jede S7-Verbindung der SIMATIC S7-Station müssen Sie die S7-Kommunikationsdienste "PUT/GET" programmieren.

Der jeweilige Funktionsbausteinaufruf sieht die Datenbereiche des zugehörigen OPC-UA-Server. Beachten Sie auch, dass die Daten auf den PC-Stationen nur lokal vorhanden sind. Ein Abgleich der gesendeten Daten zwischen den redundanten OPC-UA-Servern auf den PC-Stationen erfolgt nicht.

- Für den OPC-UA-Client ist die Verbindung auf einen OPC-UA-Server des redundanten Sets transparent. Die Verbindung wird vom NLB nach Lastausgleichskriterien auf die redundanten Server verteilt. Wenn ein Server ausfällt, wird beim Reconnect auf den redundanten Server verbunden. Es wird empfohlen diesen Ausfalltest vorab bewusst, durch Entfernen des Netzwerkkabels zu dem aktiven OPC-UA-Server, durchzuführen.

Testen Sie auch die Güte der S7-Kommunikation (Variablen) zu beiden redundanten Servern.

Um bei einem Server-Ausfall eine schnellere Wiederverbindung auf die S7-Variablen des redundanten Server zu erhalten, wird die Projektierung aktiver und permanent aufgebauter S7-Verbindungen empfohlen.

- S7-OPC-UA-Redundanzserver beinhalten keine automatische Funktion zur Synchronisation der Uhrzeit. Damit bei einer Redundanzumschaltung keine ungewollten Uhrzeitsprünge stattfinden, müssen Sie die Uhrzeiten der verschiedenen Redundanzserver manuell synchronisieren. Dazu benötigen Sie ein Konfigurationskonzept.
- Wenn für alle Server im RedundantServerArray der ServerState "Running" angegeben wird, dann ist daraus nicht zwingend zu schließen, dass das gesamte Redundanzset fehlerfrei arbeitet. Es bedeutet nur, dass der verbundene OPC-UA-Server sich im Zustand "Running" befindet und er mit den Partner-Servern kommunizieren kann. Ein anderer OPC-UA-Server des Redundanzsets könnte den ServerState seines Partner-Servers mit "Unknown" anzeigen, weil er mit ihm nicht kommunizieren kann. Der ServiceLevel eines OPC-UA-Servers mit ServerState "Running" beträgt 255, wenn dieser seinen Partner-Server erreichen kann, andernfalls den Wert 127. Der fehlerfreie Zustand wird durch den Wert "255" und "Running" beider Server im RedundantServerArray angegeben.
- Beim Aufzählen der Endpunkte eines redundanten OPC-UA-Servers werden die Endpunkte der eigenen Adresse und die der gemeinsamen Adresse des Redundanzpaars aufgezählt. Je nach Konfiguration des Netzwerks können auch die Endpunkte des Redundanzpartners aufgezählt werden.
- Beim Laden einer Redundanzprojektierung erhält ein bestehender lokaler Endpunkt den Port 14845, wenn sein bisheriger Port mit dem Port des projektierten redundanten Endpunkts identisch ist. Beim Löschen der Redundanzprojektierung, erhalten Endpunkte mit dem Port 14845 wieder den Port, den sie vor der Redundanzprojektierung hatten.
- Beachten Sie, dass die Ergebnisse der Methodenaufrufe für Bausteindienste (z.B. password(), blockread() ) nach Redundanzumschaltung dem aktuellen Server nicht zur Verfügung stehen. Die Methodenaufrufe müssen für den aktuellen Server erneut konfiguriert und aufgerufen werden.
- Besonderheiten beim Lesen/Beobachten von Statusvariablen des Servers:  
Statusvariablen sind serverspezifisch und können daher bei jedem Server für den gleichen Zeitpunkt unterschiedliche Werte haben. Ein typisches Beispiel hierfür ist die „CurrentServerId“, welche per Definition für jeden Server unterschiedlich ist. Ihr Zeitstempel ändert sich auch während der Laufzeit des Servers nicht.

#### 4.5.10 Auslesen des Zustands der Redundanzgruppe vom OPC-UA-Client

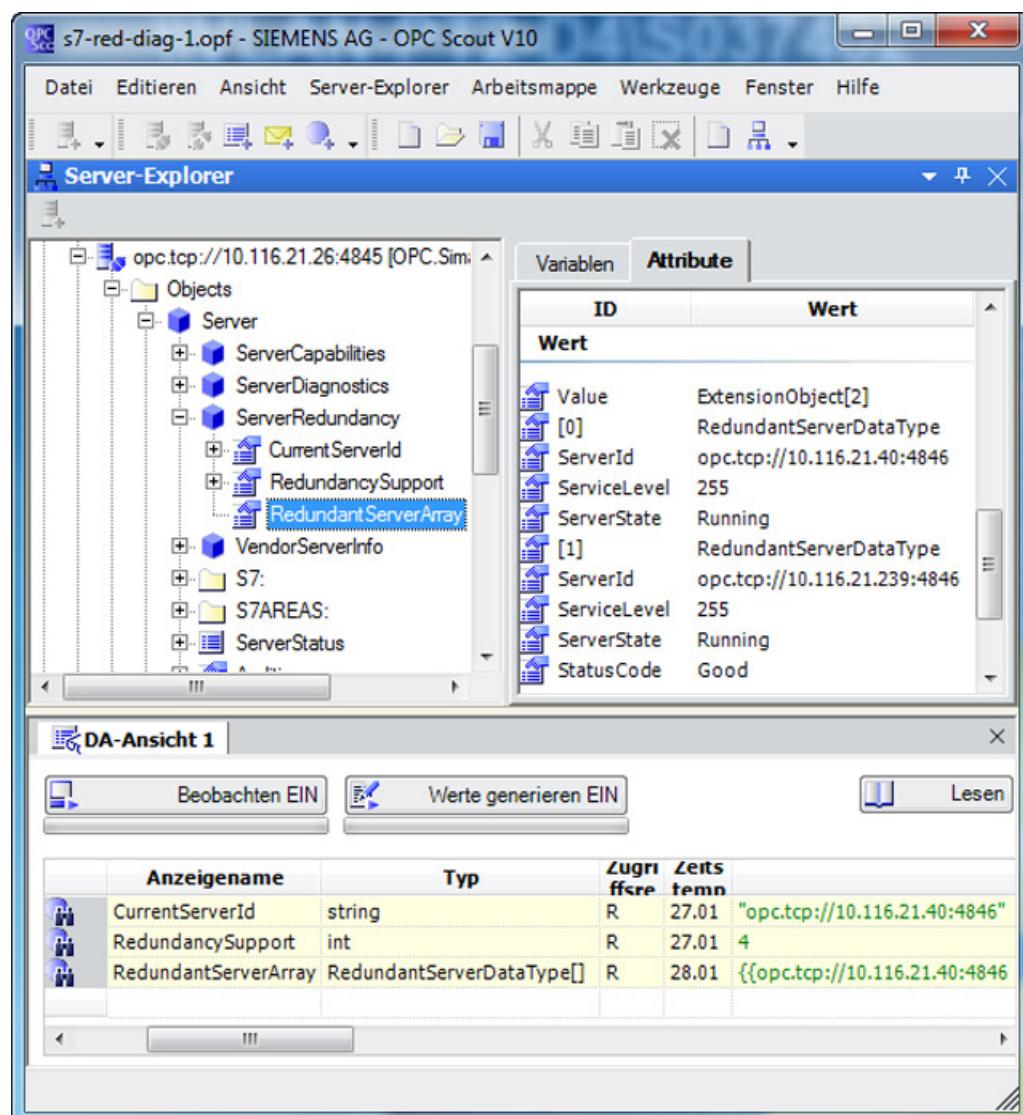
Mithilfe von Diagnosevariablen können Sie die Art der Redundanzunterstützung oder den Verbindungszustand zwischen OPC-UA-Client und OPC-UA-Server herausfinden. Folgende Diagnosevariablen werden von einem der OPC-UA-Server der Redundanzgruppe zur Verfügung gestellt. Die Diagnosevariablen können Sie sich am genormten Server-Objekt im Namensraum-Index "0" unter dem Server-Knoten im "Netzwerlastenausgleich-Manager" anzeigen lassen.

Unter dem Server-Knoten gibt es das Objekt "ServerRedundancy". Es ist ein Objekt vom Typ "ServerRedundancyType" mit der Property "RedundancySupport". Mithilfe dieses Objekts kann die Art der Redundanzunterstützung ausgelesen werden. Folgende Werte können mit Hilfe des Datentyps "enum\_OpcUa\_RedundancySupport" ausgelesen werden:

- OpcUa\_RedundancySupport\_None = 0,
- OpcUa\_RedundancySupport\_Cold = 1,
- OpcUa\_RedundancySupport\_Warm = 2,
- OpcUa\_RedundancySupport\_Hot = 3,
- OpcUa\_RedundancySupport\_Transparent = 4

Falls eine transparente Redundanz unterstützt wird, ist dieses Objekt vom Typ "TransparentRedundancyType". Dieser Objekttyp definiert die zusätzlichen Elemente "CurrentServerId" und "String". Obwohl im transparenten Betrieb der Redundanzgruppe alle Server die gleiche URI zur Verbindung aufweisen, kann mit der Kennung "CurrentServerId" der momentan verwendete Server der Redundanzgruppe identifiziert werden. Beachten Sie, dass der Wert nur innerhalb einer Session gültig ist.

Bei einem Ausfall eines Servers oder mehrerer Sessions können verschiedene Server der Redundanzgruppe dessen Aufgaben übernehmen. Trotzdem kann der Client die spezifische Datenquelle identifizieren. Im "RedundantServerArray" vom Typ "RedundantServerDataType" ist die Liste aller Server der Redundanzgruppe enthalten. In dieser Server-Liste sind verschiedene server-spezifische Informationen bzw. Werte (ServerId, ServiceLevel, ServerState) hinterlegt. Diese Werte können sich innerhalb einer Session ändern.



Die Struktur "RedundantServerDataType" beinhaltet u. A. folgende Elemente:

- ServerId

ID eines Servers. Ein String der einen eindeutigen Identifier für den Server der Redundanzgruppe enthält. Das kann nicht die Server-URI sein, da diese für alle Server identisch sein muss. Der String wird analog zur Server-URI gebildet. Als Anteil für den lokalen PC wird allerdings die eindeutige IP-Adresse verwendet, die auch für die Server-zu-Server Kommunikation verwendet wird.

- ServiceLevel

Ein Byte das anzeigt, wie gut die Datenübertragung zwischen Server und Client ist. "0" (Δ "UNKNOWN") gibt den schlechtesten und "255" (Δ "RUNNING") den bestmöglichen Zustand an. Dieser Wert zeigt den Zustand der Verfügbarkeit von redundanten Servern an.

- ServerState (enum OpcUa\_ServerState)

kann die folgenden Werte annehmen, wobei der Status "UNKNOWN" bedeutet, dass der Server nicht erreichbar ist:

- 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)

---

### **Hinweis**

Beachten Sie dass Sie den aktiven Server nicht mithilfe der Client-Diagnosevariablen für Redundanz, umschalten können.

---

# SIMATIC Computing

Mit der Software "SIMATIC Computing" als Teil der SIMATIC NET PC-Software erstellen Sie ohne großen Programmieraufwand einfache Anwendungen, mit denen Sie auf Prozessdaten zugreifen können. Die Prozessdaten können über ein beliebiges Kommunikationssystem von SIMATIC NET gelesen und geschrieben werden.

Bei "SIMATIC Computing" stehen Ihnen verschiedene .NET Assemblies zur Verfügung. Diese Controls können Sie entsprechend Ihrer Aufgabe konfigurieren und verschalten.

## OPC Data Controls für .NET

Für die .NET Schnittstellen steht ein SIMATIC NET OPC Data Control mit einfacher grafischer Bedienung zur Verfügung. Es können einfache Anwendungen mit dem Microsoft Visual Studio erstellt werden. Dabei können Prozessvariablen im Namensraum der SIMATIC NET OPC-Server gesucht und auf Oberflächen-Elemente, wie z. B. eine Textbox, verknüpft werden. Diese Funktionalitäten sind ganz ohne das Schreiben von Programmcode möglich. Weitere Informationen zu diesem Thema finden Sie im Kapitel "Auto-Hotspot".

Für erweiterte Anwendungen steht zusätzlich eine Programmierschnittstelle zur Verfügung, welche trotzdem die OPC-Schnittstellen weiter vereinfachen.

## Kompatibilität zu OPC

Die .NET Assemblies und ActiveX Controls von SIMATIC NET setzen direkt auf der OPC-Schnittstelle auf.

Anwendungen, die mit Hilfe der Controls von SIMATIC NET erstellt wurden, können nur mit OPC-Servern der SIMATIC NET PC-Software betrieben werden.

### VORSICHT

Wenn Sie eine Prozessvariable über die Controls verschalten, ist der Zugriff auf Prozessdaten hergestellt. Wenn Sie die Werte einer Prozessvariablen in einem Control ändern, können sich die Werte direkt im Prozess ändern.

Unvorhergesehene Änderungen von Prozessdaten können unvorhersehbare Reaktionen im Prozess auslösen, die zu Sach- und Personenschäden führen können.

Gehen Sie dementsprechend vorsichtig vor. Schränken Sie beispielsweise die Zugriffsrechte der Prozessvariablen ein. Installieren Sie einen physikalischen NOT-AUS-Schaltkreis für die Maschinen oder den Prozess.

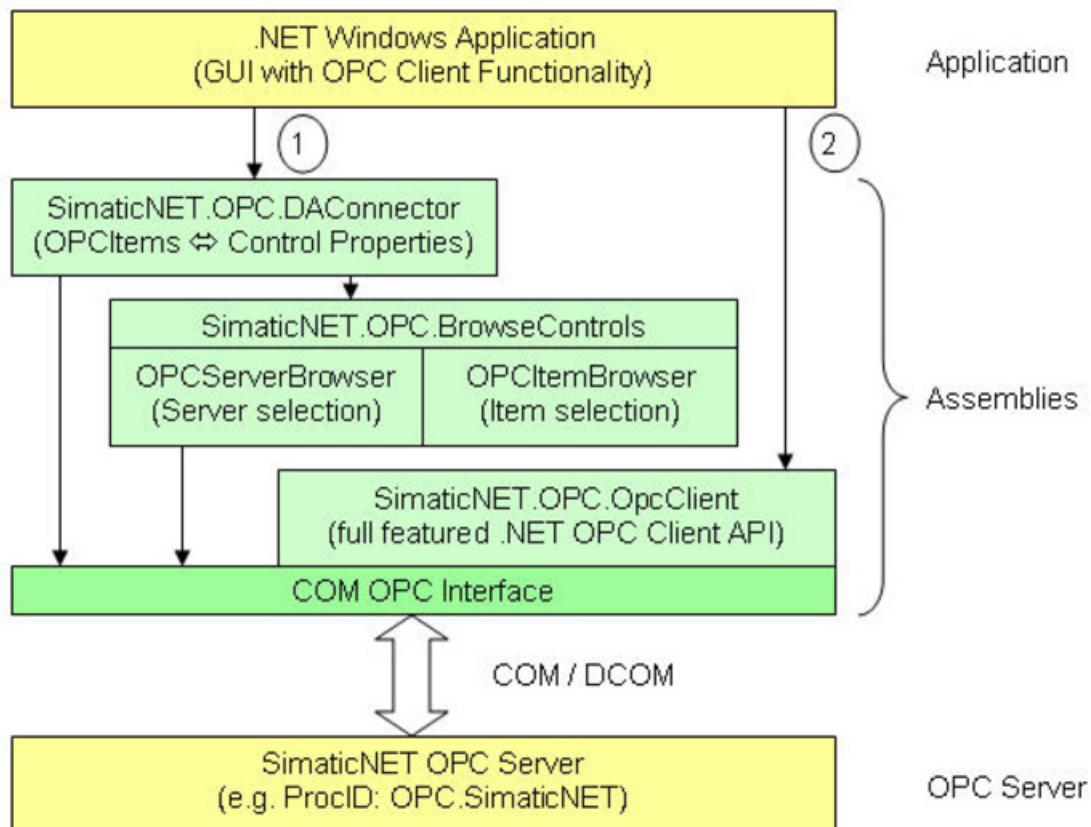
## 5.1 .NET OPC Client Control

### 5.1.1 Übersicht

#### Hintergrund

Die Entwicklungsumgebung Microsoft Visual Studio unterstützt das Einbinden von zusätzlichen Controls, die dann in den .NET Programmiersprachen verwendet werden können. Mit dem SimaticNET .NET OPC Client Control bietet Siemens eine einfache Möglichkeit Datenpunkte aus einem OPC-Server mit Standard-Controls aus dem MS Visual Studio, wie beispielsweise Textboxen, Labeln oder Buttons zu verschalten. Somit können einfache OPC-Client-Applikationen erstellt werden ohne eine einzige Zeile Code schreiben zu müssen. Dieser Zugriffsweg ist im Übersichtsbild mit "1" gekennzeichnet.

Anwender, die komplexere Applikationen schreiben wollen bzw. größere Datenmengen von OPC-Servern verarbeiten möchten, sollten die Schnittstelle SimaticNET .NET OPC Client API verwenden. Diese leistungsfähige Datenschnittstelle basiert auf einer C++ Bibliothek, vereinfacht aber den Zugriff über .NET. Dieser Zugriffsweg ist im Übersichtsbild mit "2" gekennzeichnet.



## Merkmale der .NET OPC Client Controls

Die folgenden Merkmale zeichnen die .NET Controls und die zugrunde liegende C++ Bibliothek aus:

- Der Benutzer benötigt kein detailliertes Wissen über die verschiedenen OPC Data Access-Schnittstellen.
- Die Komponente verdeckt komplett die verschiedenen Basistechnologien von OPC wie COM, DCOM, Webservices, SOAP und XML.
- Die Komponente verdeckt komplett das Verbindungshandling zu einem OPC-Server mit Verbindungsauflaufbau, Verbindungsüberwachung und erneutem Verbindungsauflaufbau im Fehlerfall.
- Die Entwicklung von OPC-Clientapplikation unter C-Sharp .NET und Visual Basic .NET ist mit den SimaticNET .NET OPC Client Controls ohne zusätzliche Programmierung möglich.
- Umsetzung der OPC-Daten aus verschiedenen OPC Data Access-Schnittstellen auf .NET Datentypen.
- Schnelles und einfaches Ermitteln von OPC-COM-Servern lokal und im Netzwerk.
- Grafisch unterstütztes Browsen von OPC-Items im ausgewählten Server.
- Performante und optimierte Client-Server-Kommunikation durch Implementierung der Kernfunktionalität in C++.
- Unterstützung von OPC UA und der Zertifikatsverwaltung für OPC UA.

---

### Hinweis

Mit diesen Controls können ausschließlich Verbindungen zu OPC-Servern der SimaticNET Produktfamilie aufgebaut werden. Verbindungen zu anderen OPC-Servern oder zu OPC-Servern von anderen Herstellern sind nicht möglich.

---

## 5.1.2 Schritt 1 - Installation

Die Installation der SimaticNET .NET OPC Client Controls erfolgt über das Setup der "SIMATIC NET PC Software". Die Assemblies werden im Pfad "%ProgramFiles%/Siemens/Simatic.Net/OPC2/OCX.NET" installiert. Das SimaticNET .NET OPC Client Control besteht aus den folgenden Dateien:

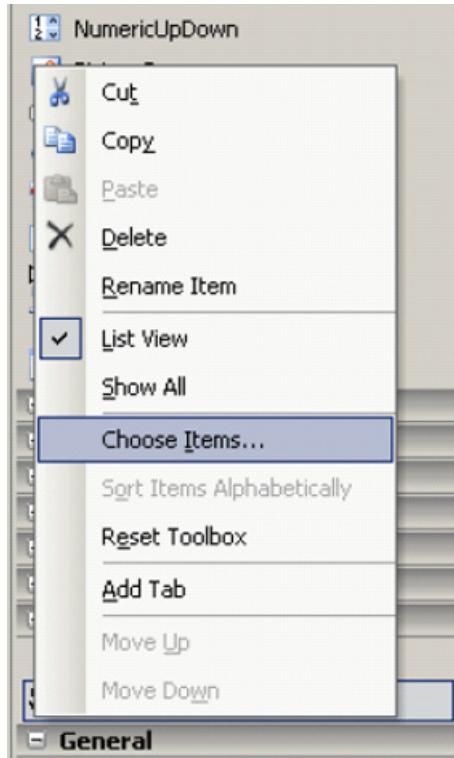
- "SimaticNET.OPC.DAConnector.dll"
- "SimaticNET.OPC.BrowseControls.dll"
- "SimaticNET.OPC.OpcClient.dll"

### 5.1.2.1 Referenzieren der Controls

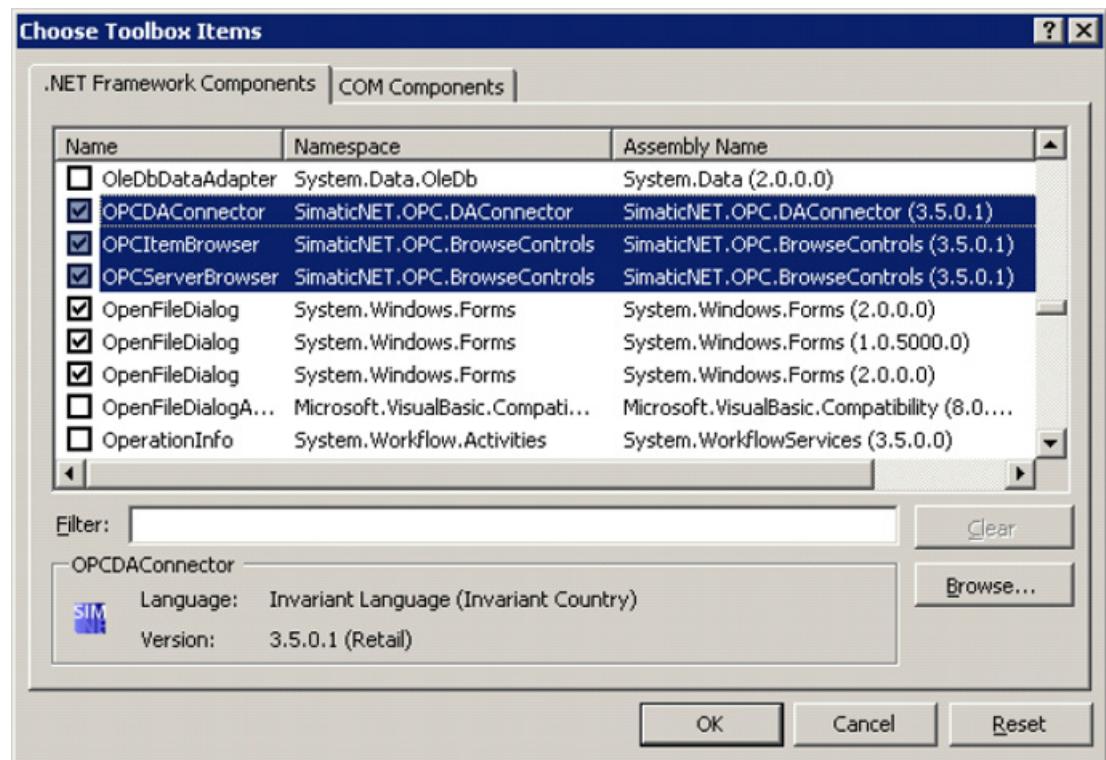
Vor der ersten Verwendung der SimaticNET .NET OPC Client Controls, müssen die Controls manuell in der Toolbox hinzugefügt werden.

Jeweils mit einem Rechtsklick in die Toolbox erhalten Sie ein Untermenü, um einen neuen Reiter über "Registerkarte hinzufügen" anzulegen bzw. neue Komponenten mit "Elemente auswählen" hinzuzufügen. In der folgenden Dialogbox ("Choose Toolbox Items") navigieren Sie ("Browse...") zu den oben genannten Installationspfad. Fügen Sie folgende Dateien hinzu:

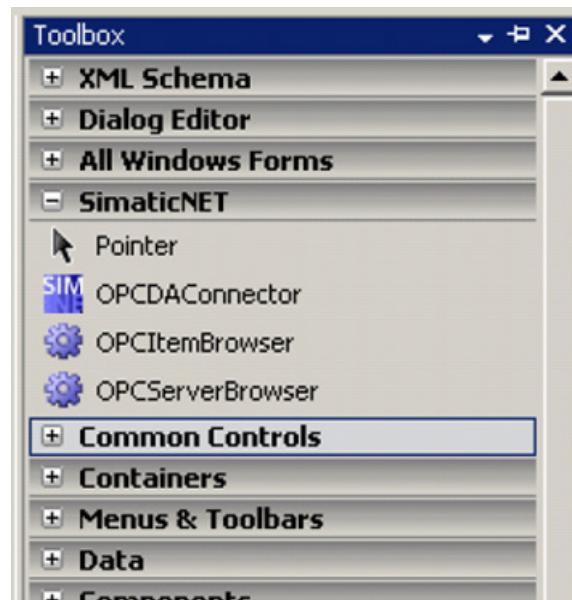
- "OPCDAConnector.dll"
- "OPCItemBrowser.dll"
- "OPCServerBrowser.dll"



⇒



In der Toolbox des Microsoft Visual Studio erscheint ein neues Register, das die einzelnen Komponenten enthält.



---

### **Hinweis**

Assemblies, die im MS Visual Studio referenziert werden, müssen auf einer lokalen Festplatte liegen und dürfen nicht auf einem Netzwerklaufwerk liegen. Dies ist eine Einschränkung der Microsoft .NET Entwicklungsumgebung.

---

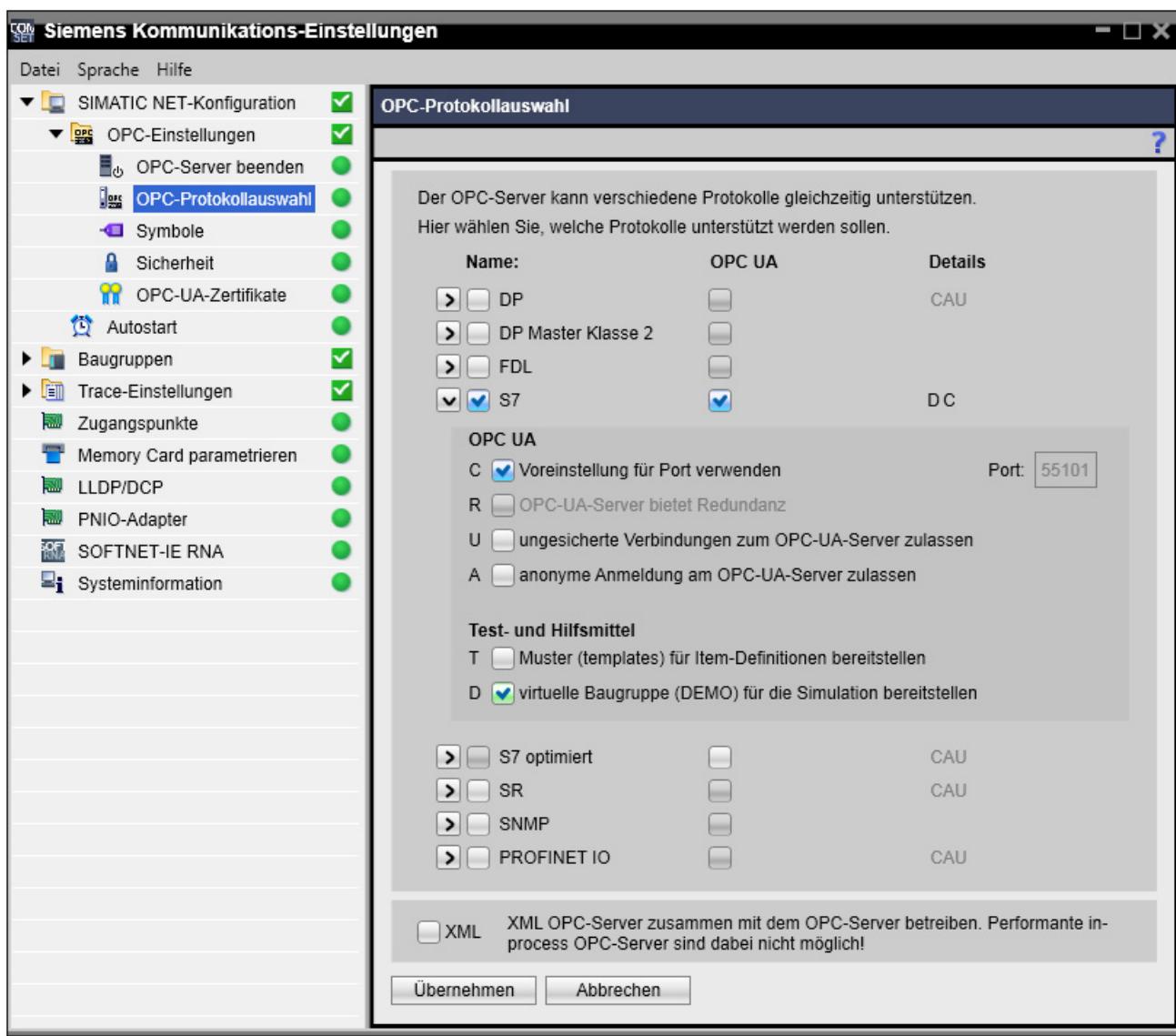
#### **5.1.2.2**

#### **Einstellung eines Beispielprogramms**

Damit das vorliegende Programm überhaupt lauffähig ist, muss die Simulationsverbindung des OPC-Servers aktiviert werden. Im Beispielprogramm werden Demo-Variablen der S7-Simulationsverbindung verwendet.

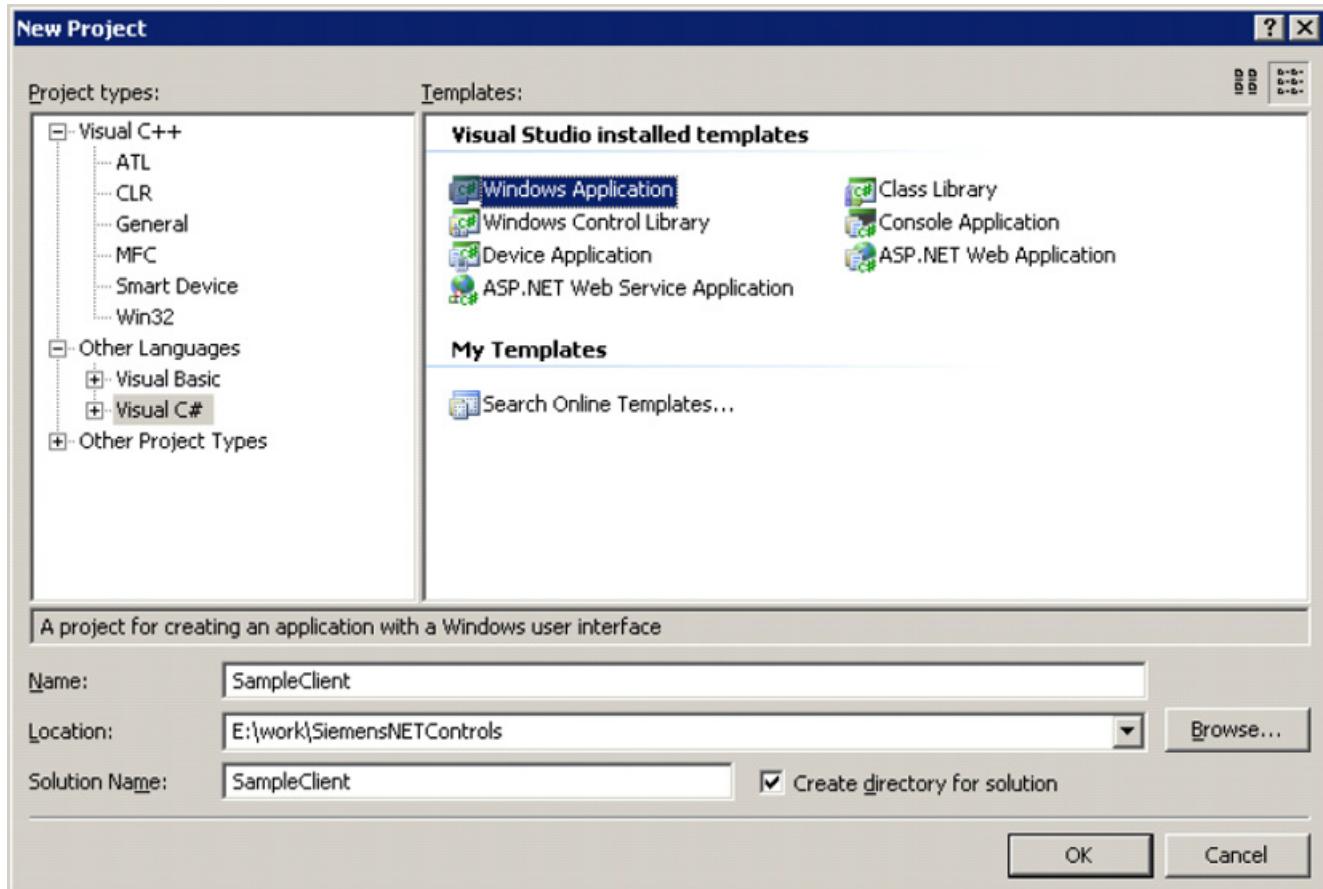
Gehen Sie folgendermaßen vor, um eine virtuelle Baugruppe (DEMO) für die Simulation bereitzustellen:

1. Starten Sie das Programm "Kommunikations-Einstellungen" über das Startmenü: "Start" > "SIMATIC" > "SIMATIC NET" > "Einstellungen" > "Kommunikations-Einstellungen"
2. Öffnen Sie im linken Navigationsfenster die Eigenschaftsseite für die OPC-Protokollauswahl.
3. Wählen Sie die S7-Verbindung aus und klicken Sie anschließend auf das Pfeilsymbol daneben.
4. Aktivieren Sie in der erweiterten Parameterliste das Kontrollkästchen D "virtuelle Baugruppe (DEMO) für die Simulation bereitstellen".
5. Bestätigen Sie die Auswahl, indem Sie die Schaltfläche "Übernehmen" klicken.
6. Beenden Sie das Programm "Kommunikations-Einstellungen".

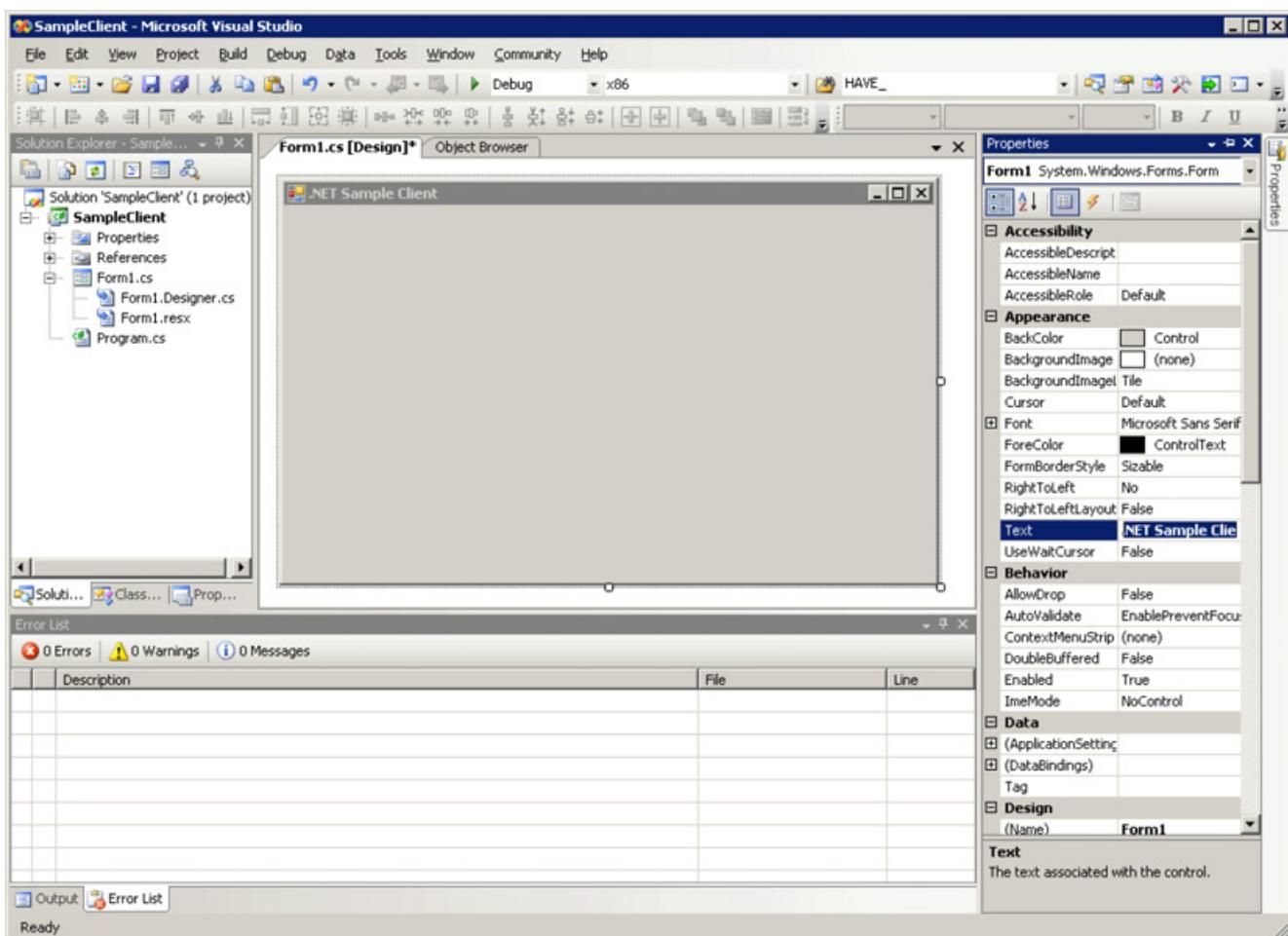


### 5.1.3 Schritt 2 - Controls in einer WINDOWS FORM

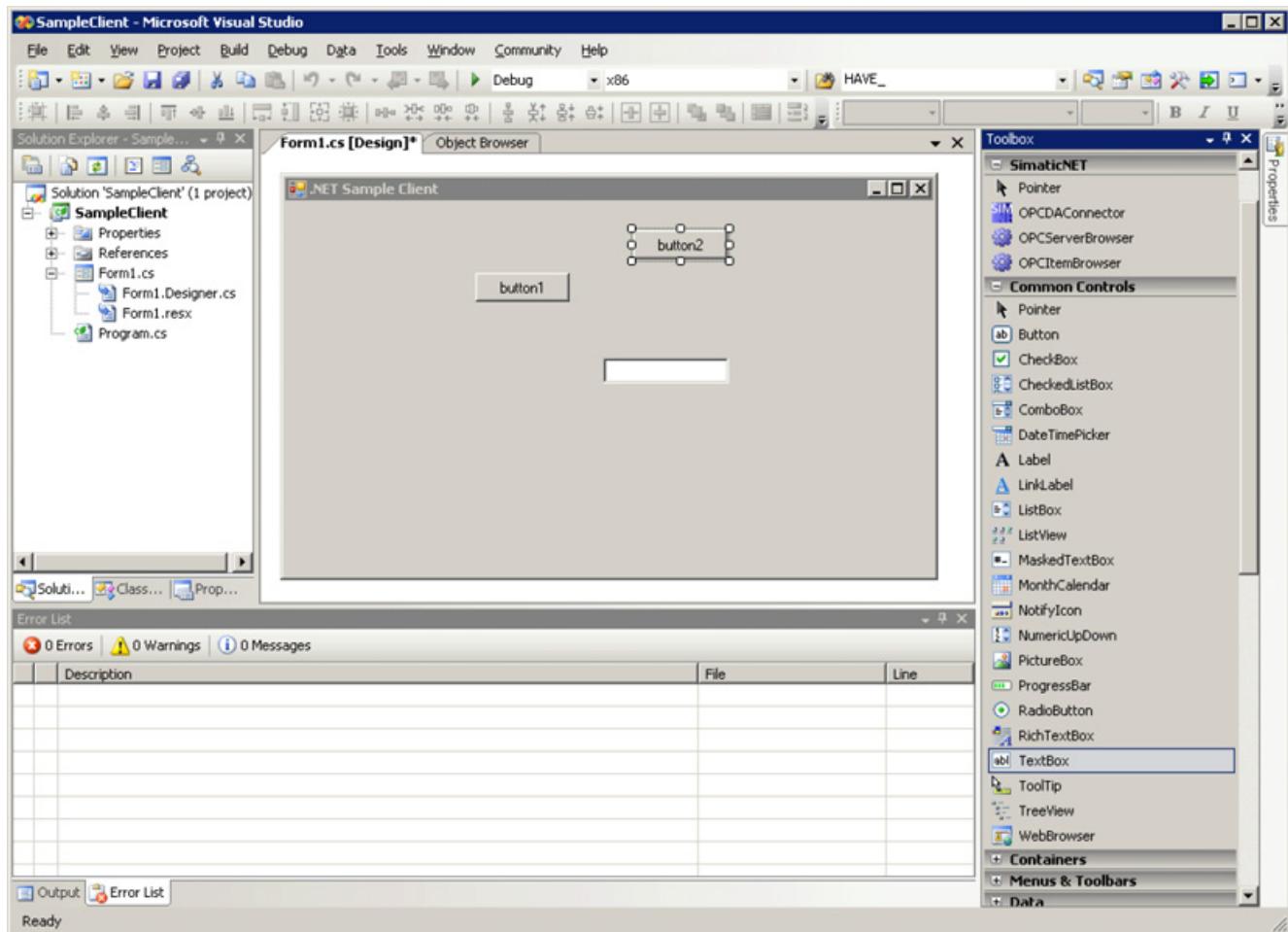
Erstellen Sie mit Microsoft Visual Studio ein neues C# -Projekt vom Template-Typ "Windows Application" (auch VB.NET ist hier möglich).



Sie erhalten ein leeres Applikationsfenster (Form1) dessen Anzeigetext (Caption) Sie umbenennen können.

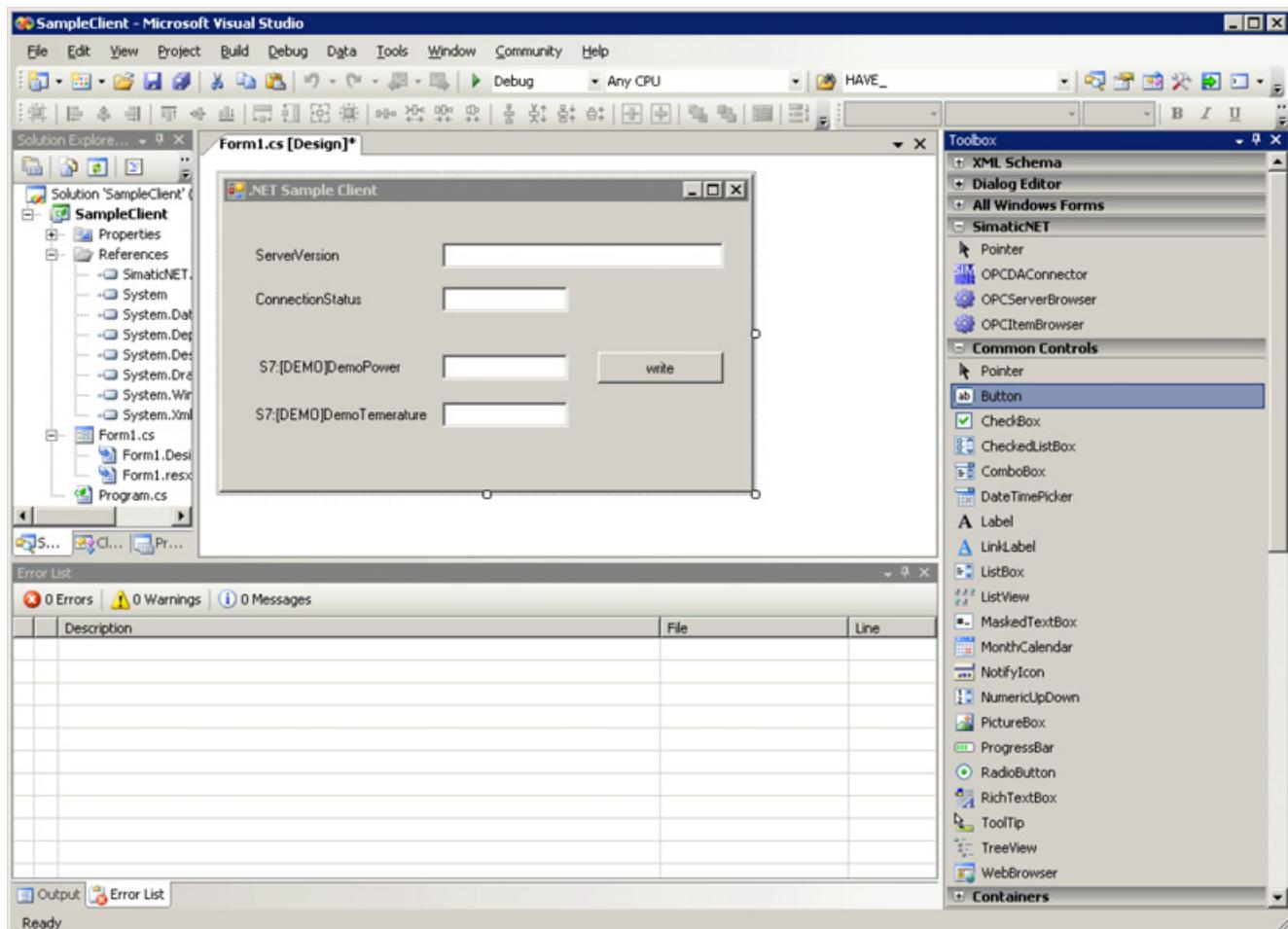


Öffnen Sie die Toolbox und ziehen Sie die erforderlichen Komponenten auf den Forms Designer ihres Beispielprogrammes.

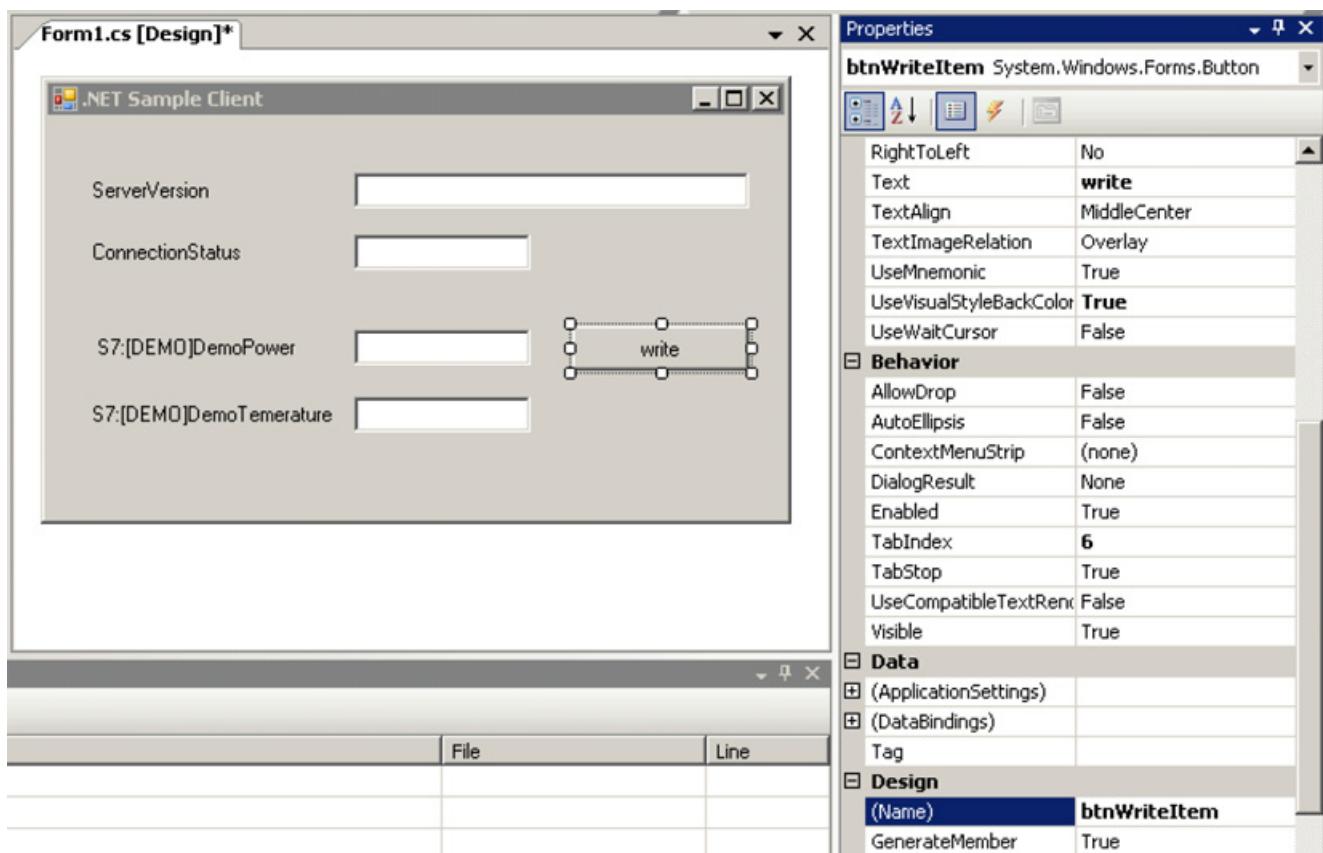


### 5.1.3.1 Windows Controls hinzufügen

Fügen Sie vier Bezeichner (Label), vier Textboxen (TextBox) sowie eine Schaltfläche (Button) entsprechend der Abbildung in das leere Form. Sie finden diese Komponenten im Reiter "Common Controls". Positionieren sie die Komponenten entsprechend der Abbildung.



Verwenden sie für dieses Beispiel möglichst die Namen, die in der folgenden Tabelle vorgegeben sind, da auf diese in der weiteren Erklärung verwiesen wird. Blenden Sie hierzu das Eigenschaftsfenster ein (Properties) und selektieren sie die Komponenten deren Eigenschaften Sie editieren möchten.

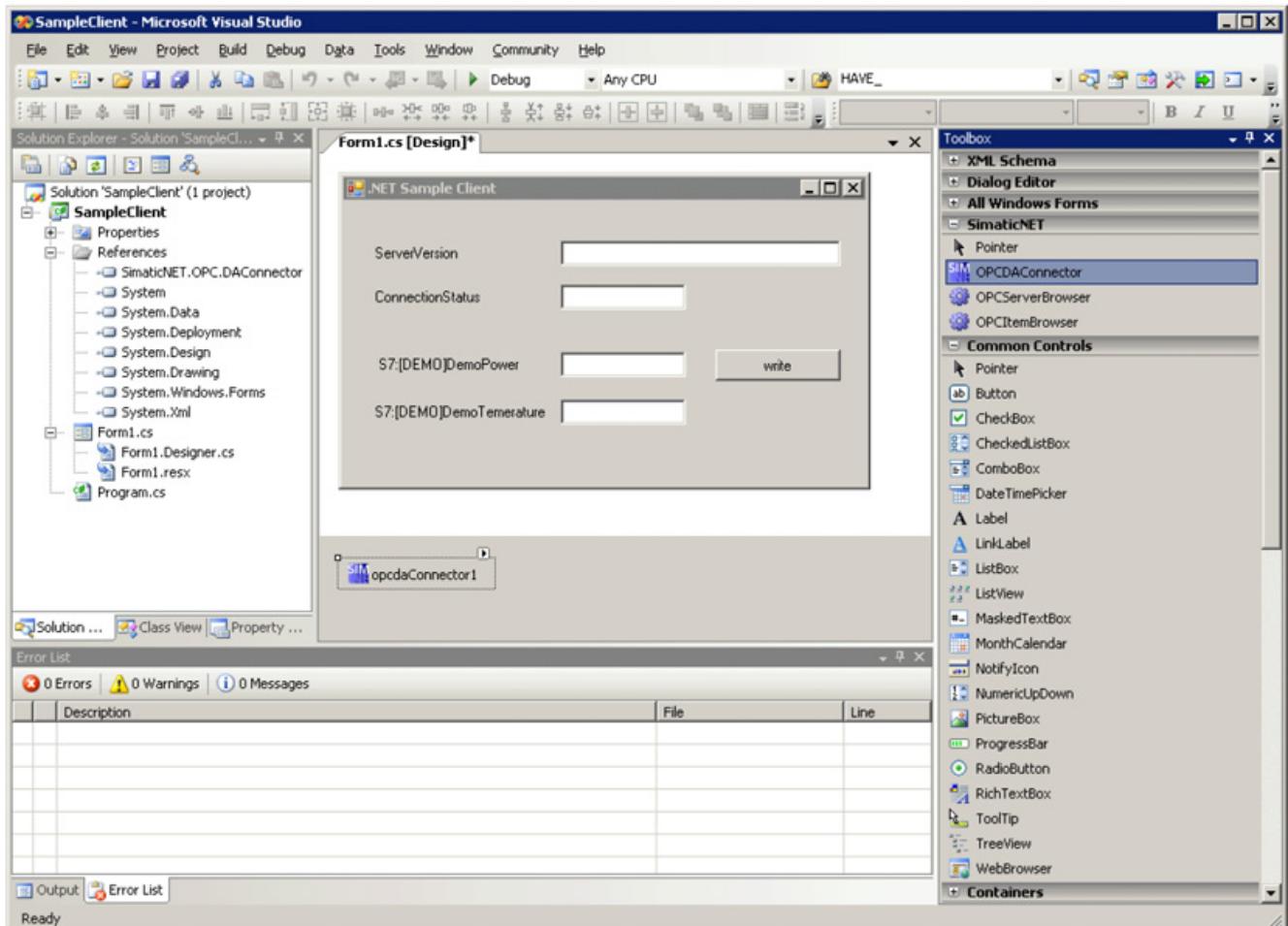


Wiederholen Sie diesen Vorgang für alle hinzugefügten Komponenten.

Originalname	Neue Eigenschaft (Name)	Neue Eigenschaft (Text)	Neue Eigenschaft (Enabled)
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 .NET OPC Client Control hinzufügen

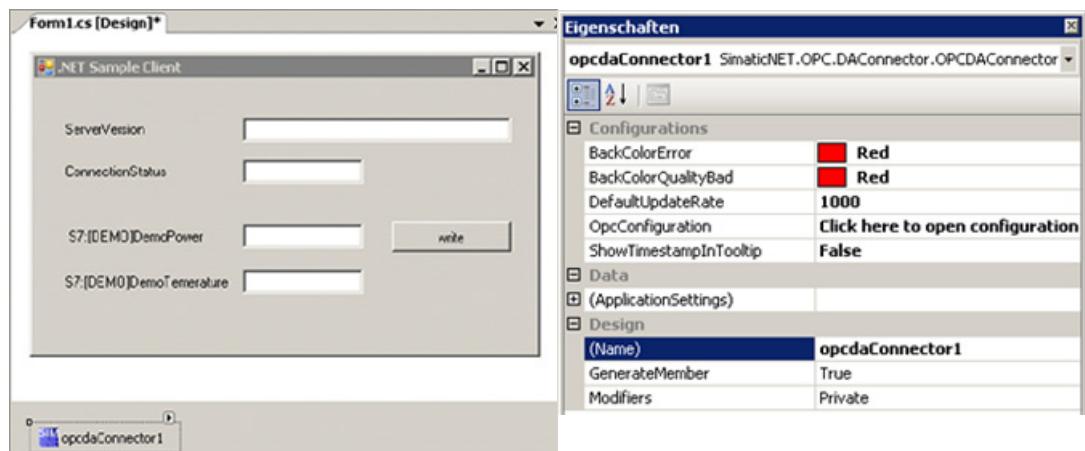
Nehmen Sie nun das Control "OPCDAConnector" aus der Toolbox und ziehen es auf das Applikationsfenster. Durch diesen Schritt wird die komplette OPC-Client-Funktionalität in Ihre Applikation eingebunden.



Generelle Einstellungen des OPC Client Controls können im Eigenschaftsfenster (Properties) vorgenommen werden.

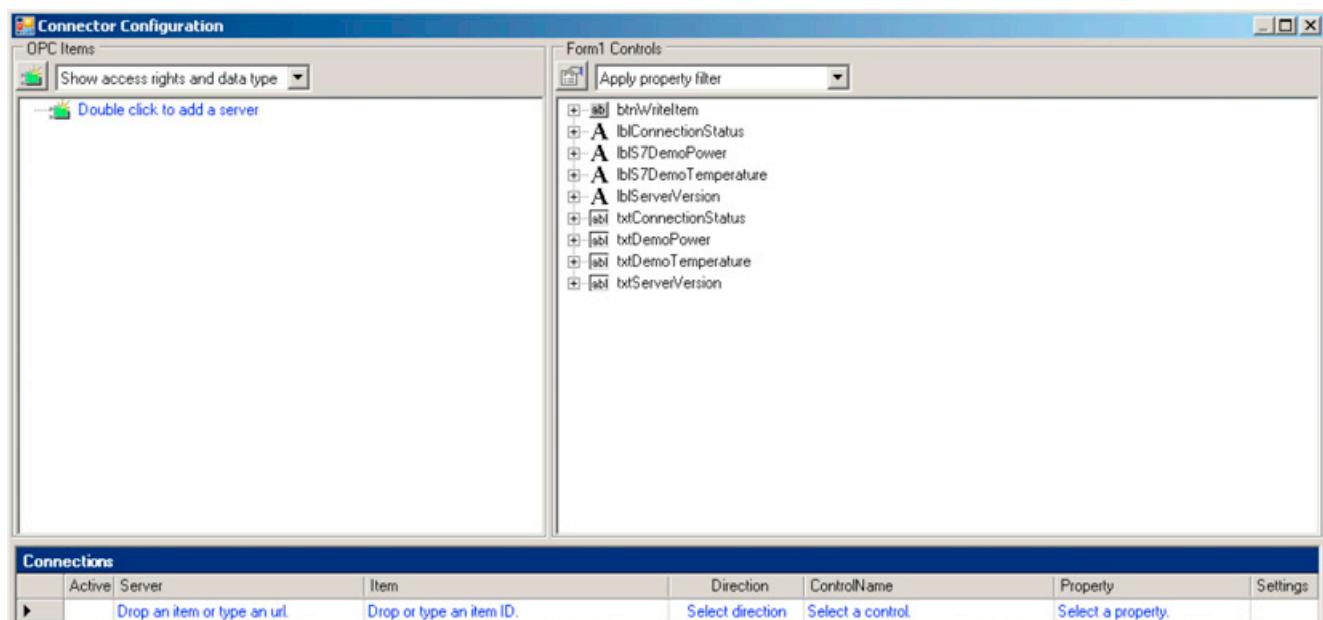
- Über "Konfigurieren" (Configuration) gelangen Sie in den Konfigurationsdialog zum Verknüpfen der OPC Items.
- Die Aktualisierungszeit (DefaultUpdateRate) wird in Millisekunden angegeben und wird an den OPC-Server weitergegeben, um Datenänderungen an die Clientapplikation zu liefern.
- Die Hintergrundfarbe (BackColorError) wird für alle Controls eingestellt, die mit OPC Items verknüpft werden. Wenn für das Item ein OPC-Fehler auftritt, wird die Hintergrundfarbe des verschalteten Controls auf die hier eingestellte Farbe geändert.

- Die Hintergrundfarbe (BackColorQualityBad) wird für alle Controls eingestellt, die mit OPC Items verknüpft werden. Wenn die Qualität des gelieferten OPC Wertes auf "Bad" umschlägt, wird die Hintergrundfarbe des verschalteten Controls auf die hier eingestellte Farbe geändert.
- Die Anzeige des OPC-Zeitstempels erscheint im Tooltip des verknüpften Controls und zeigt zusätzlich zum Wert auch den Zeitpunkt an (in UTC Zeit).



#### 5.1.4 Schritt 3 - Datenverknüpfungen konfigurieren

Um diesen Dialog zu öffnen, klicken Sie im Eigenschaftsfenster (Properties) des OPC Client Controls in das Feld "Konfigurieren" (configure) oder Doppelklicken Sie auf die Komponente im Forms Designer.

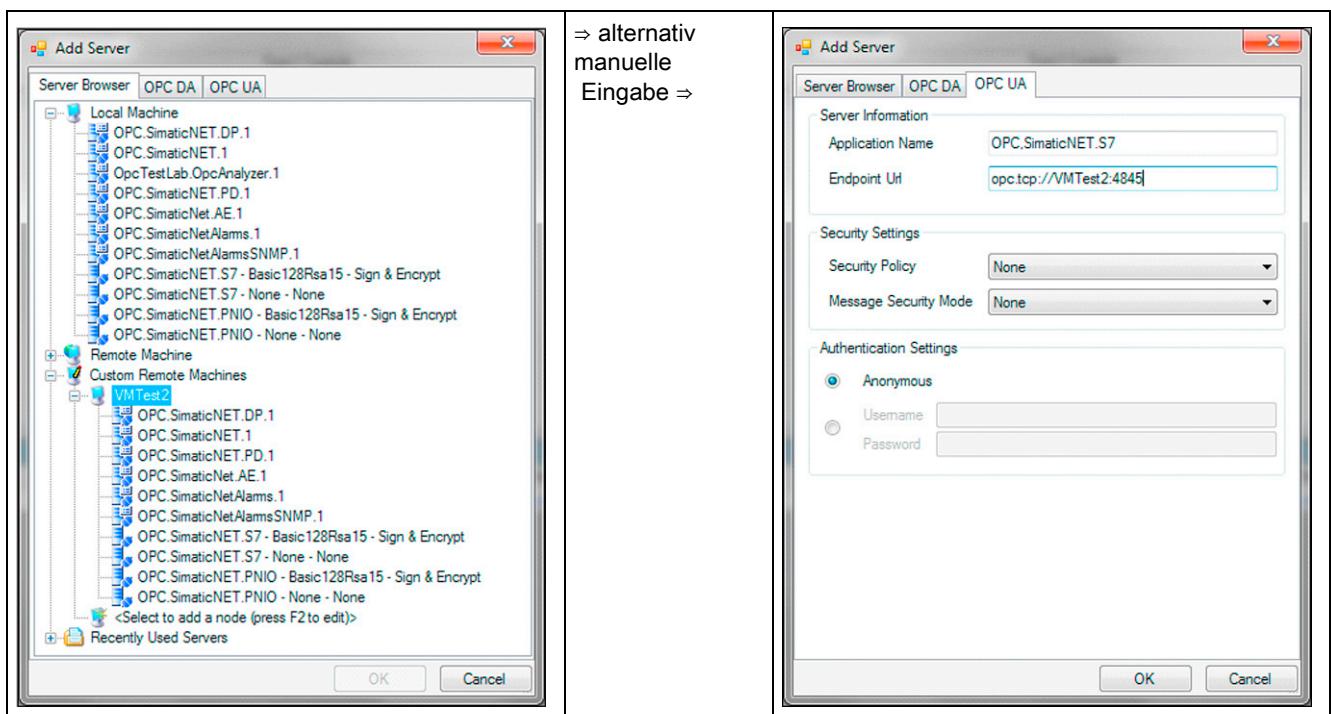


Das Dialogfenster besteht im Wesentlichen aus drei Teilen:

- Oben links: dem OPC-Browser-Fenster  
hier werden der OPC-Server und anschließend die OPC-Items ausgewählt, die verschaltet werden sollen
- Oben rechts: dem Windows Komponenten Browser  
hier werden die Eigenschaften (Properties) der Windows-Controls ausgewählt, die verschaltet werden sollen
- Unten: der Liste aller Verknüpfungen  
in diese Liste werden per "Drag&Drop" jeweils ein OPC-Item und ein Control-Property gezogen und damit miteinander verknüpft

#### 5.1.4.1 OPC-Items browsen

Im oberen linken Fenster des OPC-Client-Connector-Dialoges werden OPC-Server und deren Variablen (OPC-Items) angezeigt. Hierzu muss zunächst ein OPC-Server hinzugefügt werden. Den Dialog zum Hinzufügen von OPC-Servern (Add Server) öffnen Sie durch anklicken des grünen Symbols oder durch doppelklicken auf den angezeigten Text.



Sie können einen lokalen OPC-Server wählen, die spätere Applikation wird dann auch versuchen lokal diesen Server zu starten. Wenn Sie einen OPC-Server im Netzwerk (Remote Machines) auswählen, wird der Rechnername mit in Ihrer Applikation gespeichert und später auch verwendet. Stellen Sie sicher dass Sie ausreichende DCOM-Rechte besitzen, um entsprechende entfernte OPC-Server starten und verbinden zu dürfen. Im zweiten Register (OPC DA) gibt es die Möglichkeit einen OPC-Server manuell anzugeben (ProgID und Host-Name bzw. IP-Adresse).

---

**Hinweis**

Mit diesen Controls können ausschließlich Verbindungen zu OPC-Servern der SIMATIC NET Produktfamilie aufgebaut werden. Verbindungen zu anderen OPC-Servern oder zu OPC-Servern von anderen Herstellern sind nicht möglich.

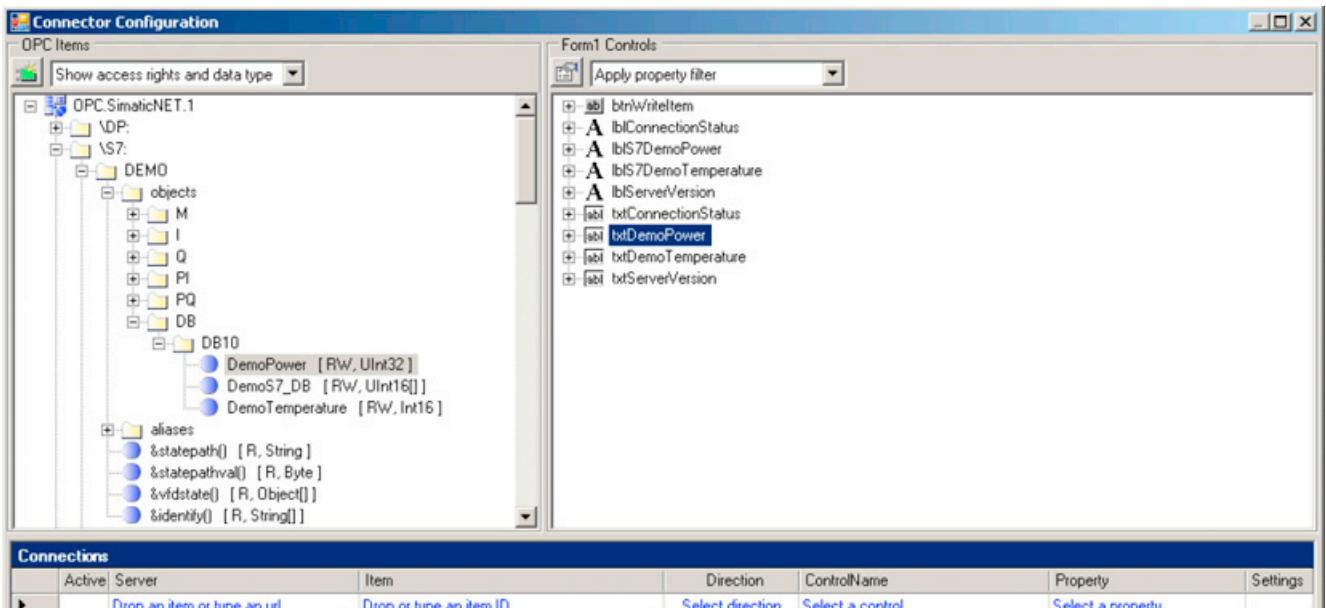
---

Wenn Sie einen OPC-Server hinzugefügt haben, können Sie durch seinen Adressraum navigieren und sich die OPC-Items, die Sie verschalten möchten, aussuchen. Beachten Sie dass die OPC-Items die nötigen Zugriffsrechte besitzen müssen damit eine Verschaltung überhaupt möglich ist. Sie können die Ansicht des Browsers umschalten, um sich Zugriffsrechte und Datentyp anzeigen zu lassen indem Sie die ComboBox links oben im Fenster umschalten. Wählen Sie zwischen zwei möglichen Ansichten:

- Show access rights and data type  
Zeigt neben dem Zugriffsrecht auf Item-Ebene auch den Datentyp des OPC-Items
- Show only item names  
Zeigt nur den jeweils den Namen des OPC-Items an

Da die Windows Controls zum Teil sehr viele Eigenschaften (Properties) besitzen, können Sie für diese Ansicht Filter setzen indem Sie die ComboBox oben rechts im Fenster umschalten. Wählen Sie zwischen zwei möglichen Ansichten:

- **Apply property filter**  
Hier wird entsprechend der Filtereinstellungen angezeigt. Es kann auf den Datentyp, die Zugriffsrechte und die Ebene der Eigenschaften gefiltert werden. Die Filtereinstellungen können über einen Dialog gesetzt werden, der über den Button links neben der ComboBox (Apply Filter) zu erreichen ist, siehe Kapitel "Windows Control Eigenschaften browsen (Seite 576)".
- **Show all properties**  
Zeigt alle Eigenschaften der Windows Controls.



Ziehen Sie nun die OPC-Items "&version()", "&statepath()" und die Variablen der Demo-Verbindung "DemoPower" und "DemoTemperature" in die Verknüpfungsliste des unteren Fensters.

#### Hinweis

Bei der Verbindung zu einem Unified Architecture Server (UA-Server) kann es je nach gewähltem Endpunkt erforderlich sein, dass zwischen Client und Server Zertifikate ausgetauscht werden, bevor eine Verbindung möglich ist (siehe Kapitel "OPC-UA-Zertifikat-Erstellung für das NET OPC Data Control (Seite 588)").

## Zugriffsrechte

Um ein OPC-Item lesen und schreiben zu können, muss es die Zugriffsrechte RW (read/writeable) besitzen.

## Datentyp

Der Datentyp des OPC-Items ist ebenfalls zu beachten, denn er sollte zum Datentyp der Eigenschaft (Property) des Windows Control passen mit dem es verschaltet werden soll. Das OPC .NET Client Control fordert die OPC-Items zwar in dem Datentyp an der zum verschalteten Property passt, jedoch sind nicht alle Konvertierungen möglich bzw. werden nicht alle Konvertierungen vom OPC-Server unterstützt.

## Status der Verknüpfung

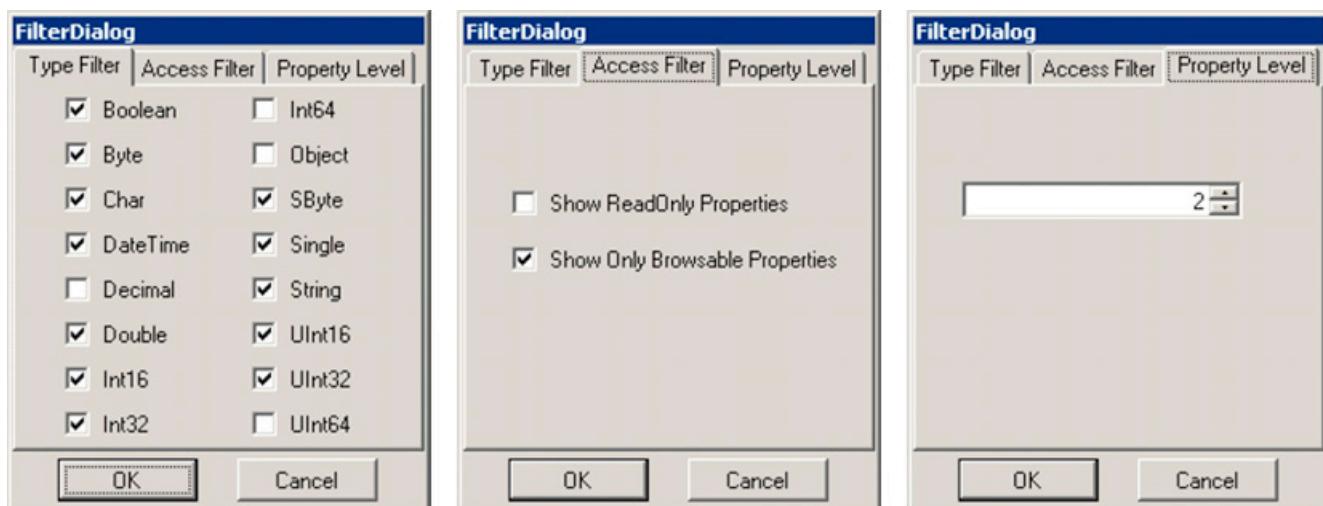
Falls eine Konvertierung nicht möglich ist oder einen Datenverlust zur Folge haben kann, wird mit entsprechenden Symbolen in der Verknüpfungsliste (Connections) darauf hingewiesen. Fahren Sie mit der Maus über das Symbol, um weitere Informationen zu erhalten.

Folgende Symbole sind möglich:

- rotes Kreuz  
die Verschaltung ist nicht möglich bzw. unvollständig
- oranger Haken  
die Verschaltung ist möglich, aber die Datentypen passen nur bedingt
- grüner Haken  
die Verschaltung wird funktionieren

### 5.1.4.2 Windows Control Eigenschaften browsen

Im oberen rechten Fenster des Dialogs "Connector Configuration" werden alle Windows-Komponenten angezeigt, die im Applikationsfenster (Form1) enthalten sind. Die Eigenschaften (Properties) die verschaltet werden sollen, können selektiert werden, indem die hierarchische Ansicht aufgeblättert wird. Um die Ansicht übersichtlicher zu gestalten, können Filter gesetzt werden, um Eigenschaften auszublenden, die aufgrund ihres Datentyps nicht zur Verschaltung mit OPC-Items in Frage kommen.



## Datentyp

Der Datentyp der jeweiligen Eigenschaft wird mit einem Symbol angezeigt. Die Eigenschaft "Text" einer Textbox ist beispielsweise von Typ "String". Da alle OPC-Datentypen in einen String konvertierbar sind, kann jeder Wert eines OPC-Items in einer Textbox angezeigt werden.

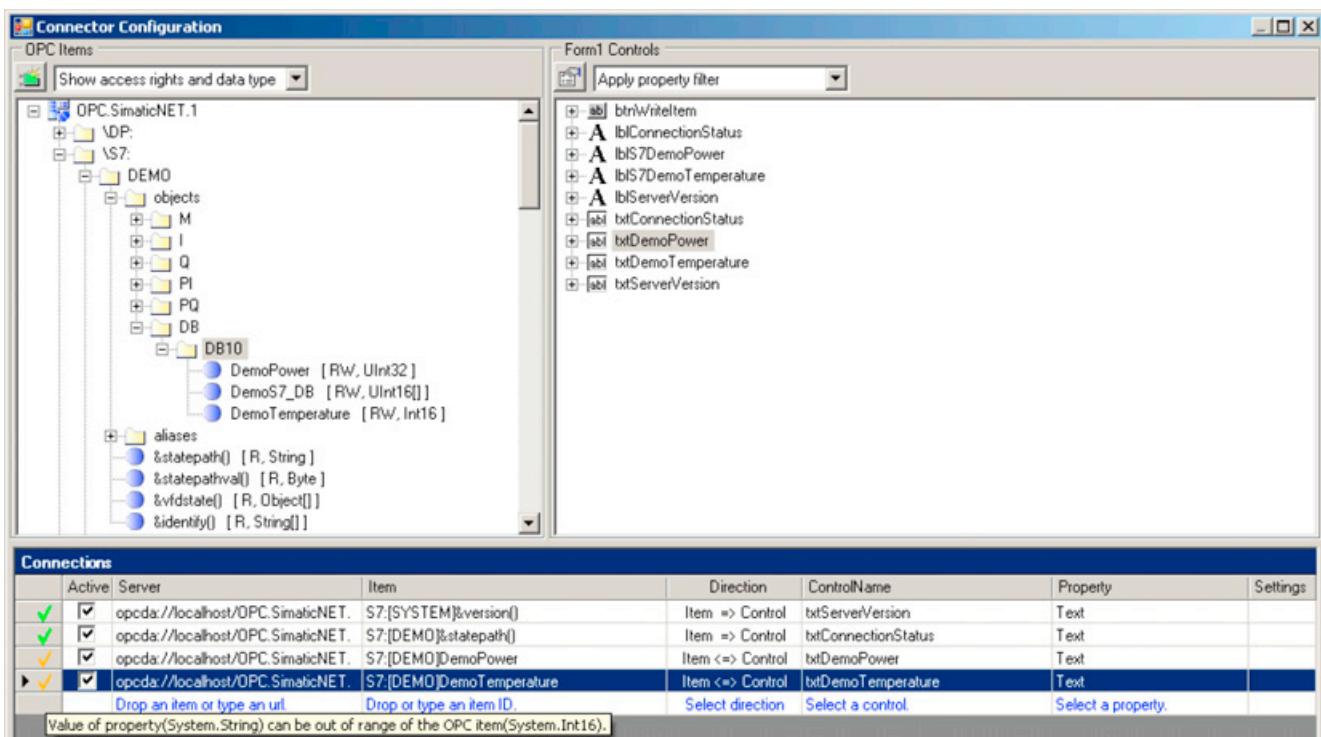
## Zugriffsrechte

Hier ist darauf zu achten, dass das ausgewählte Property die nötigen Zugriffsrechte besitzt und die Verknüpfung mit einem OPC-Item in Frage kommt. Einige Eigenschaften dürfen beispielsweise nur gelesen aber nicht beschrieben werden.

Ziehen Sie nun die Text-Eigenschaften der entsprechenden Textbox-Controls des Windows-Forms per "Drag&Drop" in das untere Fenster (Connections). Damit wird eine Verknüpfung mit dem OPC-Item dieser Zeile hergestellt.

### 5.1.4.3 Liste der Verknüpfungen erzeugen

Im unteren Fenster des Dialogs "Connector Configuration" wird die Liste der erstellten Verknüpfungen (Connections) angezeigt. Durch ziehen und fallen lassen (Drag & Drop) werden jeweils ein OPC-Item mit einer Windows Control-Eigenschaft verknüpft und die Datenübertragungsrichtung (Direction) wird eingestellt. Hiermit wird festgelegt, in welche Richtung die Daten übertragen werden.



Die Verknüpfungsliste (Connections) überprüft selbstständig ob die durchgeföhrte Verknüpfung funktioniert und zeigt dies mittels verschiedener Symbole an (grüner Haken, orangener Haken, rotes Kreuz). Weitere Informationen erhalten Sie wenn Sie mit der Maus

über das Symbol fahren. Die angezeigten Konflikte müssen aufgelöst werden, um eine funktionstüchtige Verbindung herzustellen.

Folgende Symbole sind möglich:

- rotes Kreuz  
die Verschaltung ist nicht möglich bzw. unvollständig
- oranger Haken  
die Verschaltung ist möglich, aber die Datentypen passen nur bedingt
- grüner Haken  
die Verschaltung wird funktionieren

### Nachträgliches Editieren und Deaktivieren

Verknüpfungen können in der Liste nachträglich bearbeitet werden, klicken Sie hierzu in das entsprechende Feld und editieren Sie die Zelle. Mit dem Haken (Active) kann eine Verknüpfung temporär deaktiviert werden.

### Mehrfachverschaltung

OPC-Items können mehrfach mit verschiedenen Windows-Controls bzw. mit verschiedenen Eigenschaften desselben Controls verschaltet werden.

### Übertragungsrichtung

Die Übertragungsrichtung der Daten erfolgt entsprechend der nachfolgenden Tabelle.

Richtung	Funktion	Beschreibung
Item ⇒ Control	OPC-Item wird gelesen	Bei jeder Änderung des Wertes des OPC-Items wird dieser Wert auf das verknüpfte Property eines Controls geschrieben
Item ⇐ Control	OPC-Item wird geschrieben	Bei Auslösen eines Triggerereignisses wird der Wert des Properties auf das verknüpfte OPC-Item geschrieben
Item ⇄ Control	Bidirektionale Übertragung	Jede Werteänderung des OPC-Items wird in das Property geschrieben und beim Auslösen eines Triggers wird der aktuelle Wert des Properties in das OPC-Item geschrieben. Das Schreiben bei Werteänderung des Items kann verhindert werden, indem die Option "Disable Datachange when Control has focus" gesetzt wird.

Stellen Sie die Übertragungsrichtung der verknüpften Datenverbindungen richtig ein. Die OPC-Items "&version" und "&statepath" sind read-only und werden mit "Item ⇒ Control" mit den Text-Properties der Windows Controls "txtServerVersion" und "txtConnectionStatus" verbunden. Für die OPC-Items "DemoPower" und "DemoTemperature" wählen Sie bidirektionale Übertragung (Item ⇄ Control).

Die bidirektionale Verschaltung der Variablen wird als "funktionstüchtig" angezeigt aber mit einem orangen Haken gekennzeichnet. Der Wert des OPC-Items (Integer) kann zwar problemlos in die Text-Eigenschaft des Controls (String) geschrieben werden, aber in der Gegenrichtung besteht die Möglichkeit eines Fehlers, denn es könnten sehr große Zahlen oder sogar Buchstaben in die Textbox eingegeben werden, die dann nicht mehr auf das OPC-Item schreibbar sind bzw. zu Fehlern führen.

#### 5.1.4.4 Trigger zum Schreiben

Um auf OPC-Items zu schreiben; sind Triggerereignisse erforderlich. Diese Trigger müssen konfiguriert werden und lösen dann den Schreibvorgang aus. Durch Klicken in die entsprechende Zelle der äußerst rechten Spalte (Settings) in der Verbindungsliste (Connections) des Dialogs "Connector Configuration" erscheint hier eine Schaltfläche und für jede Verknüpfung, die ein OPC-Item Schreiben soll, muss ein Triggerereignis konfiguriert werden.

Connections							
Active	Server	Item	Direction	ControlName	Property	Settings	
✓	opcda://localhost/OPC.Sim	S7:[SYSTEM]&version()	Item => Control	txtServerVersion	Text		
✓	opcda://localhost/OPC.Sim	S7:[DEMO]&statepath()	Item => Control	txtConnectionStatus	Text		
▶ ✓	opcda://localhost/OPC.Sim	S7:[DEMO]DemoPower	Item <=> Control	txtDemoPower	Text		
✓	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.		

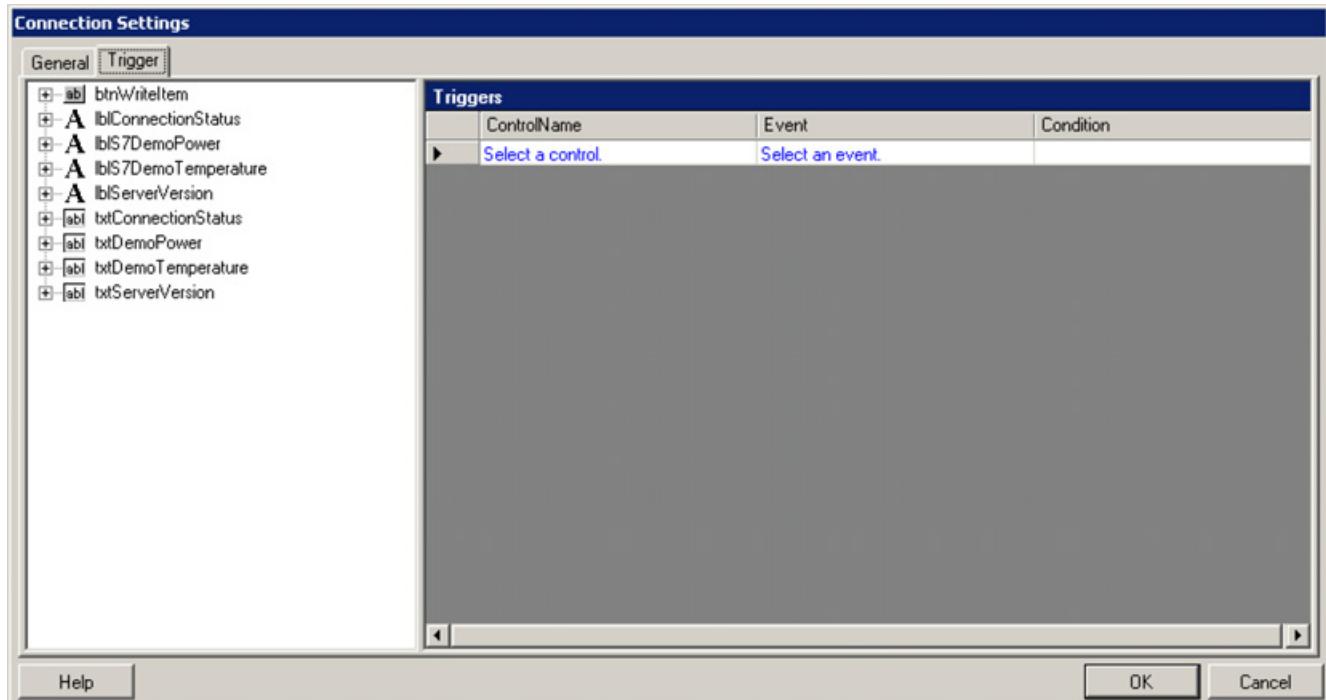
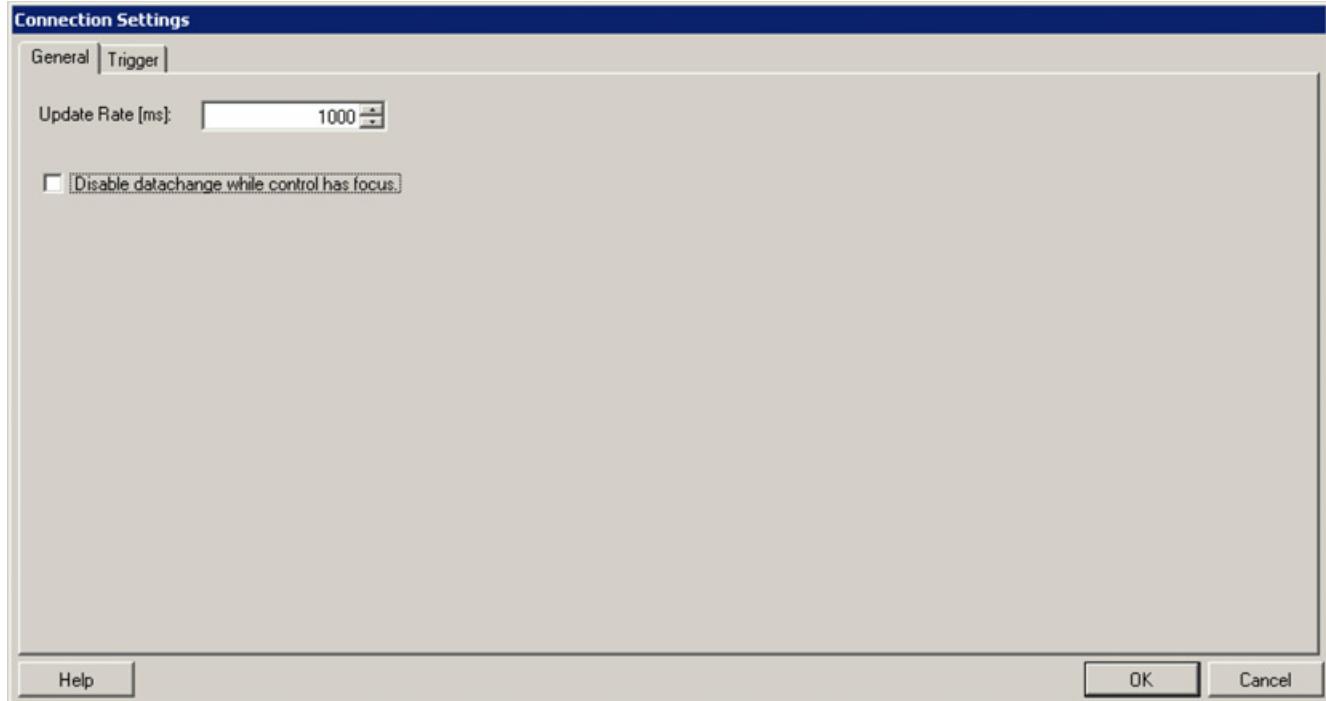
Der Dialog zum Konfigurieren des Triggerereignises einer Verknüpfung besitzt zwei Reiter (Tabs) zum Einstellen des allgemeinen und des speziellen Verhaltens.

#### Allgemeine Einstellungen:

- Updaterate  
die Aktualisierungsrate in Millisekunden mit der das OPC-Item beim Server zur Überwachung angemeldet wird
- Disable DataChange when Control has focus  
die Aktualisierung wird temporär abgeschaltet so lange das Control den Fokus hat; diese Einstellung sollte bei Bidirektionalen Übertragungen verwendet werden

## Trigger Einstellungen

- Auswahl des Ereignisses
- Einstellung spezieller Bedingungen (Condition) für dieses Ereignis



## Mehrere Trigger-Ereignisse

Es können auch mehrere Trigger-Ereignisse konfiguriert werden. Der Schreibvorgang wird immer dann für diese Verschaltung ausgeführt wenn eins dieser Trigger-Ereignisse ausgelöst wird.

Im linken Teil des Trigger-Dialogs wird ein Ereignis (Event) des zugehörigen Windows-Controls selektiert und in die Liste der Trigger gezogen (Drag & Drop). Für Events, die Parameter mitliefern, können diese als Bedingung (Condition) geprüft werden.

Für die Textboxen "txtDemoPower" und "txtDemoTemperature" schalten Sie die Aktualisierung der Felder ab wenn das Control den Fokus hat (allgemeine Einstellung des Triggers).

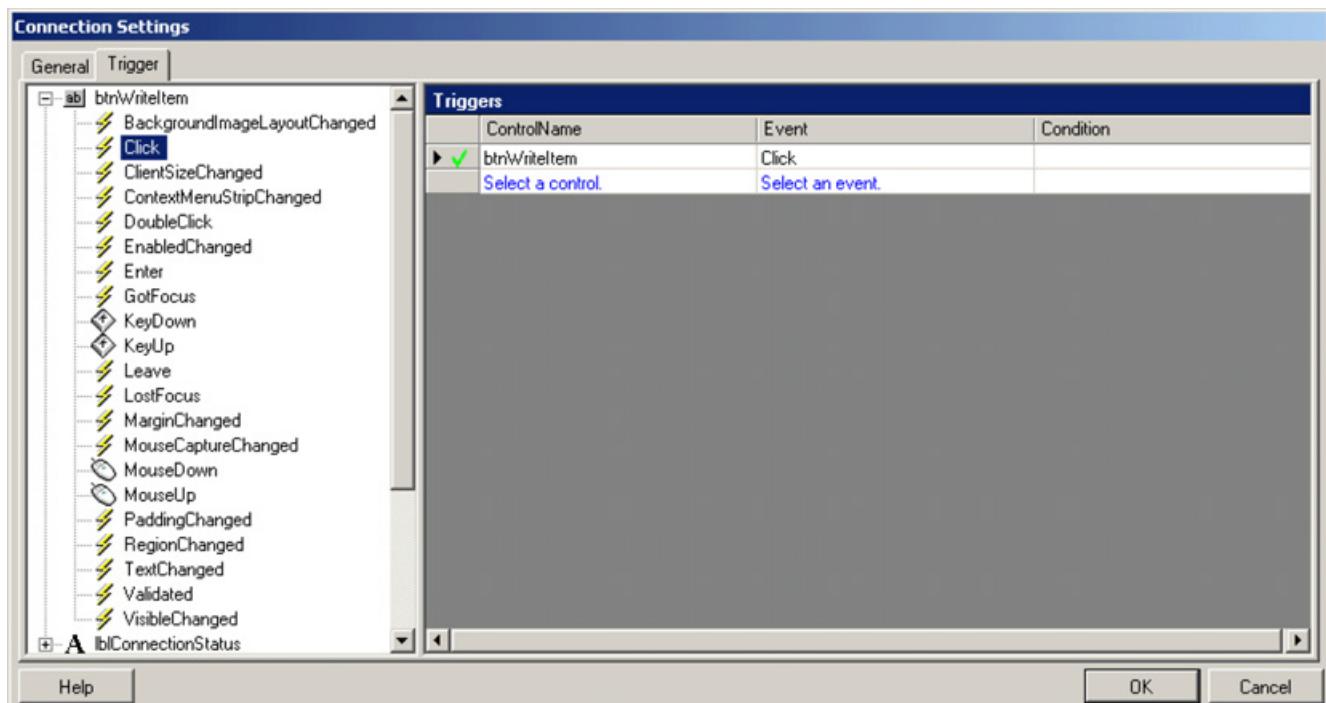
Konfigurieren Sie folgende spezielle Trigger-Ereignisse für die Datenverknüpfungen der Textboxen "txtDemoPower" bzw. "txtDemoTemperature":

ControlName	Event	Condition	Beschreibung
btnWriteltem1	Click	--	Der aktuelle Wert des Text Properties der Textbox txtDemoPower wird in das OPC-Item S7:[DEMO]DemoPower geschrieben, wenn der Button btnWriteltem1 angeklickt wird.
txtDemoTemperature	KeyDown	Return	Der aktuelle Wert des Text Properties der Textbox txtDemoTemperature wird in das OPC-Item S7:[DEMO]DemoTemperature geschrieben wenn der Key-Down Event den Wert "Return" enthält (also die Enter-Taste gedrückt wurde).

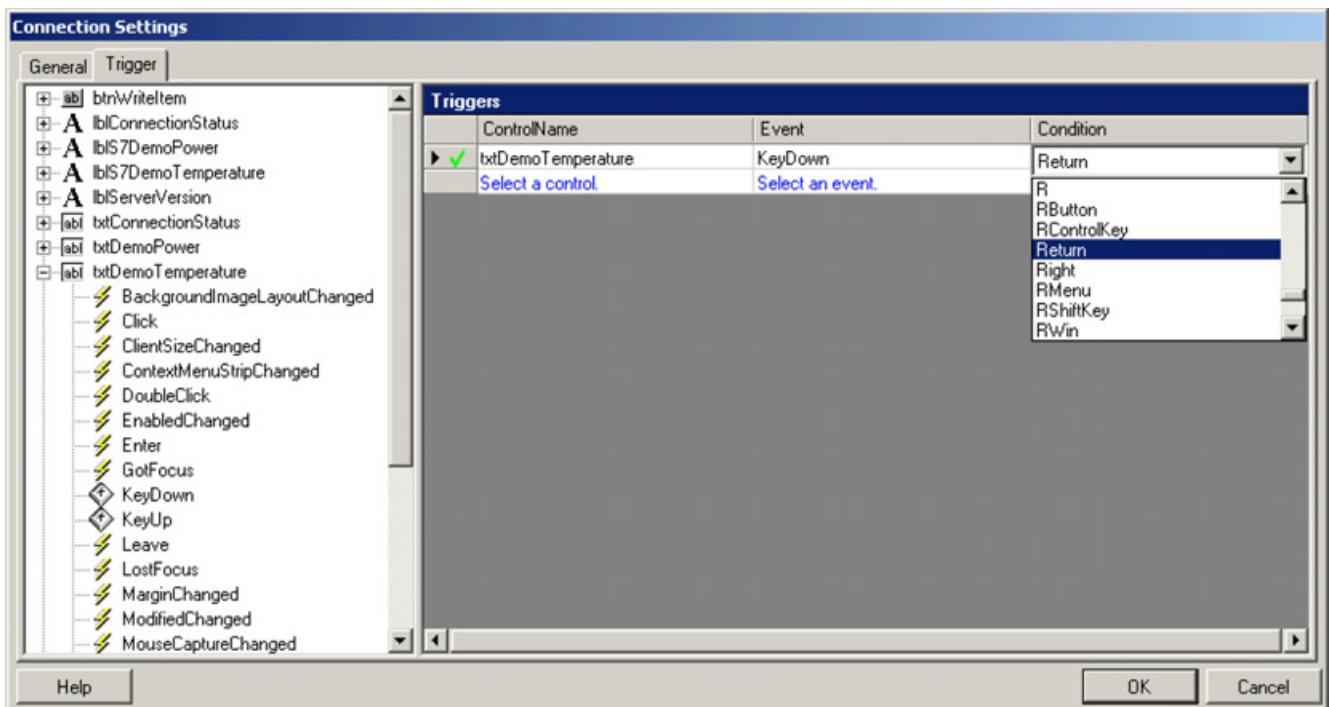
Diese beiden Beispiele demonstrieren das Verhalten des DataControls beim Schreiben auf OPC-Variablen.

### Beispiel 1:

Wenn die Textbox den Fokus erhält, also auch wenn der Cursor in der Textbox steht, wird die Textbox nicht mehr mit OPC-Werten aktualisiert (disable DataChange when Control has Focus). Damit wird verhindert, dass die Textbox plötzlich ihren Wert ändert, wenn der Anwender sie gerade editiert. Somit kann in aller Ruhe editiert werden und erst beim klicken des Buttons "btnWriteltem1", wird der Click-Event ausgelöst. Dieser löst dann den Schreibauftrag zum OPC-Server aus und der aktuelle Wert der Textbox wird auf das OPC-Item geschrieben.

**Beispiel 2:**

Wenn Sie den KeyDown-Event OHNE Bedingung (Condition) verwenden, dann würden beim Schreiben des Wertes "152" insgesamt 3 Schreibaufträge an den OPC-Server gesendet, bei jedem einzelnen Key-Down Event: Schreibe(1), Schreibe(15) und Schreibe(152). Wenn Sie aber zusätzlich die Bedingung (Condition) "Return" konfigurieren, können Sie die Textbox editieren, ohne dass Werte zum OPC-Server geschrieben werden. Erst wenn ein Key-Down Event der Return-Taste erkannt wird, wird der Wert in der Textbox zum OPC-Server geschrieben.

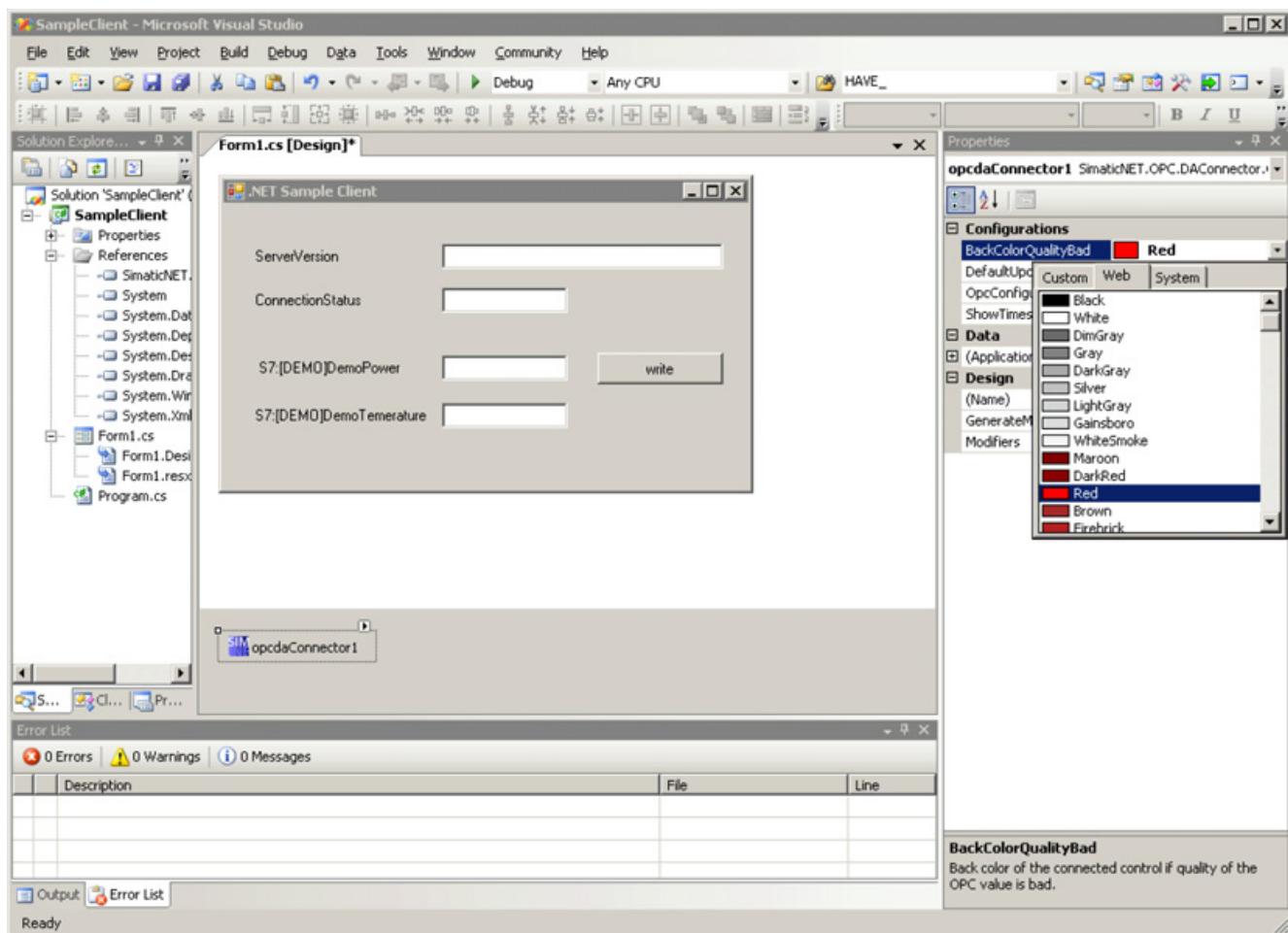


#### 5.1.4.5 OPC Quality und Error

Die Verknüpfung der OPC-Eigenschaften "Quality" und "Error", die ein jedes OPC-Item besitzt, wird durch das DataControl automatisch vorgenommen. Eine separate Verschaltung dieser OPC-Attribute ist mit dem SimaticNET .NET OPC Client Control nicht möglich.

## Quality

Die OPC-Quality eines gelesenen OPC-Items gibt an, ob der gelesene Wert vertrauenswürdig ist. Die Quality kann "good", "uncertain" oder "bad" sein und wird auf die Hintergrundfarbe (Eigenschaft BackColor) des verschalteten Windows-Controls abgebildet und auch im Tooltip angezeigt. Entsprechend wechselt die Hintergrundfarbe beispielsweise einer Textbox auf rot (default) wenn die Qualität des gelieferten OPC-Werts nicht "good" ist. Diese Verschaltung wird automatisch vorgenommen und im Dialog "Connector Configuration" wird lediglich der OPC-Value mit Windows Control Property verknüpft. Die generelle Einstellung der Farbe kann für das gesamte "OPCDAConnector"-Objekt geändert werden.

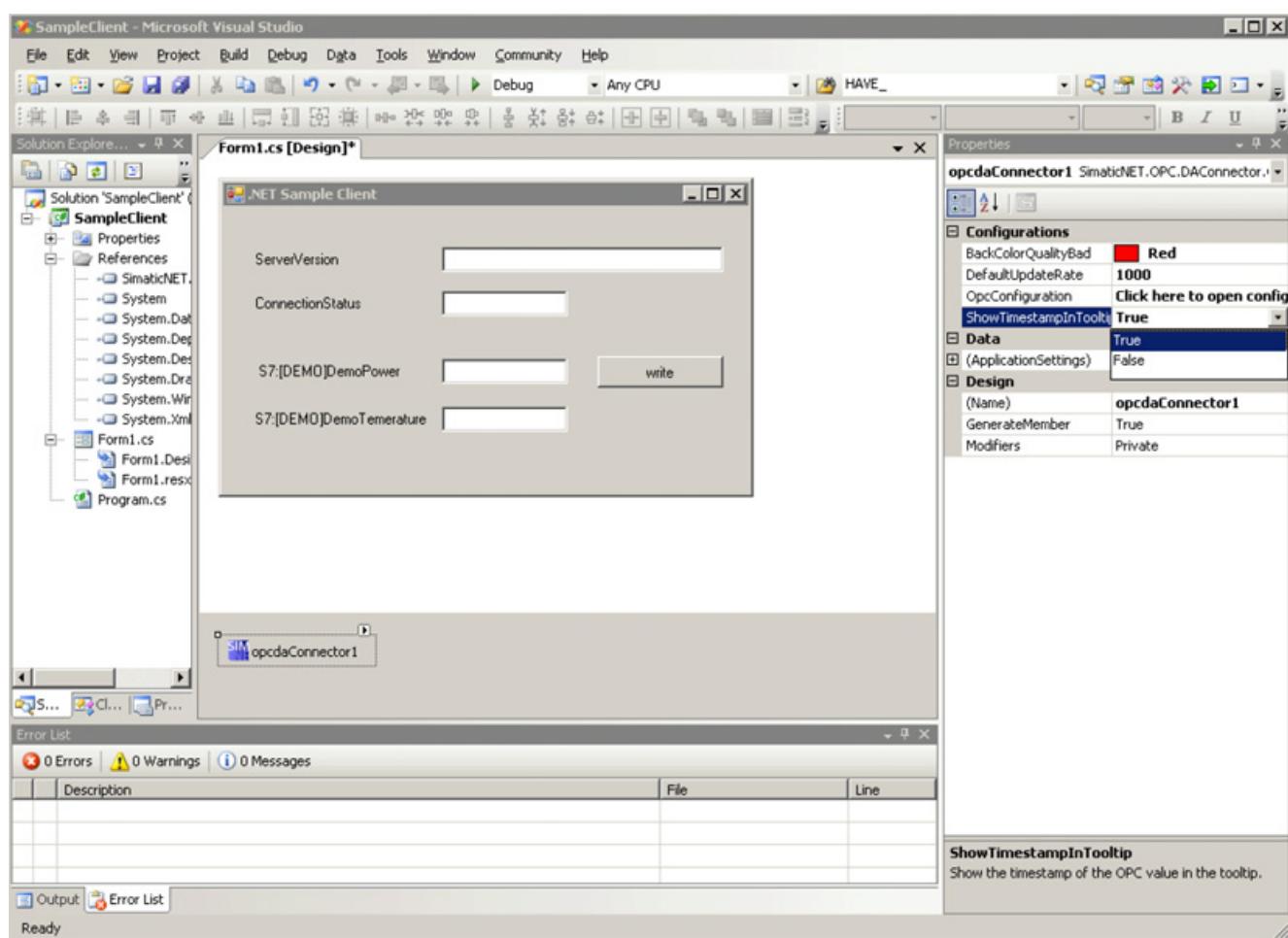


## Error

Bei Fehlern z.B. beim Schreiben, wird ein Fehler-Code geliefert. Der zugehörige fehlerbeschreibende Text wird im Tooltip des verschalteten Windows-Controls angezeigt. Beim überfahren mit der Maus können Sie sich somit den jeweils letzten Fehlertext anzeigen lassen. Auch diese Verschaltung wird automatisch vorgenommen und im Dialog "Connector Configuration" wird lediglich der OPC-Value mit dem Windows Control Property verknüpft, die Verschaltung des dazugehörigen Tooltips geschieht im Hintergrund. Diese Einstellungen sind fest vorgegeben und können nicht geändert werden. Einen Fehler können Sie in einer separaten Farbe anzeigen lassen.

## Timestamp

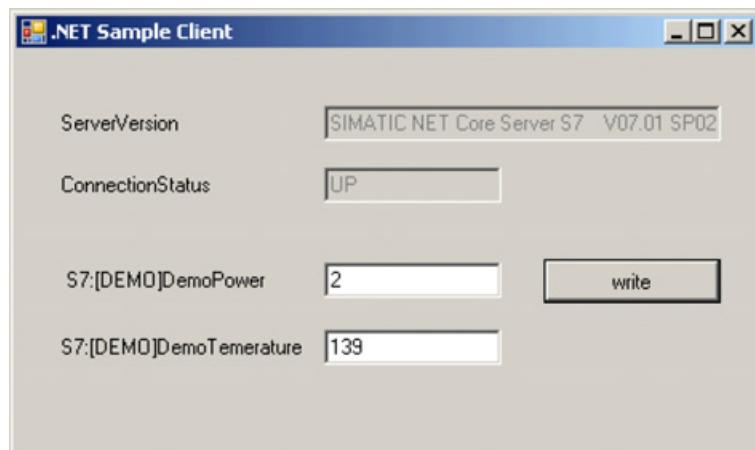
Der OPC-Zeitstempel (UTC Zeit) wird beim Lesen eines OPC-Items mitgeliefert. Dieses OPC-Attribut kann nicht separat mit der Eigenschaft eines Windows-Controls verschaltet werden. Die Verschaltung mit Tooltip des entsprechenden Windows-Objektes wird jedoch automatisch vorgenommen, wenn der "ShowTimeStampInTooltip" aktiviert ist.



## 5.1.5 Schritt 4 - Starten des Beispielprogramms

Die Beispielapplikation wird mit Doppelklick auf die "\*.exe"-Datei gestartet. Diese befindet sich im Unterordner "bin\Debug" bzw. "bin\Release" unterhalb des Visual Studio-Beispielprojekts, je nachdem in welcher Konfiguration kompiliert wurde.

Beachten Sie, dass zur Ausführung des Beispiels die S7-Demo Verbindung angeschaltet ist (siehe Kapitel "Einstellung eines Beispielprogramms (Seite 564)").



### 5.1.5.1 Debug und Release

Die Beispielapplikation kann "Debug" oder "Release" übersetzt werden. Die Assemblies der .NET Controls funktionieren für beide Varianten.

### 5.1.5.2 Applikation verteilen

Die neu erstellte OPC-Client-Applikation wird üblicherweise nicht auf dem Entwicklungsrechner ausgeführt sondern entweder auf einem SIMATIC-PC mit installierter "SIMATIC NET PC Software" (localer Client) oder auf einem weiteren Rechner, der weder MS Visual Studio noch SimaticNET-Installation besitzt (remoter OPC Client). In diesem Fall sollte sichergestellt werden, dass alle erforderlichen Dateien auf dem Zielsystem installiert sind.

#### Welche Dateien müssen mit ausgeliefert werden?

Um eine Applikation auf einen Rechner zu betreiben, werden neben der ausführbaren Datei ("\*.exe") und den .NET Assemblies zusätzlich die OPC-Proxies/Stubs benötigt sowie die VC 2010 SP1 Runtime. Die .NET Assemblies benötigen .NET Framework 4.0 oder höher. Die OPC Core Components (Proxy/Stub) müssen mit jedem OPC-Produkt (Client oder Server) installiert werden. Auf einem Rechner mit SimaticNET-Installation sind diese Dateien schon vorhanden. Alternativ können Mergemodul oder Redistributable-Setup bei der OPC Foundation heruntergeladen werden ([www.opcfoundation.org](http://www.opcfoundation.org))

#### Wo müssen die DLLs liegen bzw. hininstalliert werden?

Beim Übersetzen der Beispielapplikation werden die Assemblies automatisch in das bin-Verzeichnis des Beispielprojektes umkopiert, diese müssen mit der Client-Applikation installiert werden (im gleichen Verzeichnis wie die Applikation).

- Für die OPC Core Components gibt es ein Merge Module der OPC Foundation. Die Dateien werden nach "%systemroot%\windows\system32\" installiert
- Für die VC Runtime (2010 SP1) gibt es ein Merge Module von Microsoft.

#### **Welche Komponenten müssen gegebenenfalls in ein Setup eingebunden werden?**

Wenn ein Setup mit dem Visual Studio (Deployment Project) erstellt wird, erkennt das Studio automatisch die Abhängigkeiten und fügt die Assemblies in das Projekt mit ein. Zusätzlich müssen die Core Components der OPC Foundation eingebunden werden sowie die passende C-Runtime 2010 SP1.

---

#### **Hinweis**

Alternativ kann auch das OPC Scout V10-Client-Setup installiert werden. Darin sind bereits die OPC Proxy / Stub Redists und das .NET Framework enthalten.

---

## **5.1.6 Fehlerbehandlung**

### **Visual Studio LoaderLock Exception**

Wenn Sie die Beispielapplikation aus dem Debugger des Visual Studios heraus starten, kann eine Warnung (LoaderLock Exception) des Studios auftreten. Die Ursache für diese Warnung ist die gemischte Verwendung von "Managed -" und "Unmanaged Code". Diese Warnung kann ignoriert werden, denn das DataControl startet die gemischten "Assemblies" in der richtigen Reihenfolge und ein "Deadlock" kann daher nicht auftreten. Es wird dennoch empfohlen die Warnung abzuschalten. Um den Debugger zu stoppen, gehen folgendermaßen vor:

1. Wählen sie im Menü des Studios "Debug" > "Exceptions".
2. Öffnen Sie den Eintrag "Managed Debug Assistance".
3. Deaktivieren Sie die "LoaderLock" Checkbox.
4. Beenden Sie den Dialog mit "OK".

### **Es kann keine Verbindung zum remoten OPC-Server hergestellt werden:**

Prüfen Sie die DCOM-Einstellungen und gehen Sie anhand der Benutzerdokumentation des Servers vor. Beachten Sie hierbei dass die Applikation im Kontext des Visual Studios (DevEnv.exe) läuft wenn sie die Applikation aus der Entwicklungsumgebung heraus starten. Entsprechend müssen auch DCOM-Rechte für den Kontext, in dem dieser Prozess läuft, richtig eingestellt sein.

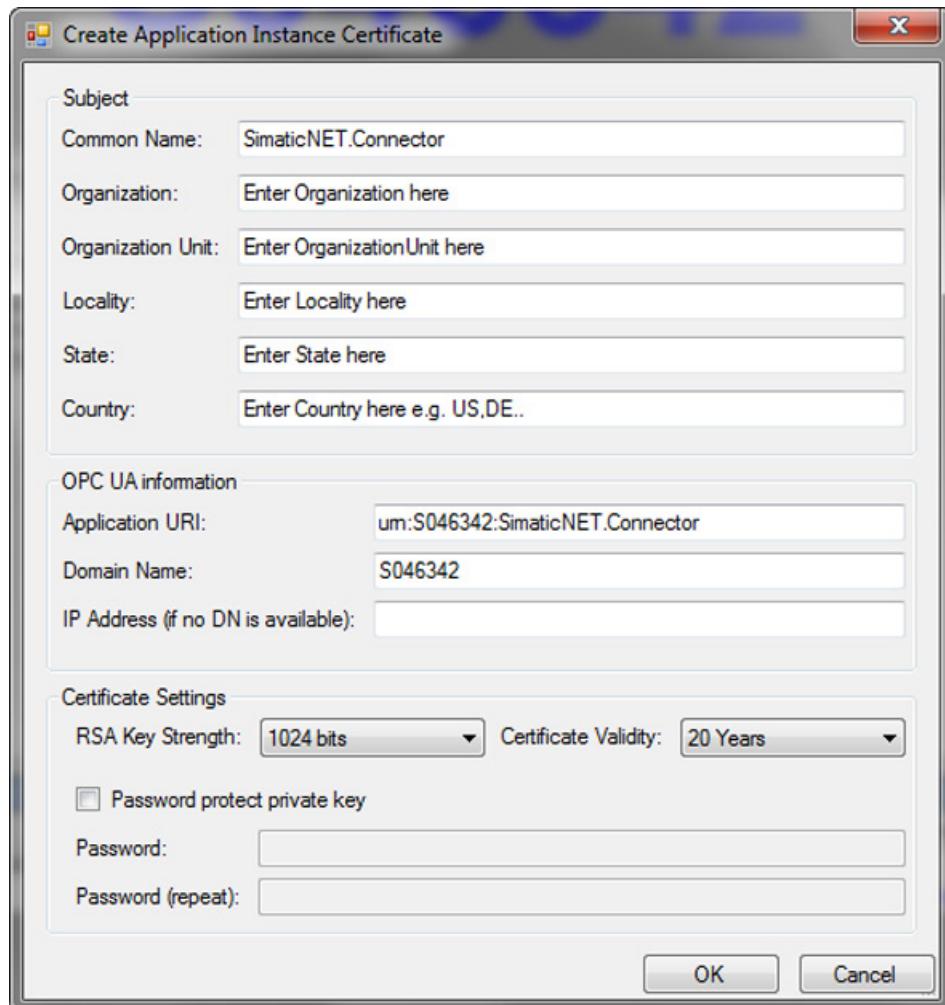
### **Es werden keine Datenänderungen vom OPC-Server gemeldet**

Die Applikation muss ihre Sicherheitseinstellungen so setzen, dass die vom OPC-Server gesendeten Nachrichten wie "Shutdown-Event" oder "DataChange-Notification", empfangen

werden dürfen. Hierzu muss die Client-Applikation den richtigen Security Level einstellen. Dies geschieht in der Initialisierungsphase der Applikation. Es muss der CoInitializeSecurity Aufruf mit den entsprechenden Einstellung (AuthenticationLevel = RPC\_C\_AUTHN\_LEVEL\_CONNECT und ImpersonationLevel = RPC\_C\_IMP\_LEVEL\_IDENTIFY) vor dem Start der Applikation ausgeführt werden (lesen Sie hierzu die MSDN). Falls die COM Security nicht von der Applikation gesetzt wird, werden die Standardeinstellungen verwendet (Default). Die DCOM Security muss dann entsprechend eingestellt werden ("Start" > "Ausführen" > "dcomcnfg.exe").

### 5.1.7 OPC-UA-Zertifikat-Erstellung für das NET OPC Data Control

Über einen Abfrage-Dialog werden für ".NET OPC Data Control self-signed X509 OPC UA Zertifikate" im Windows-Zertifikatspeicher erzeugt. Die Erstellung erfolgt, wenn noch kein passendes Zertifikat vorhanden ist.



Die Zertifikatverwaltung im Windows-Zertifikatspeicher entspricht der Zertifikatverwaltung im UA-S7-Client OPC Scout V10 (siehe Kapitel "Zertifikatsverwaltung im OPC-UA-Client "OPC Scout V10" (Seite 513)").

## 5.1.8

### OPC-UA-Zertifikate bei Verbindung zu einem OPC-UA-Server

Der Verbindungsauflauf über das .NET OPC Data Control zu einem Unified Architecture (UA) Server kann in der Regel über verschiedene Endpunkte stattfinden. Die SIMATIC NET OPC-UA-Server können so konfiguriert werden, dass ein Endpunkt unverschlüsselte Verbindungen zulässt und ein Endpunkt ausschließlich verschlüsselte Verbindungen akzeptiert.

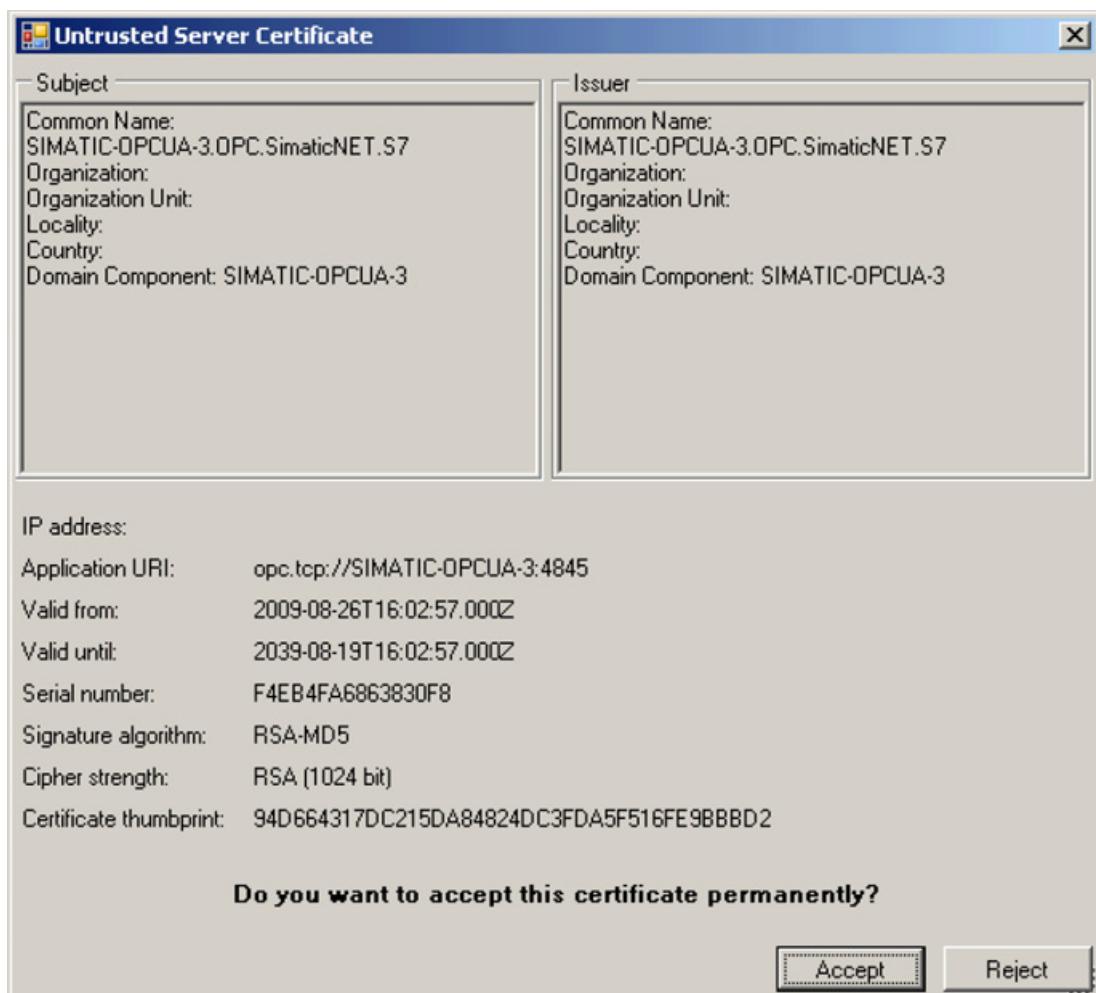
Beim Verbinden zu einem Endpunkt mit Security, d.h. die "SecurityPolicy" des Endpunktes und der "MessageSecurityMode" sind nicht "None", müssen zwischen Client und Server Zertifikate ausgetauscht werden.

Der Verbindungsauflauf erfordert dann folgende Schritte:

- Der Client prüft, ob er schon ein Zertifikat besitzt. Wenn er keines besitzt, wird der Benutzer aufgefordert eines zu erzeugen.
- Der Client lädt das Zertifikat des Servers.

Wenn das Serverzertifikat dem .NET OPC Data Control UA Client noch nicht bekannt bzw. als vertrauenswürdig eingestuft ist, wird der Benutzer gefragt, ob er das Zertifikat akzeptieren möchte.

Die Zertifikate liegen unter "My" Store und haben eine Gültigkeit von 20 Jahren.



Wenn ein Client versucht sich mit einem Server zu verbinden und diesem das Clientzertifikat nicht bekannt ist bzw. als nicht vertrauenswürdig eingestuft wird, verweigert der Server die Verbindung.

Wie man bei den SIMATIC NET OPC-UA-Servern Zertifikate akzeptieren, ablehnen und verwalten kann, können Sie im Kapitel "Zertifikatsverwaltung für den OPC-UA-Server (Seite 511)" und "Zertifikatsverwaltung mit grafischer Oberfläche (Seite 518)" nachlesen.

## 5.2 .NET OPC Client API

### Zielsetzung

Zielsetzung der SIMATIC NET .NET OPC Client API ist es Benutzern der Sprachen C-Sharp und Visual Basic .NET eine einfach und intuitiv zu benutzende und optimierte Klassenbibliothek zur Verfügung zu stellen, mit der sich schnell OPC-Clientapplikationen für den Zugriff auf OPC-Server der SIMATIC NET Produktreihe erstellen lassen.

## Merkmale der .NET OPC Client-Komponente

Die folgenden Merkmale zeichnen die .NET Klassenbibliothek und die zugrunde liegende C++ Bibliothek aus:

- Einfache, intuitive .NET-Schnittstelle (API – Application Programming Interface).
- Reduzierung der OPC Data Access-Schnittstellen auf wesentliche Funktionen.
- Der Benutzer benötigt kein detailliertes Wissen über die verschiedenen OPC Data Access-Schnittstellen.
- Die Komponente verdeckt komplett die verschiedenen Basistechnologien von OPC wie COM, DCOM, Webservices, SOAP und XML.
- Die Komponente verdeckt komplett das Verbindungshandling zu einem OPC-Server mit Verbindungsauflaufbau, Verbindungsüberwachung und erneutem Verbindungsauflaufbau im Fehlerfall.
- Die Entwicklung von OPC-Clientapplikation unter C-Sharp .NET und Visual Basic .NET ist mit der SIMATIC NET .NET OPC Client API einfach möglich.
- Umsetzung der OPC-Daten aus verschiedenen OPC Data Access-Schnittstellen auf .NET Datentypen.
- Schnelles und einfaches Ermitteln von OPC COM-Servern lokal und im Netzwerk.
- Performante und optimierte Client-Server-Kommunikation durch Implementierung der Kernfunktionalität in C++.
- Unterstützung von OPC UA und der Zertifikatsverwaltung für OPC UA.

---

### Hinweis

Mit dieser API können ausschließlich Verbindungen zu OPC-Servern der SIMATIC NET Produktfamilie aufgebaut werden. Verbindungen zu anderen OPC-Servern oder zu OPC-Servern von anderen Herstellern sind nicht möglich.

---

---

### Hinweis

Bei der Verbindung zu einem OPC-UA-Server kann es je nach gewähltem Endpunkt erforderlich sein, dass zwischen Client und Server Zertifikate ausgetauscht werden, bevor eine Verbindung möglich ist. Benutzen sie dazu die entsprechenden Klassen und Methoden.

---

## **5.2.1 Namensraum SIMATICNET.OPCDACLIENT**

Die folgende Funktionalität stellt der Namensraum SimaticNet.OpcDaClient der .NET-Komponente über das Objekt "DaServerMgt" zur Verfügung:

- Verbindung zum OPC-Server

Über die Methode "Connect" kann die Verbindung zum OPC-Server aufgebaut werden und über die Methode "Disconnect" wird die Verbindung wieder abgebaut. Die Verbindung wird von der .NET-Komponente überwacht. Treten Verbindungsfehler auf, werden Statusänderungen über den "Event ServerStateChanged" gemeldet.

- Lesen und Schreiben von OPC-Data-Access-Items

Über die Methoden "Read" und "Write" können die Werte von OPC-Items synchron und asynchron gelesen und geschrieben werden.

- Melden von Datenänderungen

Die .NET-Komponente bietet einen Mechanismus über den Werteänderungen gemeldet werden und damit kein zyklisches Lesen notwendig ist. Mit der Methode "Subscribe" können Items für die Überwachung angemeldet werden. Mit "SubscriptionCancel" werden diese wieder abgemeldet. Geänderte Werte werden über den "Event DataChanged" gemeldet.

- Informationen über den Adressraum ermitteln

Die Methode "Browse" ermöglicht es, den Adressraum eines OPC-Data-Access-Servers nach OPC-Items zu durchsuchen. Über die Methode "GetProperties" können die Eigenschaften von OPC-Items ermittelt werden.

### **5.2.1.1 Klassen des Datenmodells**

Im folgenden Kapitel werden die von der .NET API zur Verfügung gestellten Klassen beschrieben. Bis auf die Klasse "DaServerMgt" besitzen diese ausschließlich Properties, werden also zur Datenkapselung verwendet. "DaServerMgt" stellt die API für den Zugriff auf einen OPC-Data-Access-Server zur Verfügung.

#### **Klasse DaServerMgt**

Die Klasse "DaServerMgt" ermöglicht den Zugriff auf einen OPC-Data-Access-Server. Eine ausführliche Beschreibung der API und ihrer Methoden ist unter "Die Schnittstelle des Objektes DaServerMgt" zu finden.

#### **Enumerator ServerState**

Der "Enumerator ServerState" wird beim "StateChange Event" übergeben und gibt Auskunft über den aktuellen Status des Servers.

- CONNECTED  
Die Verbindung zum OPC-Server ist aufgebaut.
- DISCONNECTED  
Es ist keine Verbindung zum OPC-Server aufgebaut.

- **ERRORSHUTDOWN**  
Der OPC-Server hat einen Shutdown Event geschickt. Die Verbindung ist unterbrochen. Ein neuer Verbindungsaufbau wird versucht.
- **ERRORWATCHDOG**  
Die .NET API hat einen Verbindungsfehler zum OPC-Server festgestellt. Die Verbindung ist unterbrochen. Ein neuer Verbindungsaufbau wird versucht.
- **UNDEFINED**  
Der OPC-Server ist in einem Status, der keinem der aufgeführten Stati entspricht.

## Klasse ItemIdentifier

Die Klasse "ItemIdentifier" wird bei den Methoden "Read", "Write", "GetProperties" und "Subscribe" benötigt. Die Klasse dient zur Identifikation eines OPC-Items. Instanzen der Klasse werden bei allen Methoden als ref-Parameter(in/out) übergeben. Dies ist notwendig, da die .NET-Komponente bei der ersten Verwendung Informationen in dem Objekt speichert, die bei wiederholter Verwendung zur Optimierung der OPC-Aufrufe verwendet werden. Aus diesem Grund sollten die Objekte auch in Read- und Write-Aufrufen wieder verwendet werden und nicht für jeden Aufruf neu initialisiert werden.

Die Klasse enthält auch Fehlerinformationen, die nach einem Aufruf überprüft werden müssen, um Itemfehler erkennen zu können.

Eigenschaften der Klasse ItemIdentifier:

- **string ItemName**  
Diese Eigenschaft enthält den Itemnamen (ItemID) eines OPC-Data-Access-Items.
- **string ItemPath**  
Im Falle eines OPC-XML-DA-Server kann hier der optionale ItemPath angegeben werden. Im Falle eines COM-OPC-Data-Access-Server wird diese Eigenschaft ignoriert.
- **object ClientHandle**  
Wird ein ItemIdentifier für die Methode "Subscribe" erzeugt, wird das übergebene "Clienthandle" mit dem geänderten Wert des Items beim "DataChanged" wieder an den Client übergeben. Da das "Clienthandle" vom Typ object ist, kann hier jeder beliebige .NET-Typ übergeben werden, z.B. ein TextBox Control oder ein Objekt, das Informationen über die weitere Verarbeitung der Daten enthält. In der "DataChange Handlermethode" kann so über das "ClientHandle" das zugehörige Objekt in der Applikation identifiziert werden.
- **int ServerHandle**  
Instanzen der Klasse "ItemIdentifier" werden bei allen verwendeten Methoden als Referenz Parameter (Keyword ref) übergeben. Bei den Methodenaufrufen "Read" und "Write" wird die Eigenschaft "int ServerHandle" von der .NET API gesetzt. Werden dieselben ItemIdentifier-Instanzen ein weiteres Mal an "Read" oder "Write" übergeben, kann die .NET API die Aufrufe zum OPC-Server stark optimieren.
- **System.Type DataType**  
Hier kann der Datentyp angegeben werden, in dem der OPC-Server den Wert des Items beim "Read" oder "DataChange" liefern soll. Wird die Eigenschaft nicht gesetzt, wird der native Datentyp der Items im OPC-Server geliefert. In diesem Fall setzt die .NET API den Datentyp bei der ersten Verwendung der ItemIdentifier-Instanz.
- **ResultID ResultID**  
Ist bei einem OPC-Aufruf ein Item-Fehler aufgetreten (z.B. Unbekannter ItemName,

Versuch des Schreibens eines readOnly-Items, usw.) wird der zugehörige Fehlercode in diesem Objekt des zugehörigen "ItemIdentifier" gespeichert. Die Klasse "ResultID" stellt den Fehlercode(int), den Namen(string) und eine lokalisierte Beschreibung(string) zur Verfügung. Somit kann im Programm auf eventuell auftretende Fehler reagiert werden.

## Klasse ItemValue

Die Klasse "ItemValue" wird bei den Methoden "Read" und "Write" verwendet. Beim Lesen enthält "ItemValue" Wert, Qualität und Zeitstempel. Beim Schreiben enthält "ItemValue" nur den Wert des OPC-Items.

Beim Aufruf der Methode "Read" ist das "ItemValue Array" ein out-Parameter, das heißt das Objekt wird komplett von der .NET API erzeugt.

Beim Aufruf der Methode "Write" muss das "ItemValue Array" entsprechend dem "ItemIdentifier Array" vom Client erzeugt und mit den zu schreibenden Werten gefüllt werden.

Eigenschaften der Klasse "ItemValue":

### **object Value**

Der Wert, der gelesen wurde bzw. geschrieben werden soll. Da die Eigenschaft "Value" vom Typ "object" ist, kann sie jeden beliebigen Datentyp aufnehmen oder enthalten. Im Normalfall ist "Value" vom selben Typ, der über den korrespondierenden "ItemIdentifier" angefordert wurde.

### **QualityID Quality**

Die Qualität des Wertes. Die Klasse "QualityID" stellt sowohl den Qualitätscode(int), als auch den Namen(string) und eine Beschreibung(string) zur Verfügung.

### **System.DateTime TimeStamp**

Der Zeitstempel des Wertes.

## Klasse ItemValueCallback

Die Klasse "ItemValueCallback" leitet sich von der Klasse "ItemValue" ab und besitzt zusätzlich zu den unter "ItemValue" beschriebenen Eigenschaften noch die folgenden:

- **object ClientHandle**

Hier wird das bei der Methode "Subscribe" oder "ReadAsync" übergebene "ClientHandle" mitgeliefert. "ClientHandle" dient dem Client zur eindeutigen Zuordnung des gelieferten Wertes.

- **ResultID ResultID**

Hier wird ein etwaiger Fehler bezüglich des über "ClientHandle" zu identifizieren Items mitgeteilt. Die Klasse "ResultID" stellt den Fehlercode(int), den Namen(string) und eine lokalisierte Beschreibung(string) zur Verfügung. Somit kann im Programm auf eventuell auftretende Fehler reagiert werden.

## Klasse ItemResultCallback

Die Klasse "ItemResultCallback" wird im "WriteCompleted Callback" verwendet und hat die folgenden Eigenschaften:

- **object ClientHandle**  
Hier wird das bei der Methode "WriteAsync" übergebene "ClientHandle" mitgeliefert. "ClientHandle" dient dem Client zur eindeutigen Zuordnung des gelieferten Wertes.
- **ResultID ResultID**  
Hier wird ein etwaiger Fehler bezüglich des über "ClientHandle" zu identifizieren Items mitgeteilt. Die Klasse "ResultID" stellt den Fehlercode(int), den Namen(string) und eine lokalisierte Beschreibung(string) zur Verfügung. Somit kann im Programm auf eventuell auftretende Fehler reagiert werden.

## Klasse BrowseElement

Die Klasse "BrowseElement" enthält alle Daten bezüglich eines durch die Methode "Browse" ermittelten OPC-Data-Access-Items.

Eigenschaften der Klasse "BrowseElement":

**string Name**

Der Name des zurückgegebenen Elements. Normalerweise wird dieser Name für die Baumarstellung des Namensraums eines OPC-Servers benutzt.

**string ItemName**

Der ItemName des Elements.

**string ItemPath**

Der ItemPath des Elements.

**bool HasChildren**

Zeigt an, ob das Element Kinderelemente im Namensraum besitzt.

**bool IsItem**

Zeigt an, ob das Element ein OPC-Data-Access-Item ist.

**ItemProperties ItemProperties**

Die über die Methode "Browse" angeforderten Eigenschaften des Elements.

## Enumerator BrowseFilter

Durch die Übergabe eines "BrowseFilter" wird beim Aufruf der Methode "Browse" angegeben, von welchem Typ die gelieferten Kinderelemente des angegebenen Elementes sein sollen. Mögliche Filter sind hierbei:

- ALL  
Alle Elemente werden zurückgegeben.
- BRANCH  
Nur Elemente vom Typ Branch werden zurückgegeben.
- ITEM  
Nur Elemente vom Typ Item werden zurückgegeben.

## Klasse ItemProperties

Die Klasse "ItemProperties" wird ausschließlich von der .NET API erzeugt. Sie enthält die Eigenschaften eines OPC-Items.

Eigenschaften der Klasse "ItemProperties":

- **ItemProperty[ ] RequestedItemProperties**

Ein Array von Objekten der Klasse "ItemProperty". In diesem Array befinden sich alle bei "GetProperties" oder "Browse" angeforderten Eigenschaften eines OPC-Items.

## Klasse ItemProperty

Die Klasse "ItemProperty" repräsentiert eine Eigenschaft eines OPC-Items.

Eigenschaften der Klasse "ItemProperty":

- **string ItemName**

Wenn ein OPC-Server das Lesen und Schreiben eines "Property" über ein Item unterstützt, wird hier der "ItemName" der "Property" zurückgegeben.

- **string ItemPath**

Wenn ein OPC-Server das Lesen und Schreiben eines "Property" über ein Item unterstützt, wird hier der "ItemPath" der "Property" zurückgegeben.

- **string Description**

Die Beschreibung des "Property". Diese Information kann zur Darstellung eines "Property" in einer grafischen Oberfläche benutzt werden.

- **object Value**

Der Wert des "Property".

- **ResultID ResultID**

Sollte ein Fehler beim Ermitteln des "Property" auftreten, wird der zugehörige Fehlercode in dieser Eigenschaft gespeichert.

- **System.Type DataType**

Der Datentyp des Wertes des "Property".

- **int PropertyID**

Die ID des "Property".

## Klasse ResultID

Die Klasse "ResultID" enthält einen Fehlercode, dessen String-Repräsentation und eine lokalisierte Beschreibung des aufgetretenen Fehlers. "ResultIDs" werden benutzt um Fehler auf Itemebene anzuzeigen und sind Teil der OPCExceptions.

Die Eigenschaften der Klasse "ResultID":

- **int Code**

Der Code der bei einer Aktion vom Server übermittelt wurde.

- **string Name**

Die String-Repräsentation des Codes.

- **string Description**  
Die lokalisierte Beschreibung des aufgetretenen Fehlers.
- **bool Succeeded**  
Über diese Eigenschaft kann ermittelt werden, ob "ResultID" den Code einer erfolgreichen Operation enthält, ohne den Code selbst genau auswerten zu müssen.

Eine Instanz der Klasse "ResultID" kann folgende Werte annehmen:

Value	Code	Name	Beschreibung
WIN_S_OK	0x00000000	S_OK	Operation succeeded.
WIN_S_FALSE	0x00000001	S_FALSE	The function was partially successful.
S_UNSUPPORTEDRATE	0x0004000D	OPC_S_UNSUPPORTEDRATE	The server does not support the requested data rate but will use the closest available rate.
S_INUSE	0x0004000F	OPC_S_INUSE	The operation cannot be performed because the object is being referenced.
S_DATAQUEUEOVERFLOW	0x00040404	OPC_S_DATAQUEUEOVERFLOW	Not every detected change has been returned since the server's buffer reached its limit and had to purge out the oldest data.
S_CLAMP	0x0004000E	OPC_S_CLAMP	A value passed to write was accepted but the output was clamped.
RPC_S_SERVER_UNAVAILABLE	0x800706BA	RPC_S_SERVER_UNAVAILABLE	The RPC server is unavailable.
RPC_S_CALL_FAILED	0x800706BE	RPC_S_CALL_FAILED	The remote procedure call failed.
E_UNKNOWNPATH	0xC004000A	OPC_E_UNKNOWNPATH	The item's access path is not known to the server.
E_UNKNOWNITEMID	0xC0040007	OPC_E_UNKNOWNITEMID	The item ID is not defined in the server address space or no longer exists in the server address space.
E_RATENOTSET	0xC0040405	OPC_E_RATENOTSET	There is no sampling rate set for the specified item.
E_RANGE	0xC004000B	OPC_E_RANGE	The value was out of range.
E_PUBLIC	0xC0040005	OPC_E_PUBLIC	The requested operation cannot be done on a public group.
E_NOTSUPPORTED	0xC0040406	OPC_E_NOTSUPPORTED	The server does not support writing of quality and/or timestamp.
E_NOTFOUND	0xC0040011	OPC_E_NOTFOUND	The requested object (e.g. a public group) was not found.

Value	Code	Name	Beschreibung
E_NOBUFFERING	0xC0040402	OPC_E_NOBUFFERING	The server does not support buffering of data items that are collected at a faster rate than the group update rate.
E_INVALIDITEMID	0xC0040008	OPC_E_INVALIDITEMID	The item ID does not conform to the server's syntax.
E_INVALIDHANDLE	0xC0040001	OPC_E_INVALIDHANDLE	The value of the handle is invalid.
E_INVALIDFILTER	0xC0040009	OPC_E_INVALIDFILTER	The filter string was not valid.
E_INVALIDCONTINUATIONPOINT	0xC0040403	OPC_E_INVALIDCONTINUATIONPOINT	The continuation point is not valid.
E_INVALIDCONFIGFILE	0xC0040010	OPC_E_INVALIDCONFIGFILE	The server's configuration file is an invalid format.
E_INVALIDARG	0x80070057	E_INVALIDARG	An argument to the function was invalid.
E_INVALID_PID	0xC0040203	OPC_E_INVALID_PID	The specified property ID is not valid for the item.
E_FAIL	0x80004005	E_FAIL	Unspecified Error.
E_DUPLICATENAME	0xC004000C	OPC_E_DUPLICATENAME	Duplicate name not allowed.
E_DEADBANDNOTSUPPORTED	0xC0040401	OPC_E_DEADBANDNOTSUPPORTED	The item does not support deadband.
E_DEADBANDNOTSET	0xC0040400	OPC_E_DEADBANDNOTSET	The item deadband has not been set for this item.
E_BADTYPE	0xC0040004	OPC_E_BADTYPE	The server cannot convert the data between the specified format and/or requested data type and the canonical data type.
E_BADRIGHTS	0xC0040006	OPC_E_BADRIGHTS	The item's access rights do not allow the operation.
DISP_E_TYPEMISMATCH	0x80020005	DISP_E_TYPEMISMATCH	Type mismatch.
CONNECT_E_NOCONNECTION	0x80040200	CONNECT_E_NOCONNECTION	The client has not registered a callback through IConnectionPoint::Advise.
CONNECT_E_ADVISELIMIT	0x80040201	CONNECT_E_ADVISELIMIT	Advise limit exceeded for this object.

### Klasse QualityID

Die Klasse "QualityID" enthält alle Informationen bezüglich eines vom Server übermittelten Qualitäts-Code.

Die Eigenschaften der Klasse "QualityID":

- **int FullCode**  
Der vom Server übermittelte Code.
- **int Quality**  
Der Code, der die Qualität des übermittelten Wertes beschreibt
- **int LimitBits**  
Der im vom Server übermittelten Code enthaltene Limit-Anteil
- **int VendorBits**  
Der im vom Server übermittelten Code enthaltenen Hersteller-Anteil
- **bool IsGood**  
Über diese "Property" kann ermittelt werden, ob der gelesene Wert von guter Qualität ist.
- **string Name**  
Die String-Repräsentation des Codes.
- **string Description**  
Die lokalisierte Beschreibung des Qualitäts-Code.

## Klasse ConnectInfo

Die Klasse "ConnectInfo" enthält Initialisierungsdaten, die vom Serverobjekt beim Aufruf der "Connect-Methode" ausgewertet werden.

Die Eigenschaften der Klasse "ConnectInfo":

- **bool RetryAfterConnectionError**  
Wird dieses Flag gesetzt, wird die OPC-Client-API bei einem Verbindungsabbruch solange einen erneuten Verbindungsaufbau versuchen, bis sie erfolgreich war. Konnte die Verbindung wiederhergestellt werden, sind die vor der Unterbrechung erzeugten Gruppen-Handles weiterhin gültig. Auch die "Eventhandler-Methoden" sind weiterhin an den Events angemeldet.
- **bool RetryInitialConnect**  
Wird dieses Flag gesetzt, wird die OPC-Client API auch dann weiterhin versuchen sich zum Server zu verbinden, wenn der erste Verbindungsversuch nicht erfolgreich war.
- **int KeepAliveTime**  
Die OPC-Client-API prüft zur Laufzeit konsequent die Verfügbarkeit und die Verbindung des Servers. KeepAliveTime repräsentiert das Zeitintervall in Millisekunden, nach dem eine Validierung stattfindet.  
Der Initialwert von "KeepAliveTime" ist 10000 ms.  
Das Intervall für den Reconnect zum OPC-Server beginnt mit zweimal "KeepAliveTime" und wird bis zu 10-mal "KeepAliveTime" erhöht wenn der OPC-Server längere Zeit nicht verfügbar ist. Das Intervall für einen Reconnect nach einem OPC-Server-Shutdown beträgt eine Minute.
- **string LocalID**  
Über LocalID kann ein Länderkürzel ("us", "en", usw.) an den Server übergeben werden. Das Setzen einer LocalID bewirkt, dass lokalisierbare Rückgabewerte sprachspezifisch übergeben werden. Liegt der Wert in keiner lokalisierbaren Version vor wird der Defaultwert übergeben. Die folgende Tabelle zeigt beispielhaft einige LocalIDs:

Locale	LocalID
System default	Empty string
English	en
English - United States	en - us
English - United Kingdom	en - gb
German	de
German - Germany	de - de
German - Austria	de - at

---

### Hinweis

Die Änderung des Verbindungszustands sollte grundsätzlich über das Event "ServerStateChanged" mitverfolgt werden. Erst wenn der Status des Servers auf Connected wechselt, können weiter Methoden wie z. B. "Browse", "Read" oder "Subscribe" erfolgreich ausgeführt werden.

---

- **string SecurityPolicyUri**  
Die SecurityPolicy definiert, welche Art der Sicherheit für die Übertragung über den UA SecureChannel benutzt wird z. B. unverschlüsselt oder Basic128Rsa verschlüsselt.
- **byte MessageSecurityMode**  
Der MessageSecurityMode definiert, welche Sicherheit auf Nachrichtenebene verwendet wird z. B. 1 = None, 2 = Encrypt, 3 = Encrypt&Sign.
- **byte[] ServerCertificate**  
Das X509 Zertifikat des OPC-UA-Servers zu dem die Verbindung aufgebaut werden soll. Das Zertifikat kann z. B. über "OpcServerEnum::getCertificateForEndpoint" geladen werden.
- **byte[] ClientCertificate**  
Das X509-Zertifikat des Client.
- **byte[] ClientPrivateKey**  
Der Private-Schlüssel für das X509-Zertifikat des Client.
- **string CertificateStoreName**  
Der Name des "WindowsCertificateStore" in dem die Zertifikate für Client und Server gespeichert werden. Verwenden Sie "UA Applications".
- **WinStoreLocation CertificateStoreLocation**  
Die Location im "WindowsCertificateStore" z. B. LocalMachine

---

### Hinweis

Die Zertifikate und die Einstellungen für den "WindowsCertificateStore" werden nur benötigt, wenn die Verbindung zum OPC-UA-Server gesichert ist d.h. "SecurityPolicy" und "MessageSecurityMode" sind nicht auf "None" eingestellt.

Um die Location "LocalMachine" nutzen zu können, werden Administratorrechte benötigt.

---

## Enumerator ReturnCode

Durch die Rückgabe eines "ReturnCode" wird beim Aufruf verschiedener Methoden angezeigt, ob die betreffende Funktion erfolgreich war bzw. ob die Qualität der Werte nicht gut war:

- SUCCEEDED  
Die Funktion wurde erfolgreich abgeschlossen.
- ITEMERROR  
Bei mindestens einem Item ist ein Fehler während der Operation aufgetreten. Um welches Item es sich handelt und wie der exakte Fehlercode lautet, muss mit Hilfe der Funktionsparameter (z.B. ref ItemIdentifier[]) herausgefunden werden.
- QUALITYNOTGOOD  
Bei mindestens einem Item ist die Qualität nicht gut. Um welches Item es sich handelt und wie der exakte Qualitäts-Code lautet, muss mit Hilfe der Funktionsparameter (ref ItemIdentifier[]) herausgefunden werden.
- ITEMERRORANDQUALITYBAD  
Bei mindestens einem Item ist ein Fehler während der Operation aufgetreten und bei mindestens einem Item (demselben oder einem anderen) ist die Qualität nicht gut. Um welche Items es sich handelt, muss mit Hilfe der Funktionsparameter (ref ItemIdentifier[]) herausgefunden werden.
- UNSUPPORTEDUPDATERATE  
Die angeforderte Aktualisierungsrate wird vom Server nicht unterstützt.

### 5.2.1.2 Die Schnittstelle des Objektes DaServerMgt

#### Erzeugen des DaServerMgt-Objektes

Wie für .NET-Typen üblich, muss zuerst eine Instanz der Klasse "DaServerMgt" erzeugt werden:

```
DaServerMgt daServerMgt = new DaServerMgt();
```

#### Methode Connect

Die Methode "Connect" stellt die Verbindung zum Server her.

Das Property "IsConnected" gibt den Verbindungsstatus des Client zur .NET API an. Die .NET API überprüft die Verbindung zum OPC-Server wenn "connect" aufgerufen wurde. Wenn die Verbindung zum Server abbricht, versucht die .NET API die Verbindung wieder aufzubauen. Das Property "ServerState" gibt den Status des OPC-Servers an. Mögliche Werte sind DISCONNECTED, CONNECTED, ERRORSHUTDOWN, ERRORWATCHDOG. Ist der Status auf einem der beiden Fehlerzustände, versucht die .NET API die Verbindung neu aufzubauen. Der Client wird über das Event "ServerStateChanged" über Statusänderungen informiert.

Parameter	Funktionalität
<b>string url</b>	Die URL des OPC-Servers (Siehe unten: "Aufbau der URL").
<b>ConnectInfo</b>	"ConnectInfo" enthält mehrere Properties, die zur Initialisierung der .NET API dienen. String LocalID: Die Ländereinstellungen, die für sprachspezifische Rückgabewerte verwendet werden. Wird ein Leerstring übergeben, wird die Standardeinstellung verwendet. "bool ReconnectAfterConnectionError" / "bool RetryInitialConnect ". Ein Verbindungsauflauf zum OPC-Server kann wegen einer zeitweise nicht verfügbaren Netzwerkverbindung fehlgeschlagen. Durch Setzen der beiden Parameter, versucht die .NET API im Fehlerfall weiterhin die Verbindung aufzubauen. Das Property "IsConnected" wird dann auf jeden Fall gesetzt und der Client kann trotz des Verbindungsfehlers "Subscriptions" anlegen. Für die Items in den "Subscriptions" wird dann allerdings eine schlechte Qualität im "DataChanged"- Event geliefert, bis die Verbindung zum Server aufgebaut werden kann. Wenn der Verbindungsauflauf fehlgeschlagen und dieser Parameter nicht gesetzt ist, schlägt die Methode "Connect" fehl und es wird eine Exception gemeldet. "int keepAliveTime" ist ein Zeitintervall in Millisekunden in dem die Verbindung zum OPC-Server geprüft wird.
<b>int clientHandle</b>	Mit diesem Parameter kann die Applikation einen Index für dieses Serverobjekt vergeben. Dieses "ClientHandle" wird von der .NET API beim Event "ServerStateChanged" übergeben, um das Objekt in der Applikation zu identifizieren.
<b>out bool connectFailed</b>	Gibt an, ob der Verbindungsauflauf fehlgeschlagen ist, wenn der Parameter "retryConnect" gesetzt war. Wenn der Verbindungsauflauf fehlgeschlagen, und der Parameter "retryConnect" nicht gesetzt war, schlägt die Methode "connect" fehl und es wird eine Exception geworfen.

Die geforderte Form der URL kann entweder durch den Anwender eingegeben werden oder durch die Klasse "OpcServerEnum" im Namespace "SimaticNet.OpcCmn" automatisch ermittelt werden.

```
string url = "opcda://localhost/OPC.SimaticNET/{b6eacb30-42d5-11d0-9517-0020afaa4b3c}  
";  
  
string localId = "de";  
  
int clientHandle = 0;  
  
int keepAliveTime = 5000; // 5 Sekunden
```

---

```
daServerMgt.Connect(url, localId, clientHandle, true, keepAliveTime, out
connectFailed);
```

---

**Hinweis**

Mit dieser API können ausschließlich Verbindungen zu OPC-Servern der SIMATIC NET-Produktfamilie aufgebaut werden. Verbindungen zu anderen OPC-Servern oder zu OPC-Servern von anderen Herstellern sind nicht möglich.

---

**Hinweis**

Wenn diese API in einer Multi-Threaded-Applikation eingesetzt wird, stellen Sie sicher, dass der Thread in dem die Methode Connect ausgeführt wird, im selben "Appartment State" ausgeführt wird, wie die Applikation selbst.

Beispiel:

Wenn Sie eine Applikation mit "AppartementState STA" entwickeln und Connect in einem anderen Thread als dem Basisthread ausgeführt werden soll, setzen Sie den "Appartement state" des neuen Threads vor dem Start wie folgt:

```
Thread MyThread = new Thread(MyThreadStart);
MyThread.SetApartmentState(ApartmentState.STA);
```

Gehen Sie bei "Apartment State MTA" analog vor.

---

**Aufbau der URL**

Die URL, die den Server eindeutig bestimmt ist folgendermaßen aufgebaut:

- Für OPC COM Data Access  
[OpcSpecification]:///[Hostname]/[ServerIdentifier]
- Für OPC Unified Architecture  
[Endpunkt-Server-URL]

URL-Teil	Beschreibung
OpcSpecification	Gibt die zu verwendende OPC-DA-Spezifikation an <ul style="list-style-type: none"> <li>• opcda für OPC Data Access 2.05A bzw. 3.0 (COM)</li> <li>• http für OPC XML-DA 1.01</li> </ul>
Hostname	Rechnername oder IP-Adresse des Rechners, auf dem der OPC-Server betrieben wird. Für den lokalen Rechner ist dies localhost. Bei OPC-XML-DA-Servern kann neben der IP-Adresse auch noch ein Port angegeben sein.

URL-Teil	Beschreibung
ServerIdentifier	Identifiziert den OPC-Server auf dem angegebenen Rechner. <ul style="list-style-type: none"> <li>• OPC XML-DA - Pfad zum Webservice</li> <li>• OPC COM DA – [ProgID]/[optional ClassID]</li> </ul>
Endpunkt-Server-URL	opc.tcp://[hostname]:[port] <ul style="list-style-type: none"> <li>• opct.tcp ist die OPC-UA-Binary-TCP-Protokoll-Kennung</li> <li>• hostname ist der Rechnername oder IP-Adresse</li> <li>• port ist die Portnummer</li> </ul>

Beispiele für eine korrekte URL sind:

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 (für OPC.SimaticNET.S7)

### Methode Disconnect

Durch den Aufruf dieser Methode wird die Verbindung zum OPC-Server abgebaut. Alle bestehenden "Subscriptions" und Ressourcen werden freigegeben.

### Property IsConnected

Über diese Eigenschaft kann der aktuelle Connect-Zustand zur Client API abgefragt werden. Der Verbindungsstatus zum OPC-Server kann über das Property "ServerState" abgefragt werden.

```
if(daServerMgt.IsConnected == true)
{
    daServerMgt.Disconnect();
}
```

### Property ServerState

Über diese Eigenschaft kann der aktuelle Status der Verbindung zum OPC-Server abgefragt werden.

## Methode Read

Mit der Methode "Read" können Items des OPC-Servers gelesen werden. Werden Items zyklisch gelesen, empfiehlt sich das Anlegen einer "Subscription" und das Empfangen der geänderten Daten über den Event "DataChanged".

Parameter	Funktionalität
<b>int maxAge</b>	Das maximale Alter der zu lesenden Werte in Millisekunden. Sind die Werte, die sich aktuell im Cache des OPC-Servers befinden älter als durch maxAge angegeben, werden die Werte vom Device gelesen. Wird der Wert 0 übergeben erfolgt auf jeden Fall ein Lesen vom Device.
<b>ref ItemIdentifier[] itemIdentifiers</b>	Über das Array "itemIdentifiers" werden die OPC-Items angegeben, die gelesen werden sollen. Eventuelle Item-Fehler werden im Objekt "ResultID" des jeweiligen "ItemIdentifiers" zurückgegeben. Der Parameter ist ein in/out-Parameter, damit die .NET API Fehlerinformationen und "ServerHandle" im Objekt zurückgeben kann. Werden die gleichen Items mehrfach für "Read"- und "Write"-Aufrufe verwendet, sollten die Objekte nur einmal angelegt werden und dann für alle Aufrufe verwendet werden. Dadurch kann die .NET API die im Objekt gespeicherten Informationen für die Optimierung des OPC-Server-Zugriffs verwenden.
<b>out ItemValue[] itemValues</b>	Das Array "itemValues" enthält für die gelesenen OPC-Items jeweils Werte, Qualität und Zeitstempel.

Returntyp	Funktionalität
<b>ReturnCode</b>	Über den Rückgabewert kann ermittelt werden, ob es bezüglich der übergebenen Parameter Fehler gegeben hat und ob die Qualität aller gelesenen Werte gut ist. In Abhängigkeit vom Wert des "Enumerators" sollte eine Überprüfung des "ItemIdentifier[]" vorgenommen werden.

Der folgende Code zeigt beispielhaft, wie zwei Items mit einem "Read" gelesen werden:

```

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[]  
    }  
}
```

---

### Hinweis

Alle Methoden der .NET API erlauben Mengenafrufe, d.h. es können mit einem "Read"-Aufruf beliebig viele Items gelesen werden.

Es sollten bevorzugt Mengenafrufe zu verwenden, da durch diese Vorgehensweise zum einen die Client-Server-Kommunikation auf ein Minimum reduziert wird und zum anderen die Optimierungs-Mechanismen der API voll zur Geltung kommen.

---

## Methode ReadAsync

Mit der Methode "ReadAsync" können Items des OPC-Server asynchron gelesen werden. Die gelesenen Werte werden über den Event "ReadCompleted" geliefert. Werden Items zyklisch gelesen, empfiehlt sich das Anlegen einer "Subscription" und das Empfangen der geänderten Daten über den Event "DataChanged".

Parameter	Funktionalität
<b>int transactionHandle</b>	Ein Handle zur Identifikation für die "Read"-Transaktion. Das Handle wird beim Event "ReadCompleted" wieder an den Client übergeben.
<b>int maxAge</b>	Das maximale Alter der zu lesenden Werte in Millisekunden. Sind die Werte, die sich aktuell im Cache des OPC-Server befinden älter als durch "maxAge" angegeben, werden die Werte vom Device gelesen. Wird der Wert 0 übergeben erfolgt auf jeden Fall ein Lesen vom Device.
<b>ref ItemIdentifier[] itemIdentifiers</b>	Über das Array "itemIdentifiers" werden die OPC-Items angegeben, die gelesen werden sollen. Eventuelle Item-Fehler werden im Objekt "ResultID" des jeweiligen "ItemIdentifiers" zurückgegeben. Der Parameter ist ein in/out-Parameter, damit die .NET API Fehlerinformationen und "Server-Handle" im Objekt zurückgeben kann. Werden die gleichen Items mehrfach für "Read"- und "Write"-Aufrufe verwendet, sollten die Objekte nur einmal angelegt werden und dann für alle Aufrufe verwendet werden. Dadurch kann die .NET API die im Objekt gespeicherten Informationen für die Optimierung des OPC-Server-Zugriffs verwenden.

Returntype	Funktionalität
<b>ReturnCode</b>	Über den Rückgabewert kann ermittelt werden, ob es bezüglich der übergebenen Parameter Fehler gegeben hat und ob die Qualität aller gelesenen Werte gut ist. In Abhängigkeit vom Wert des "Enumerators" sollte eine Überprüfung des "ItemIdentifier[]" vorgenommen werden.

## Event ReadCompleted

Über den Event "ReadCompleted" schickt die .NET API das Ergebnis eines "ReadAsync"-Aufrufes an die Applikation.

Parameter	Funktionalität
<b>int transactionHandle</b>	Von der Applikation beim "ReadAsync" übergebenes "Handle".
<b>bool allQualitiesGood</b>	Zeigt an, dass keine Werte mit schlechter Qualität im "ItemValueCallback"-Array enthalten sind.
<b>bool noErrors</b>	Zeigt an, dass keine Fehler im "ItemValueCallback"-Array enthalten sind.
<b>ItemValueCallback[] itemValues</b>	Array mit Wert, Qualität und Zeitstempel für jedes Item im "ReadAsync"-Aufruf. Die Zuordnung erfolgt über das "ClientHandle" im Item "ValueCallback". Ist der Wert von "noErrors" "false", muss auch die Eigenschaft "ResultID" von "ItemValueCallback" überprüft werden.

Um sich am "DataChanged"-Event eines OPC-Server anzumelden, muss eine "Handler"-Methode am Event angemeldet werden. Der folgende Code zeigt beispielhaft, wie man eine "Handler"-Methode am "DataChanged"-Event anmeldet:

```
daServerMgt.ReadCompleted += new DaServerMgt.ReadCompletedEventHandler
(MyReadCompleted);

void MyReadCompleted(int transactionHandle,
bool allQualitiesGood,
bool noErrors,
ItemValueCallback[] itemValues)
{
    // ToDo: Fehlerprüfung
    // ToDo: Übernehmen der Werte in die Applikation
}
```

## Methode Write

Parameter	Funktionalität
<b>ref ItemIdentifier[] itemIdentifiers</b>	Über das Array "itemIdentifiers" werden die OPC-Items angegeben, die geschrieben werden sollen. Eventuelle Item-Fehler werden im Objekt "ResultID" des jeweiligen "ItemIdentifiers" zurückgegeben. Der Parameter ist ein in/out-Parameter, damit die .NET API Fehlerinformationen und "ServerHandle" im Objekt zurückgeben kann. Werden die gleichen Items mehrfach für "Read"- und "Write"-Aufrufe verwendet, sollten die Objekte nur einmal angelegt werden und dann für alle Aufrufe verwendet werden. Dadurch kann die .NET API die im Objekt gespeicherten Informationen für die Optimierung des OPC-Server-Zugriffs verwenden.
<b>ItemValue[] itemValues</b>	Das Array "itemValues" enthält die zu schreibenden Werte.

Returntype	Funktionalität
<b>ReturnCode</b>	Über den Rückgabewert kann ermittelt werden, ob es bezüglich der übergebenen Parameter Fehler gegeben hat. In Abhängigkeit vom Wert des "Enumerators" sollte eine Überprüfung des "ItemIdentifier[]" vorgenommen werden.

Der folgende Code zeigt beispielhaft wie ein einzelnes Item mit dem ganzzahligen Wert 11 bzw. 22 geschrieben wird:

```
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);

```

### Hinweis

Alle Methoden der .NET API erlauben Mengenaufrufe, d.h. es können mit einem "Write"-Aufruf beliebig viele Items geschrieben werden.

Es sollten bevorzugt Mengenaufrufe zu verwenden, da durch diese Vorgehensweise zum einen die Client- Server-Kommunikation auf ein Minimum reduziert wird und zum anderen die Optimierungsmechanismen der API voll zur Geltung kommen.

## Methode WriteAsync

Mit dieser Methode können Werte asynchron in den OPC-Server geschrieben werden. Die Information ob die Werte erfolgreich geschrieben wurden, wird über den Event "WriteCompleted" geliefert.

Parameter	Funktionalität
<b>int transactionHandle</b>	Ein Handle zur Identifikation für die "Write"-Transaktion. Das Handle wird beim Event "WriteCompleted" wieder an den Client übergeben.
<b>ref ItemIdentifier[] itemIdentifiers</b>	Über das Array "itemIdentifiers" werden die OPC-Items angegeben, die geschrieben werden sollen. Eventuelle Item-Fehler werden im Objekt ResultID des jeweiligen "ItemIdentifiers" zurückgegeben. Der Parameter ist ein in/out-Parameter, damit die .NET API Fehlerinformationen und "ServerHandle" im Objekt zurückgeben kann. Werden die gleichen Items mehrfach für "Read"- und "Write"-Aufrufe verwendet, sollten die Objekte nur einmal angelegt werden und dann für alle Aufrufe verwendet werden. Dadurch kann die .NET API die im Objekt gespeicherten Informationen für die Optimierung des OPC-Server-Zugriffs verwenden.
<b>ItemValue[] itemValues</b>	Das Array "itemValues" enthält die zu schreibenden Werte.

Returntyp	Funktionalität
<b>ReturnCode</b>	Über den Rückgabewert kann ermittelt werden, ob es bezüglich der übergebenen Parameter Fehler gegeben hat. In Abhängigkeit vom Wert des "Enumerators" sollte eine Überprüfung des "ItemIdentifier[]" vorgenommen werden.

## Event WriteCompleted

Über den Event "WriteCompleted" schickt die .NET API das Ergebnis eines "WriteAsync"-Aufrufes an die Applikation.

Parameter	Funktionalität
<b>int transactionHandle</b>	Von der Applikation beim "WriteAsync" übergebenes "Handle".
<b>bool noErrors</b>	Zeigt an, dass keine Fehler im "ItemResultCallback"-Array enthalten sind.
<b>ItemResultCallback[] itemResults</b>	Array mit "ClientHandle" und Fehlercode für jedes Item im "WriteAsync"-Aufruf. Die Zuordnung erfolgt über das "ClientHandle" im "ItemResultCallback". Ist der Wert von "noErrors" "false", muss auch die Eigenschaft "ResultID" von "ItemResultCallback" überprüft werden.

Um sich am "WriteCompleted"-Event eines OPC-Servers anzumelden, muss eine "Handler"-Methode am Event angemeldet werden. Der folgende Code zeigt beispielhaft, wie man eine "Handler"-Methode am "WriteCompleted"-Event anmeldet:

```
daServerMgt.WriteCompleted += new DaServerMgt.WriteCompletedEventHandler
(MyWriteCompleted);

void MyWriteCompleted(int transactionHandle,
                      bool noErrors,
                      ItemResultCallback[] itemResults)
{
    // ToDo: Fehlerprüfung
}
```

## Methode Browse

Mit der Methode Browse kann der Namensraum eines OPC-Servers durchsucht werden. Üblicherweise wird der Namensraum in einer Baumstruktur angezeigt, da diese Darstellung der Gliederung der Items und Ordner eines Servers entspricht.

Parameter	Funktionalität
<b>string itemName</b>	Der Parameter "itemName" gibt das Element (Verzeichnis) an für das alle Kinderelemente ermittelt werden sollen. Soll der Namensraum des Servers auf Wurzelebene durchsucht werden, ist ein Leerstring zu übergeben.
<b>string itemPath</b>	Über diesen Parameter wird bei einem OPC XML DA Server der "ItemPath" des Elements angegeben werden.

Parameter	Funktionalität
<b>ref string continuationPoint</b>	Wird die Anzahl der zurückgegebenen Elemente durch den Client (Parameter <code>maxElementsReturned</code> ) oder durch den Server auf eine bestimmte Anzahl beschränkt, gibt der Server über den Parameter "continuationPoint" einen Aufsetzpunkt für weitere "Browse"-Aufrufe auf dem gleichen Element zurück. Wird vom OPC-Server ein "ContinuationPoint" zurückgegeben, muss der "Browse"-Aufruf mit den gleichen Parametern und dem zurückgegebenen "ContinuationPoint" wiederholt werden, um die restlichen Kinderelemente zu ermitteln.
<b>int maxElementsReturned</b>	Hier kann die maximale Anzahl der Elemente angegeben werden, die beim Aufruf der Funktion zurückgeliefert werden sollen. Beträgt der übergebene Wert 0 werden alle Elemente zurückgegeben.
<b>BrowseFilter browseFilter</b>	Hier muss ein "BrowseFilter" bestimmt werden, der beschreibt, welche Art von Elementen zurückgegeben werden sollen. ("All", "Items" oder "Branch")
<b>int[] propertyIDs</b>	Hier können die IDs der Properties übergeben werden, die beim Aufruf der "Browse"-Methode ermittelt werden sollen. Die Properties werden im jeweiligen "BrowseElement" zurückgegeben.
<b>bool returnAllProperties</b>	Wird dieses Flag gesetzt, werden alle Properties eines Items automatisch ermittelt. Die Properties werden im jeweiligen "BrowseElement" zurückgegeben.
<b>bool returnPropertyValues</b>	Wird dieses Flag gesetzt, werden zusätzlich die Werte der angeforderten Properties ermittelt.
<b>out BrowseElement[] browseElements</b>	In diesem Array befinden sich die Kinderelemente des übergebenen Elementes im Adressraum.
<b>out bool moreElements</b>	"moreElements" gibt Auskunft darüber, ob alle Kinderelemente zurückgegeben wurden.

Returntype	Funktionalität
<b>ReturnCode</b>	Über den Rückgabewert kann ermittelt werden, ob es bezüglich der übergebenen Parameter Fehler gegeben hat. In Abhängigkeit vom Wert des "Enumerators" sollte eine Überprüfung des "BrowseElement[]" vorgenommen werden.

Der folgende Code zeigt beispielhaft, wie man alle Items und Ordner auf Wurzelebene ohne zugehörige Properties ermittelt:

```

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);

```

## Methode GetProperties

Mit der Methode "GetProperties" können die Eigenschaften eines OPC-Items ermittelt werden.

Parameter	Funktionalität
<b>ref ItemIdentifier[] itemIdentifiers</b>	Der Parameter "itemIdentifiers" beschreibt die OPC-Items für die Properties ermittelt werden sollen.
<b>int[] propertyIDs</b>	Hier können die IDs der Properties übergeben werden, die beim Aufruf der "GetProperties"-Methode ermittelt werden sollen. Die Properties werden im jeweiligen "ItemProperties"-Element zurückgegeben.
<b>bool returnAllProperties</b>	Wird dieses Flag gesetzt, werden alle Properties eines Items automatisch ermittelt. Die Properties werden im jeweiligen "ItemProperties"-Element zurückgegeben.
<b>out ItemProperties[] itemProperties</b>	In diesem Array befinden sich die angeforderten "ItemProperties"-Objekte. Die Properties in einem "ItemProperties"-Objekt beziehen sich jeweils auf das korrespondierende "ItemIdentifier"-Objekt mit dem gleichen Array-Index.

Returntyp	Funktionalität
<b>ReturnCode</b>	Über den Rückgabewert kann ermittelt werden, ob es bezüglich der übergebenen Parameter Fehler gegeben hat. In Abhängigkeit vom Wert des "Enumerators" sollte eine Überprüfung des "ItemIdentifier[]" vorgenommen werden.

Der folgende Code zeigt beispielhaft, wie man die Properties "AccessRights" und "DataType" des Items "Opcltem1" ermitteln kann:

```

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);

```

### Hinweis

Alle Methoden der .NET API erlauben Mengenaufrufe, d.h. es können mit einem "GetProperties"-Aufruf beliebig viele Items abgefragt werden.

Es sind bevorzugt Mengenaufrufe zu verwenden, da durch diese Vorgehensweise zum einen die Client- Server-Kommunikation auf ein Minimum reduziert wird und zum anderen die Optimierungsmechanismen der API voll zur Geltung kommen.

## Methode Subscribe

Über die Methode "Subscribe" können Items für die Überwachung von Werteänderungen angemeldet werden.

Vor dem Anlegen der ersten "Subscription" muss sich die Applikation am "DataChanged"-Event der .NET API anmelden. Diese Anmeldung ist beim Event "DataChanged" beschrieben.

Parameter	Funktionalität
<b>int clientSubscription</b>	Über "clientSubscription" kann die Applikation einen Index für die "Subscription" angeben. Dieses Handle wird benötigt, wenn mehrere "Subscriptions" angemeldet sind und die Applikation die "Subscriptions" im "DataChanged"-Event unterscheiden will.
<b>bool active</b>	Über das Flag "active" kann die "Subscription" aktiv oder inaktiv erzeugt werden.
<b>int updateRate</b>	Der Parameter "UpdateRate" gibt an, in welchem Zyklus Werteänderungen gemeldet werden sollen. Die "UpdateRate" wird in Millisekunden angegeben.
<b>out int revisedUpdateRate</b>	Dieser out-Parameter liefert die vom Server eingestellte "UpdateRate". Diese kann von der angeforderten "UpdateRate" abweichen.
<b>float deadband</b>	Über den Parameter "Deadband" wird die minimale Abweichung angegeben, ab der eine Werteänderung gemeldet wird. Der Wert gibt die prozentual Abweichung (Wert zwischen 0,0 und 100,0) vom Wertebereich an, ab der eine Änderung gemeldet wird. Der Wertebereich wird über die Properties "EULow" und "EUHigh" des OPC-Items angegeben. Der "Deadband" wird normalerweise nur auf Analogwerte angewendet.

Parameter	Funktionalität
<b>ref ItemIdentifier[] itemIdentifiers</b>	Über das Array "itemIdentifiers" werden die OPC-Items angegeben, die in die "Subscription" eingefügt werden sollen. Eventuelle Item-Fehler werden im Objekt "ResultID" des jeweiligen "ItemIdentifiers" zurückgegeben. Der Parameter ist ein in/out Parameter, damit die .NET API Fehlerinformationen und "ServerHandles" im Objekt zurückgeben kann. Die Objekte werden benötigt, wenn einzelne Items aus der "Subscription" entfernt werden sollen.
<b>out int serverSubscription</b>	Die .NET API gibt im Parameter "serverSubscription" das Handle für die "Subscription" zurück mit dem diese über die Methode "CancelSubscription" wieder gelöscht werden kann.

Returntyp	Funktionalität
<b>ReturnCode</b>	Über den Rückgabewert kann ermittelt werden, ob es bezüglich der übergebenen Parameter Fehler gegeben hat und ob die "UpdateRate" vom Server unterstützt wird. In Abhängigkeit vom Wert des "Enumerators" sollte eine Überprüfung des "ItemIdentifier[]" vorgenommen werden.

Der folgende Code zeigt beispielhaft, wie man eine "Subscription" anlegt:

```
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);
```

### Hinweis

Um alle "DataChanged"-Events bearbeiten zu können, muss zwingend die "Eventhandler"-Methode der Applikation an der .NET API angemeldet werden bevor die erste "Subscription" erzeugt wird.

## Methode SubscriptionModify

Über die Methode "SubscriptionModify" können die Eigenschaften einer "Subscription" verändert werden.

Parameter	Funktionalität
<b>int serverSubscription</b>	Identifiziert die "Subscription" in der .NET API. Dieses Handle wird beim Anlegen der "Subscription" mit der Methode "Subscribe" geliefert.
<b>bool active</b>	Über das Flag "active" kann die "Subscription" aktiv oder inaktiv geschaltet werden. Wird null übergeben, wird die Eigenschaft nicht verändert.
<b>int updateRate</b>	Der Parameter "updateRate" gibt an, in welchem Zyklus Werteänderungen gemeldet werden sollen. Die "UpdateRate" wird in Millisekunden angegeben. Wird null übergeben, wird die Eigenschaft nicht verändert.
<b>out int revisedUpdateRate</b>	Dieser out-Parameter liefert die vom Server eingestellte "UpdateRate". Diese kann von der angeforderten "UpdateRate" abweichen.
<b>float deadband</b>	Über den Parameter "deadband" wird die minimale Abweichung angegeben, ab der eine Werteänderung gemeldet wird. Der Wert gibt die prozentual Abweichung (Wert zwischen 0,0 und 100,0) vom Wertebereich an, ab der eine Änderung gemeldet wird. Der Wertebereich wird über die Properties "EULow" und "EUHigh" des OPC-Items angegeben. Der "Deadband" wird normalerweise nur auf Analogwerte angewendet. Wird null übergeben, wird die Eigenschaft nicht verändert.

Returntyp	Funktionalität
<b>ReturnCode</b>	Über den Rückgabewert kann ermittelt werden, ob es bezüglich der übergebenen Parameter Fehler gegeben hat und ob die "UpdateRate" vom Server unterstützt wird. In Abhängigkeit vom Wert des "Enumerators" sollte eine Überprüfung des "ItemIdentifier[]" vorgenommen werden.

Es stehen zusätzlich 3 Überladungen der Methode "SubscriptionModify" zur Verfügung, bei denen die Einstellungen "Active", "UpdateRate" und "Deadband" der "Subscription" jeweils einzeln verändert werden können.

## Methode SubscriptionAddItems

Über die Methode "SubscriptionAddItems" können Items zu einer bestehenden "Subscription" hinzugefügt werden.

Parameter	Funktionalität
<b>int serverSubscription</b>	Identifiziert die "Subscription" in der .NET API. Dieses Handle wird beim Anlegen der "Subscription" mit der Methode "Subscribe" geliefert.
<b>ref ItemIdentifier[] itemIdentifiers</b>	Über das Array "itemIdentifiers" werden die OPC-Items angegeben, die in die "Subscription" eingefügt werden sollen. Eventuelle Item-Fehler werden im Objekt "ResultID" des jeweiligen "ItemIdentifiers" zurückgegeben. Der Parameter ist ein in/out Parameter, damit die .NET API Fehlerinformationen und "ServerHandles" im Objekt zurückgeben kann. Die Objekte werden benötigt, wenn einzelne Items aus der "Subscription" entfernt werden sollen.

Returntyp	Funktionalität
<b>ReturnCode</b>	Über den Rückgabewert kann ermittelt werden, ob es bezüglich der übergebenen Parameter Fehler gegeben hat. In Abhängigkeit vom Wert des "Enumerators" sollte eine Überprüfung des "ItemIdentifier[]" vorgenommen werden.

## Methode SubscriptionRemoveItems

Über die Methode "SubscriptionRemoveItems" können Items von einer bestehenden "Subscription" entfernt werden.

Parameter	Funktionalität
<b>int serverSubscription</b>	Identifiziert die "Subscription" in der .NET API. Dieses Handle wird beim Anlegen der "Subscription" mit der Methode "Subscribe" geliefert.
<b>ref ItemIdentifier[] itemIdentifiers</b>	Über das Array "itemIdentifiers" werden die OPC-Items angegeben, die aus der "Subscription" entfernt werden sollen. Eventuelle Item-Fehler werden im Objekt "ResultID" des jeweiligen "ItemIdentifiers" zurückgegeben. Dieselben Objekte, die von "Subscribe" oder "SubscriptionAddItems" zurückgegeben werden sollten hier verwendet werden, um die richtigen Objekte durch das "ServerHandle" zu identifizieren. Der Parameter ist ein in/out-Parameter, damit die .NET API Fehlerinformationen im Objekt zurückgeben kann.

Returntyp	Funktionalität
<b>ReturnCode</b>	Über den Rückgabewert kann ermittelt werden, ob es bezüglich der übergebenen Parameter Fehler gegeben hat. In Abhängigkeit vom Wert des "Enumerators" sollte eine Überprüfung des "ItemIdentifier[]" vorgenommen werden.

## Methode SubscriptionCancel

Über die Methode "SubscriptionCancel" kann eine "Subscription" wieder vom "DataChanged" abgemeldet werden. Hierbei muss der bei "Subscribe" erhaltene Parameter "serverSubscription" übergeben werden.

Parameter	Funktionalität
int serverSubscription	Identifiziert die "Subscription" in der .NET API. Dieses Handle wird beim Anlegen der "Subscription" mit der Methode "Subscribe" geliefert.

Der folgende Code zeigt, wie man eine bestehende "Subscription" wieder entfernen kann:

```
daServerMgt.SubscriptionCancel(serverSubscription);
```

## Event DataChanged

Über den Event "DataChanged" schickt die .NET API geänderte Werte für Items einer "Subscription" an die Applikation.

Parameter	Funktionalität
int clientSubscription	Von der Applikation beim Anlegen der "Subscription" übergebenes "Handle".
bool allQualitiesGood	Zeigt an, dass keine Werte mit schlechter Qualität im "ItemValueCallback"-Array enthalten sind.
bool noErrors	Zeigt an, dass keine Fehler im "ItemValueCallback"-Array enthalten sind.
ItemValueCallback[] itemValues	Array mit Wert, Qualität und Zeitstempel für jedes geänderte Item. Die Zuordnung erfolgt über das "ClientHandle" im "ItemValueCallback". Ist der Wert von "noErrors" "false", muss auch die Eigenschaft "ResultID" von "ItemValueCallback" überprüft werden.

Um sich am "DataChanged"-Event eines OPC-Servers anzumelden, muss eine "Handler"-Methode am Event angemeldet werden. Der folgende Code zeigt beispielhaft, wie man eine "Handler"-Methode am "DataChanged"-Event anmeldet:

```
daServerMgt.DataChanged += new DaServerMgt.DataChangedEventHandler(MyDataChanged);
```

```
void MyDataChanged(int clientSubscription,
                   bool allQualitiesGood,
                   bool noErrors,
                   ItemValueCallback[] itemValues)
{
    if (noErrors)
    {
        if (allQualitiesGood)
        {
            ...
        }
    }
}
```

```

    // alle Items sind korrekt

    // ToDo: Übernehmen der Werte in die Applikation

}

else

{

    // bei mindestens einem Item ist die Quality nicht gut

    // ToDo: Fehlerbehandlung

}

}

else

{

    // mindestens ein Item hat einen Fehler

    // ToDo: Fehlerbehandlung

}

}
}

```

### Event ServerStateChanged

Über den Event "ServerStateChanged" informiert die .NET API die Applikation über Änderungen des Verbindungsstatus.

Parameter	Funktionalität
<b>int clientHandle</b>	Von der Applikation beim "Connect" übergebenes "Handle" für die Verbindung zum OPC-Server.
<b>ServerState state</b>	Aktueller Status der Verbindung zum OPC-Server.

Um sich am "ServerStateChanged"-Event eines OPC-Servers anzumelden, muss eine "Handler"-Methode am Event angemeldet werden. Der folgende Code zeigt beispielhaft die Anmeldung einer "Handler"-Methode am "ServerStateChanged"-Event:

```

daServerMgt.ServerStateChanged += new
DaServerMgt.ServerStateChangedEventHandler(MyServerStateChanged);

void MyServerStateChanged(int clientHandle,
ServerState state)
{
    // identifizieren der Verbindung
    // ToDo: Reaktion auf den neuen Status des Servers
}

```

### 5.2.1.3 Fehlerbehandlung

Auftretende Fehler werden über Exceptions an den Benutzer weitergemeldet. Hierzu werden zwei Arten von Exceptions benutzt. Fehler, die durch die Übergabe falscher Parameter, Null-Parameter oder unzulässiger Kombinationen von Parameter verursacht werden, gibt die .NET API als System Exceptions zurück. Fehler, die von dem darunterliegenden OPC-Server zurück gegeben werden, gibt die .NET API als OPC-Exceptions zurück.

#### OPCExceptions

Die Klasse "OpcException" ist von "ApplicationException" abgeleitet. Als zusätzlicher Parameter hat "OPCException" eine "ResultID".

#### SystemExceptions

Die .NET OPC API generiert drei verschiedene Klassen von "SystemExceptions".

- ArgumentNullException – wird benutzt, wenn eine Methode einen null Parameter nicht zuläßt.
- ArgumentOutOfRangeException – z.B. wenn eine Zahl > 100 als "Deadband" gesetzt wird.
- ArgumentException – z.B. wenn beim Aufrufen von Write() die Arrays "ItemIdentifiers" und "Values" nicht die gleiche Länge haben.

```

int ErrorCode;
string ErrorName;
string ErrorDescription;
try
{
    daServerMgt.Write( ref itemIdentifiers,
    itemValues);
}
catch(OPCEexception opcex)
{
    // Fehler
    ErrorCode = opcex.ResultID.Code;
    ErrorName = opcex.ResultID.Name;
    ErrorDescription = opcex.ResultID.Description;

    // OPC spezifische Fehlerbehandlung
    // ToDo. . .
}
catch(System.ArgumentException sysarex)
{
    // Fehler
    ErrorDescription = sysarex.Message;
    // Spezielle Fehlerbehandlung für ArgumentException
    // ToDo. . .
}

```

```

        catch(System.Exception)
        {
            // Fehlerbehandlung für unerwartete Exception
            // ToDo. . .
        }
    }
}

```

## 5.2.2 Beispiel zu Namensraum SIMATICNET.OPCDAClient

In diesem Beispiel wird abstrakt beschrieben, wie ein Client, der die OPCDAClient-API benutzt, sich zu verschiedenen OPC-Server-Typen verbinden kann.

Detaillierte Informationen können sie in den jeweilig angegebenen Kapiteln finden.

### Beispiel für den Verbindungsaufbau zu OPC DA und OPC-UA-Servern (mit / ohne Verschlüsselung)

```

//Server Instanz Initialisierung
//Informationen dazu können im Kapitel "DaServerMgt" gefunden werden
DaServerMgt m_Server = new DaServerMgt();

//Server Identifier Instanz Initialisierung
//Informationen dazu können im Kapitel "Klasse ServerIdentifier"
//gefunden werden
ServerIdentifier m_ServerId =
    new ServerIdentifier(m_EndpointUrl, m_EndpointIdentifier);

//Nur zum Server verbinden, wenn noch keine Verbindung besteht
if (m_Server.IsConnected == false)
{
    //ConnectInfo Instanz ausfüllen
    //Informationen dazu können im Kapitel "Klasse ConnectInfo"
    //gefunden werden
    ConnectInfo connectInfo             = new ConnectInfo();
    connectInfo.RetryAfterConnectionError = true;
    connectInfo.RetryInitialConnection   = true;
    connectInfo.KeepAliveTime           = 1000;

    //UA-spezifische Parameter setzen, wenn der zu verbindende Server
    //ein UA-Server ist
    if (m_ServerId.Category == ServerCategory.OPCUA)
    {
        //ConnectInfo Instanz ausfüllen
        //Informationen dazu können im Kapitel "Klasse ConnectInfo"
        //gefunden werden
        connectInfo.MessageSecurityMode      =
            m_ServerId.Endpoint.MessageSecurityMode;
        connectInfo.SecurityPolicyUri        =
            m_ServerId.Endpoint.SecurityPolicyUri;
        connectInfo.ServerCertificate        =
            m_ServerId.Endpoint.ServerCertificate;

        //Setzen von Parametern, die zum Verbindungsaufbau mit
        //verschlüsselten UA-Servern notwendig sind
    }
}

```

```

if (1 < connectInfo.MessageSecurityMode)
{
    //ConnectInfo Instanz ausfüllen
    //Informationen dazu können im Kapitel "Klasse ConnectInfo"
    //gefunden werden
    //Beispiel für Ablagepfad im WindowsStore
    connectInfo.CertificateStoreName =
        Application.ProductName;
    connectInfo.CertificateStoreLocation =
        WinStoreLocation.CurrentUser;

    //Das Certificate Handling wird gemacht, damit nicht bei
    //jedem Start der Applikation
    //ein neues Zertifikat erzeugt werden muss, das wiederum
    //vom Server akzeptiert werden muss.
    //Dazu müssen die Thumbprints der verwendeten Zertifikate
    //z.B. im Dateisystem gespeichert werden.

    #region Certificate Handling Client

    //Der Thumbprint ist nötig um das Client Zertifikat im
    //Windows Store wiederzufinden
    byte[] clientThumbprint = null;
    //Clientzertifikat - Informationen dazu können im Kapitel
    //"Klasse PkiCertificate" gefunden werden
    PkiCertificate cert_Client = null;

    //ToDo: Auslesen des gespeicherten Thumbprint des Client
    //""clientThumbprint" muss hier geschehen

    //Bestehendes Zertifikat aus dem Windows Store holen
    if (clientThumbprint != null)
    {
        try
        {
            cert_Client =
                PkiCertificate.fromWindowsStoreWithPrivateKey(
                    connectInfo.CertificateStoreLocation,
                    connectInfo.CertificateStoreName,
                    clientThumbprint);
        }
        catch
        {
            cert_Client = null;
        }
    }

    //Sollte noch kein Zertifikat im WindowsStore vorhanden
    //sein, muss es neu angelegt werden
    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: Speichern des Thumbprint
    //"cert_Client.Thumbprint"
    //muss hier geschehen
}

#endregion

#region Certificate Handling Server
//Der Thumbprint ist nötig um das Server Zertifikat im
//Windows Store wiederzufinden
byte[] serverThumbprint = null;
//Serverzertifikat - Informationen dazu können im Kapitel
//"Klasse PkiCertificate" gefunden werden
PkiCertificate cert_Server = null;

//ToDo: Auslesen des gespeicherten Serverthumbprint
//"serverThumbprint" muss hier geschehen

//Bestehendes Zertifikat aus Windows Store holen
if (serverThumbprint != null)
{
    try
    {
        cert_Server =
            PkiCertificate.fromWindowsStore(
                connectInfo.CertificateStoreLocation,
                connectInfo.CertificateStoreName,
                serverThumbprint);
    }
    catch
    {
        cert_Server = null;
    }
}

// Sollte noch kein Zertifikat im WindowsStore vorhanden
//sein, muss das Zertifikat des zu verbindenden Server
//in den gleichen Windows Store wie das Client Zertifikat
//abgelegt werden, damit es als akzeptiert gilt
if (cert_Server == null)
{
    //Hier kann abgefragt werden, ob das neue Zertifikat
    //akzeptiert werden soll

    cert_Server =
        PkiCertificate.fromDER(connectInfo.ServerCertificate);
}

```

```

        cert_Server.toWindowsStore(
            connectInfo.CertificateStoreLocation,
            connectInfo.CertificateStoreName);

        //ToDo: Speichern des Thumbprint
        //"cert_Server.Thumbprint" muss hier geschehen
    }
    else
    {
        //Hier kann überprüft werden, ob das aktuelle
        //Serverzertifikat dem gespeicherten Zertifikat
        //entspricht
    }

#endregion

//Private key setzen
connectInfo.ClientPrivateKey = cert_Client.PrivateKey;

//Zertifikat verschlüsseln
connectInfo.ClientCertificate = cert_Client.toDER();
}
}

bool connectFailed;

//Verbindungs-Auftrag an den Server senden
m_Server.Connect(m_ServerId.Url, m_ClientHandle,
    ref connectInfo, out connectFailed);

if (connectFailed)
{
    //Hier können Fehler beim Verbindungsaufbau zum Server
    //behandelt werden
}
else
{
    //Server ist schon verbunden
}

```

## 5.2.3

## Namensraum SIMATICNET.OPCCMN

Im Namensraum "SimaticNet.OpcCmn" sind allgemeine Funktionalitäten für OPC zusammengefasst. Dies betrifft vor allem das Suchen von OPC-Servern und das Ermitteln von Informationen die für den Verbindungsaufbau notwendig sind.

### 5.2.3.1 Die Schnittstelle des Objektes OPCServerEnum

Über die Klasse "OpcServerEnum" können COM-OPC-Server auf einem Rechner gesucht werden. Außerdem können Informationen ermittelt werden, die für den Verbindungsauflauf zum OPC-Server benötigt werden.

#### Erzeugen des OPCServerEnum-Objektes

Um die Klasse "OpcServerEnum" verwenden zu können, muss zuerst eine Instanz der Klasse erzeugt werden.

Der folgende Code zeigt, wie ein "OpcServerEnum"-Objekt erzeugt wird:

```
OpcServerEnum opcServerEnum = new OpcServerEnum();
```

#### Methode EnumComServers

Mit der Methode "EnumComServers" können die OPC-Server auf einem Rechner ermittelt werden. Hier ist eine Unterscheidung der unterschiedlichen OPC-Spezifikationen möglich z.B. "OPC Data Access", "OPC Historical Data Access" oder "OPC Alarm & Events".

Parameter	Funktionalität
<b>string</b> nodeName	Der Name oder die IP-Adresse des Rechners auf dem OPC-Server gesucht werden sollen. (z.B. localhost, PCTest, 192.168.0.120 usw.)
<b>bool</b> returnAllServers	Gibt an, ob alle OPC-Server auf dem Rechner zurückgegeben werden sollen. Ist dieser Parameter "true" wird das Array "serverCategories" nicht ausgewertet.
<b>ServerCategory[]</b> serverCategories	Über diesen Parameter kann angegeben werden, welche Arten von OPC-Servern zurückgegeben werden sollen.
<b>out ServerIdentifier[]</b> serverIdentifiers	Eine Liste von "ServerIdentifier"-Objekten. Die Objekte enthalten jeweils die Informationen, die zum Verbindungsauflauf mit dem OPC-Server benötigt werden.

Der folgende Code ermittelt alle OPC-Data-Access-Server auf dem lokalen Rechner:

```
ServerCategory[] serverCategories = new ServerCategory[1];
serverCategories[0] = ServerCategory.OPCDA;
ServerIdentifier[] serverIdentifier = null;

opcServerEnum.EnumComServer("localhost",
false,
serverCategories,
out serverIdentifier);
```

## Methode ClsidFromProgID

Die Methode "ClsidFromProgID" ermittelt in Abhängigkeit der übergebenen Parametern Rechnername und "ProgID" die zugehörige "CLSID" des OPC-Server.

Parameter	Funktionalität
<b>string nodeName</b>	Der Name bzw. die IP-Adresse des Rechners auf dem die "CLSID" ermittelt werden soll. (z.B. localhost, PCTest, 192.168.0.120 usw.)
<b>string progID</b>	Die "ProgID" für die die zugehörige "CLSID" ermittelt werden soll.
<b>out string clsID</b>	Die angeforderte "CLSID"

Der folgende Code ermittelt die "CLSID" zur "ProgID OPC.SimaticNET" auf dem lokalen Rechner:

```
string progID = "OPC.SimaticNET";
string clsID = null;
opcServerEnum.ClsidFromProgId("localhost",
    progID,
    out clsID);
```

## Methode getCertificateForEndpoint

Die Methode "getCertificateForEndpoint" verbindet sich zu der übergebenen Endpunkt-Url und lädt das Serverzertifikat, welches als Out-Parameter zurückgegeben wird. Die In-Parameter erhalten Sie zuvor durch browsen auf dem Discovery Server.

Parameter	Funktionalität
<b>string endpointUrl</b>	Die Url des UA-Endpunktes (z.B. opc.tcp://localhost:4841)
<b>string securityPolicyUri</b>	Die SecurityPolicy des Endpunktes (z.B."http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15")
<b>Byte messageSecurityMode</b>	Der MessageSecurityMode des Endpunktes None, Sign oder SignAndEncrypt
<b>out byte[] serverCertificate</b>	Das Zertifikat des OPC-UA-Servers.

Der folgende Code lädt das Zertifikat von dem durch die Parameter vorgegebenen Endpunkt:

```
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 Klasse ServerIdentifier

Die Klasse "ServerIdentifier" liefert alle Daten, die zum Verbindungsaufbau mit einem OPC-Server benötigt werden.

Eigenschaften der Klasse "ServerIdentifier":

- **string Url**

Die vollständige URL des OPC-Servers. Diese kann zum Beispiel bei der "Connect"-Methode übergeben werden. Siehe auch Aufbau der Url.

- **ServerCategory Category**

Die Opc-Category, in die der Server einzuordnen ist, z.B. OPC DA, OPC DX, usw.

- **string ProgID**

Die "ProgID" des OPC-Servers.

- **string CLSID**

Die "CLSID" des OPC-Servers.

- **string HostName**

Der Name des Rechners, auf dem der OPC-Server ermittelt wurde.

- **EndpointIdentifier**

Zusätzliche Informationen die zur Verbindung mit einem OPC-UA-Server notwendig sind.

### 5.2.3.3 Enumerator ServerCategory

Über den Enumerator "ServerCategory" kann die Art des OPC-Servers angegeben werden. Enthaltene Elemente sind:

- OPCDA

Entspricht den OPC-Spezifikationen "OPC DA 2.05A" und "OPC DA 3.00"

- OPCUA

Entspricht der OPC-Spezifikation Unified Architecture

- OPCXMLDA

Entspricht der OPC-Spezifikation "OPC XML DA 1.01" (derzeit nicht unterstützt)

- OPCDX

Entspricht der OPC-Spezifikation "OPC DX 1.00" (derzeit nicht unterstützt)

- OPCAE

Entspricht der OPC-Spezifikation "OPC AE 1.10" (derzeit nicht unterstützt)

- OPCHDA

Entspricht der OPC-Spezifikation "OPC HDA 1.10" (derzeit nicht unterstützt)

---

#### Hinweis

Da OPC-XML-Da-Server nicht wie COM-OPC-Server auf einem Rechner registriert sind, können diese nicht über den "OpcServerEnum" ermittelt werden.

Um sich zu einem OPC-XML-DA-Server zu verbinden, muss die URL bekannt sein.

---

#### 5.2.3.4 Klasse EndpointIdentifier

Die Klasse "EndpointIdentifier" liefert alle Daten, die zusätzlich zum Verbindungsauftbau mit einem OPC-UA-Server benötigt werden.

Eigenschaften der Klasse "EndpointIdentifier":

- string SecurityPolicy  
Die URI der SecurityPolicy die zur Sicherung der Verbindung benutzt wird.
- byte MessageSecurityMode  
Der Typ der Verbindungssicherheit auf Nachrichten Ebene.
- byte ServerCertificate  
Das X509 Zertifikat des OPC-UA-Servers.
- string ApplicationUri  
Die URI der OPC-UA-Server-Applikation.
- string ProductUri  
Die Product URI der OPC-UA-Server-Applikation.
- string ApplicationName  
Der Name des OPC-UA-Servers.

#### 5.2.3.5 Klasse PkiCertificate

Die Klasse "PkiCertificate" kapselt ein X509-Zertifikat und ermöglicht eine einfache Erzeugung eines solchen Zertifikats. Weiter werden Methoden zum Zugriff auf den "WindowsCertificateStore" zum Laden und Speichern von Zertifikaten zur Verfügung gestellt. Die Felder des Zertifikats können über Properties nur gelesen werden.

#### 5.2.3.6 Erzeugen eines neuen Zertifikats

Die Klasse "PkiCertificate" bietet einen Konstruktor, der als Parameter alle Informationen übernimmt, die zum Erzeugen des Zertifikats benötigt werden.

Parameter	Funktionalität
<b>string URI</b>	Die "ApplicationURI" des Zertifikats. Ein eindeutiger Bezeichner für das Zertifikat, z.B: urn:<RechnerName>:<Firma>:<ProductName>
<b>string IP</b>	Die IP-Adresse des Rechners, auf dem die Applikation läuft. Diese wird nur benutzt, wenn kein "DomainName" verfügbar ist.
<b>string DNS</b>	Der "DomainName" des Rechners (Rechnername), auf dem die Applikation läuft.
<b>int validTime</b>	Die Zeit in Sekunden, die das Zertifikat gültig sein soll.
<b>string CommonName</b>	Der Anzeigename des Zertifikats.
<b>string Organization</b>	Organisation oder Firma z.B. Siemens
<b>string OrganizationUnit</b>	Abteilung innerhalb der Organisation oder Firma.
<b>string Locality</b>	Ort des Ausstellers.
<b>string State</b>	State oder Bundesland des Ausstellers.
<b>string Country</b>	Ländercode z.B. DE, US, ...
<b>int KeyStrength</b>	Länge des Schlüssels z.B. 1024 bit, 2048 bit ...

Der folgende Code zeigt beispielhaft, wie ein neues "PkiCertificate" erzeugt werden kann:

```
PkiCertificate clientCert = new PkiCertificate(
    "urn:my_machine:Siemens:SimaticNETSampleApplication",
    "",
    "Rechner1",
    31536000, // 1 Jahr = 31536000 Sekunden
    "SampleApplication",
    "Siemens",
    "SimaticNET",
    "Karlsruhe",
    "Baden-Württemberg",
    "DE",
    2048);
```

### 5.2.3.7 Methode toDER

Mit dieser Methode kann aus einem "PkiCertificate" ein DER-kodiertes Byte-Array generiert werden. Zertifikate in Form eines DER-kodierten Byte-Arrays werden z.B. in der Klasse EndpointIdentifier oder in der benötigt.

Returntyp	Funktionalität
byte[]	Byte-Array, das die DER-kodierten Daten des Zertifikats oder null enthält, wenn die Konvertierung fehlschlägt.

Der folgende Code zeigt beispielhaft, wie aus einem existierenden PkiCertificate ein DER-kodiertes Byte-Array erzeugt wird:

```
// Lade Client Zertifikat aus Windows Store
PkiCertificate clientCert;
...
byte[] clientCertificateDER = clientCertificate.toDER();
```

### 5.2.3.8 Methode fromDER

Mit dieser statischen Methode kann aus einem DER-kodierten Byte-Array ein "PkiCertificate" erzeugt werden.

Parameter	Funktionalität
byte[] DERData	Das DER-kodierte Byte-Array, welches die Daten eines X509-Zertifikats enthält.

Returntyp	Funktionalität
PkiCertificate	Das neu erzeugte "PkiCertificate" oder null, wenn die Konvertierung fehlschlägt.

Der folgende Code zeigt beispielhaft, wie aus einem vorhandenen byte[] ein "PkiCertificate" erzeugt wird:

```
Byte[] DERdata;
...
PkiCertificate cert = PkiCertificate.fromDER(DERdata);
```

### 5.2.3.9 Methode toWindowsStore

Mit dieser Methode wird das "PkiCertificate" im "WindowsCertificateStore" gespeichert.

Parameter	Funktionalität
<b>WinStoreLocation location</b>	Benutzerkontext zum Speichern im "WindowsCertificateStore".
<b>string StoreName</b>	Name des "Store" zum Speichern im "WindowsCertificateStore".

Der folgende Code zeigt beispielhaft, wie ein vorhandenes "PkiCertificate" im "WindowsCertificateStore" gespeichert wird:

```
PkiCertificate cert;
...
Cert.toWindowsStore(WinStoreLocation.CurrentUser, "UA
Applications");
```

---

#### Hinweis

Es wird empfohlen die Zertifikate im "WinStoreLocation.LocalMachine" unter dem StoreName "UA Applications" abzuspeichern, da dies von der OPC Foundation vorgeschlagen ist.

---

### 5.2.3.10 Methode toWindowsStoreWithPrivateKey

Mit dieser Methode wird das "PkiCertificate" inklusive dem Private-Schlüssel im "WindowsCertificateStore" gespeichert.

Parameter	Funktionalität
<b>WinStoreLocation location</b>	Benutzerkontext zum Speichern im "WindowsCertificateStore".
<b>string StoreName</b>	Name des "Store" zum Speichern im "WindowsCertificateStore".

Der folgende Code zeigt beispielhaft, wie ein vorhandenes "PkiCertificate" im "WindowsCertificateStore" gespeichert wird:

```
PkiCertificate cert;
...
Cert.toWindowsStore(WinStoreLocation.CurrentUser, "MyStoreName");
```

### 5.2.3.11 Methode fromWindowsStore

Mit dieser Methode wird ein "PkICertificate" aus dem "WindowsCertificateStore" geladen.

Parameter	Funktionalität
<b>WinStoreLocation</b> location	Benutzerkontext zum Laden aus dem "WindowsCertificateStore".
<b>string</b> StoreName	Name des "Store" zum Laden aus dem "WindowsCertificateStore".
<b>byte[]</b> Thumbprint	Der "Thumbprint" des Zertifikats. Dieser kann z.B. aus einer Konfigurationsdatei geladen werden.

Returntyp	Funktionalität
<b>PkICertificate</b>	Das neu erzeugte "PkICertificate" oder null, wenn das Laden fehlgeschlägt.

Der folgende Code zeigt beispielhaft, wie ein "PkICertificate" aus dem "WindowsCertificateStore" geladen wird:

```
byte[] Thumbprint;
// ... lade Thumbprint aus Konfiguration ...
PkICertificate cert = PkICertificate.fromWindowsStore(
    WinStoreLocation.CurrentUser,
    "MyStoreName",
    Thumbprint);
```

### 5.2.3.12 Methode fromWindowsStoreWithPrivateKey

Mit dieser Methode werden ein "PkICertificate" und der dazu gehörige Private-Schlüssel aus dem "WindowsCertificateStore" geladen. Der Private-Schlüssel ist eine Property des "PkICertificate".

Parameter	Funktionalität
<b>WinStoreLocation</b> location	Benutzerkontext zum Laden aus dem "WindowsCertificateStore".
<b>string</b> StoreName	Name des "Store" zum Laden aus dem "WindowsCertificateStore".
<b>byte[]</b> Thumbprint	Der "Thumbprint" des Zertifikats. Dieser kann z.B. aus einer Konfigurationsdatei geladen werden.

Returntyp	Funktionalität
<b>PkICertificate</b>	Das neu erzeugte "PkICertificate" oder null, wenn das Laden fehlgeschlägt.

Der folgende Code zeigt beispielhaft, wie ein "PkICertificate" und der dazu gehörige Private-Schlüssel aus dem "WindowsCertificateStore" geladen werden:

```

byte[] Thumbprint;
// ... lade Thumbprint aus Konfiguration ...
PkiCertificate cert = PkiCertificate.fromWindowsStoreWithPrivateKey(
    WinStoreLocation.CurrentUser,
    "MyStoreName",
    Thumbprint);

```

### 5.2.3.13 Methode fromWindowsStoreWithPrivateKey

Mit dieser Methode werden ein "PkiCertificate" und der dazu gehörige Private-Schlüssel aus dem "WindowsCertificateStore" geladen. Der Private-Schlüssel ist eine Property des "PkiCertificate".

Parameter	Funktionalität
<b>WinStoreLocation</b> location	Benutzerkontext zum Laden aus dem "WindowsCertificateStore".
<b>string</b> StoreName	Name des "Store" zum Laden aus dem "WindowsCertificateStore".
<b>string</b> ApplicationURI	Der "Thumbprint" des Zertifikats. Dieser kann z.B. aus einer Konfigurationsdatei geladen werden.

Returntyp	Funktionalität
<b>PkiCertificate</b>	Das neu erzeugte "PkiCertificate" oder null, wenn das Laden fehlgeschlägt

Der folgende Code zeigt beispielhaft, wie ein "PkiCertificate" und der dazu gehörige private Schlüssel aus dem "WindowsCertificateStore" geladen werden:

```

String sApplicationUri;
// ... lade ApplicationUri aus Konfiguration ...
PkiCertificate cert = PkiCertificate.fromWindowsStoreWithPrivateKey(
    WinStoreLocation.CurrentUser,
    "MyStoreName",
    sApplicationUri);

```

### 5.2.3.14 Properties

Die Properties an dieser Klasse sind alle nur lesbar. Sie repräsentieren die verschiedenen Felder des intern gehaltenen X509-Zertifikats.

- **PrivateKey** [readonly]
- **CommonName** [readonly]
- **Thumbprint** [readonly]
- **IPAddress** [readonly]
- **ApplicationURI** [readonly]
- **ValidFrom** [readonly]
- **ValidTo** [readonly]

- SerialNumber [readonly]
- SignatureAlgorithm [readonly]
- CipherStrength [readonly]
- Subject [readonly]
- Issuer [readonly]

### 5.2.3.15 Enumerator WinStoreLocation

Über den "Enumerator WinStoreLocation" kann angegeben werden unter welchem Benutzerkontext die Zertifikate im "WindowsCertificateStore" geladen bzw. gespeichert werden.

Enthaltene Elemente sind:

- LocalMachine  
Zertifikate werden an folgender Stelle in der Registry gespeichert:  
"HKEY\_LOCAL\_MACHINE\Software\Microsoft\SystemCertificates"
- CurrentUser  
Zertifikate werden an folgender Stelle in der Registry gespeichert:  
"HKEY\_CURRENT\_USER\Software\Microsoft\SystemCertificates"
- CurrentService  
Zertifikate werden an folgender Stelle in der Registry gespeichert:  
"HKEY\_LOCAL\_MACHINE\Software\Microsoft\Cryptography\Services\<ServiceName>\SystemCertificates"
- Services  
Zertifikate werden an folgender Stelle in der Registry gespeichert:  
"HKEY\_LOCAL\_MACHINE\Software\Microsoft\Cryptography\Services\<ServiceName>\SystemCertificates"
- Users  
Zertifikate werden an folgender Stelle in der Registry gespeichert:  
"HKEY\_USERS\<UserName>\Software\Microsoft\SystemCertificates"

# Beispielprogramme

Dieses Kapitel enthält Beispielprogramme, die die Custom- bzw. Automation-Schnittstelle benutzen.

Die Beispielprogramme finden Sie nach der Installation der "SIMATIC NET PC Software" unter: "<Installationspfad>\SIEMENS\SIMATIC.NET\opc2\samples"

## 6.1 OPC-Automation-Schnittstelle (Synchrone Kommunikation) in VB.NET

Das vorliegende Beispiel in Visual Basic nutzt die Automation-Schnittstelle für Data Access V2.0 von OPC, um Daten synchron zu lesen und zu schreiben.

Diese Beschreibung gliedert sich in folgende Abschnitte:

### 6.1.1 Aktivieren der Simulationsverbindung

Damit das vorliegende Programm überhaupt lauffähig ist, müssen Sie eine Simulationsverbindung aktivieren, die die im Programm verwendeten Demo-Variablen verfügbar macht. Gehen Sie dazu folgendermaßen vor:

1. Starten Sie das Konfigurationsprogramm "Kommunikations-Einstellungen" über das Startmenü:  
"Start" > "Alle Programme" > "Siemens Automation" > "SIMATIC" > "SIMATIC NET" > "Kommunikations-Einstellungen".  
Reaktion: Das Konfigurationsprogramm "Kommunikations-Einstellungen" wird geöffnet.
2. Öffnen Sie im linken Navigationsfenster den Eintrag "OPC-Protokollauswahl":  
"SIMATIC NET-Einstellungen" > "OPC-Einstellungen" > "OPC-Protokollauswahl".
3. Aktivieren Sie die Optionskästchen für das zu simulierende Protokoll.  
Das vorliegende Beispiel verwendet das S7-Protokoll.  
Aktivieren Sie deshalb unter "Name: S7" die Optionskästchen und klicken Sie auf das Pfeilsymbol neben dem S7-Protokoll.  
Reaktion: Die erweiterte Parameterliste wird geöffnet.
4. Aktivieren Sie das Optionskästchen "virtuelle Baugruppe (DEMO) für die Simulation bereitstellen".
5. Bestätigen Sie mit "Übernehmen".
6. Beenden Sie das Konfigurationsprogramm "Kommunikations-Einstellungen".

**Hinweis**

Damit die Änderungen wirksam werden, müssen zuvor alle OPC-Clients beendet werden!

### 6.1.2

### Bedienung des Beispielprogramms

Das Programm befindet sich auf Ihrer Festplatte unter  
"<Installationspfad>\SIEMENS\SIMATIC.NET\opc2\samples\automation\sync.net"

Beim Programmstart ist nur die Schaltfläche "Start Sample" aktiviert:

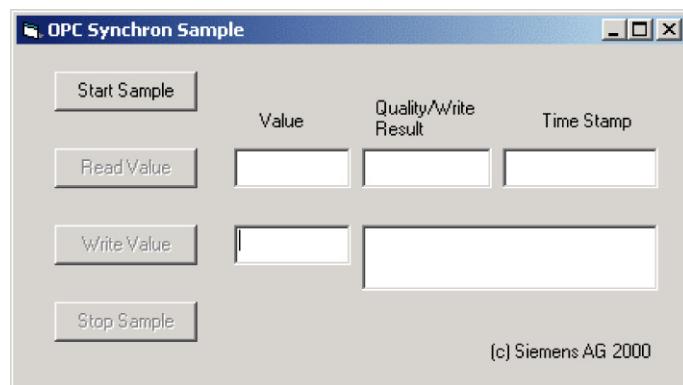


Bild 6-1 Dialog nach dem Start des Beispielprogramms für die OPC-Automation-Schnittstelle

Nach dem Betätigen der Schaltfläche "Start Sample" erzeugt das Programm die notwendigen OPC-Objekte. Danach können auch die anderen Schaltflächen benutzt werden.

Nach dem Drücken der Schaltfläche "Read Value" erscheinen die gelesenen Werte in den entsprechenden Textfeldern:

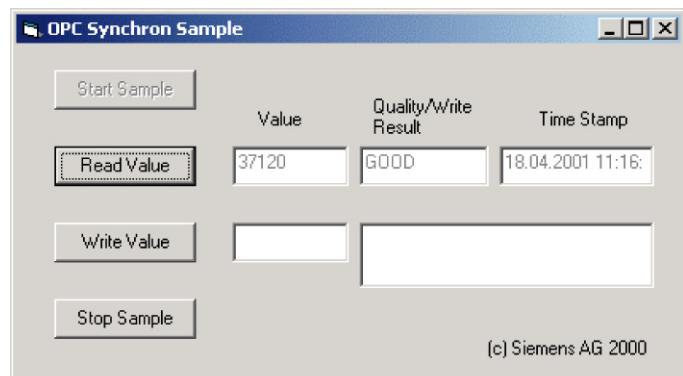


Bild 6-2 Dialog nach Betätigen der Schaltfläche "Read Value"

Im Ausgangszustand ist das Beispielprogramm für den Betrieb mit einer Demo-Verbindung

ausgelegt. Soll das Beispielprogramm in einer realen Umgebung laufen, müssen Sie die folgende Zeile im Programmcode ändern:

```
Set ItemObj = GroupObj.OPCItems.AddItem("S7:[DEMO]MW1", 1)
```

Beachten Sie bitte auch einige Beispiele und das Kapitel "OPC-Prozessvariablen für SIMATIC NET (Seite 21)" bezüglich des Aufbaus der ItemIDs.

Um einen Wert zu schreiben, muss dieser in das entsprechende Textfeld eingetragen werden. Nach dem Betätigen der Schaltfläche "Write Value" gibt das Programm eine Meldung über das Ergebnis dieser Operation aus:

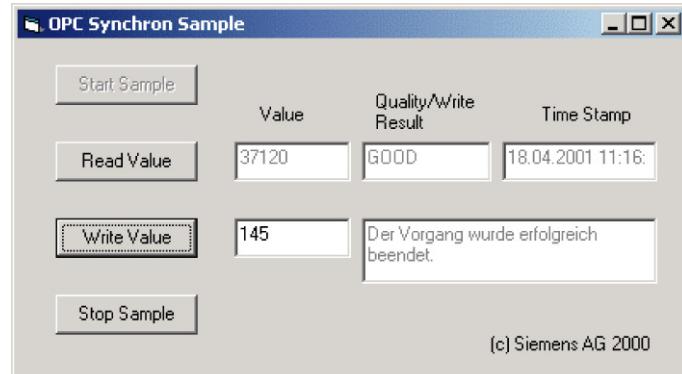


Bild 6-3 Dialog nach Schreiben eines Werts in das Textfeld und Betätigung der Schaltfläche "Write Value"

Mit der Schaltfläche "Stop Sample" wird das Programm beendet, d.h. alle OPC-Objekte werden abgebaut und die zugehörigen Ressourcen werden freigegeben.

### 6.1.3 Beschreibung des Programmablaufs

Bedingt durch das Klassenmodell von OPC muss bei Methodenaufrufen von OPC-Objekten eine bestimmte Reihenfolge eingehalten werden. Um eine Instanz der Klasse *OPCItem* erzeugen zu können, ist ein Objekt der Klasse *OPCGroup* nötig. Das kann aber erst geschehen, wenn eine Instanz der Klasse *OPCServer* vorhanden ist und eine Verbindung zu diesem Server hergestellt wurde.

Die grundsätzliche Abfolge von Befehlen zum Erzeugen und Löschen von OPC-Objekten wird in der folgenden Grafik dargestellt. Dabei werden die Variablennamen des Beispielprogramms verwendet.

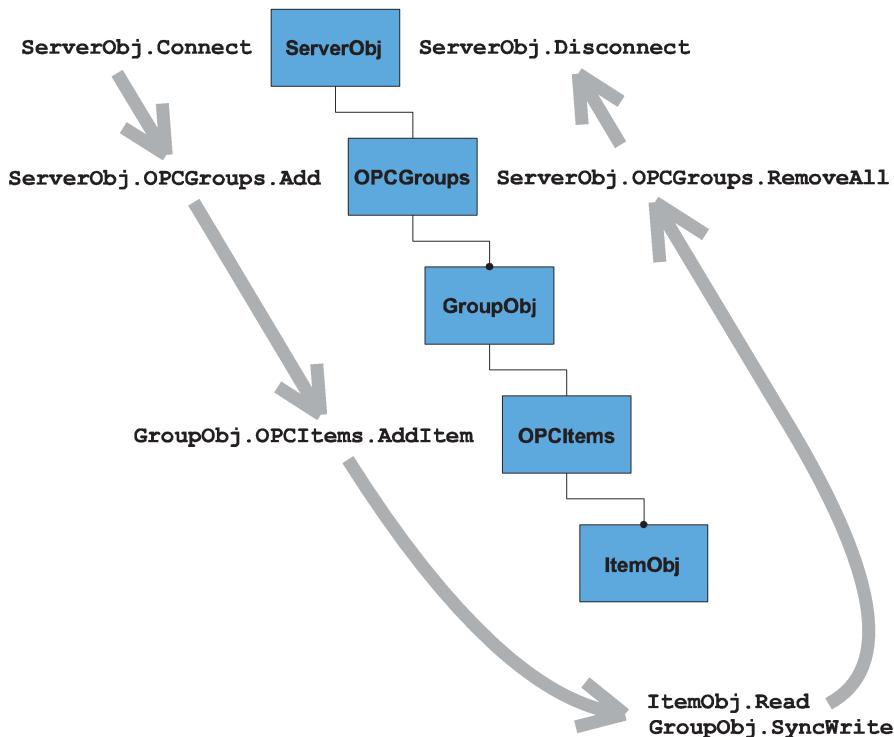


Bild 6-4 Abfolge von Befehlen zum Erzeugen und Löschen von OPC-Objekten

Das Beispielprogramm enthält alle Bestandteile, die in einer typischen Client-Anwendung vorkommen. Dazu gehören der Verbindungsaufbau zum OPC-Server, das Einrichten einer Gruppe mit Variablen sowie das Lesen und Schreiben von Werten für ein Item.

#### 6.1.4 Nutzung der OPC-Automation-Schnittstelle mit dem .NET-Framework

##### Einleitung

Dieser Abschnitt beschreibt die Nutzung und Migration des Beispielprogramms zum synchronen Lesen und Schreiben über die OPC-Automation-Schnittstelle unter dem .NET-Framework für Visual Basic.NET.

Nach der Installation der SIMATIC NET-Software finden Sie dieses Programm auf Ihrer Festplatte in folgendem Verzeichnis:  
"<Installationspfad>\SIEMENS\SIMATIC.NET\opc2\samples\automation\sync.net"

Die Bedienung und der Ablauf des Beispiels bleiben gegenüber dem vorangegangenen Beispiel unverändert. Das folgende Kapitel beschreibt nur die notwendigen Änderungen.

## Voraussetzungen für die Verwendung des Beispielprogramms

Auf dem Rechner, auf dem das Beispielprogramm ablaufen soll, muss ein .NET-Framework ab Version 4.0 installiert sein.

Im vorhandenen VB.NET Projekt wurde bereits die COM-Komponente "Siemens OPC DAAutomation 2.0" hinzugefügt:

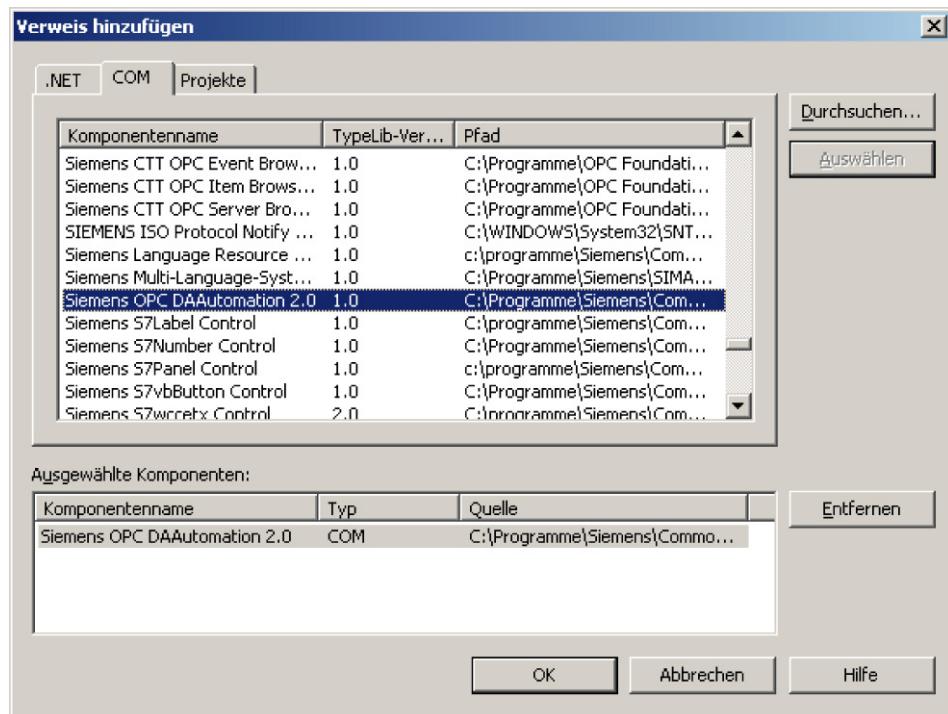


Bild 6-5 Hinzufügen der COM-Komponente "Siemens OPC DAAutomation 2.0" im VB.NET Projekt

### Mit dem Befehl

```
Imports OPCSiemensDAAutomation
```

kann auf einfache Weise der Namensraum und Methoden der OPC-Automation-Schnittstelle unter dem .NET Framework genutzt werden.

## Herstellen einer Verbindung zum OPC-Server, das Hinzufügen einer Gruppe und eines Items

Mittels der Methoden *Connect*, *Add* und *AddItem* wird analog dem bisherigen Beispiel verfahren:

```
ServerObj = New OPCServer
ServerObj.Connect(("OPC.SimaticNET"))

GroupsObj = ServerObj.OPCGroups
GroupObj = GroupsObj.Add("MyOPCGroup")

ItemObj = GroupObj.OPCItems.AddItem("S7:[DEMO]MW1", 1)
```

## Synchron lesen

Um Werte synchron zu lesen, wird die Methode *Read* der Klasse OPCItem aufgerufen:

```
ItemObj.Read(OPCDevice, myValue, myQuality, myTimeStamp)
```

## Synchron schreiben

### Hinweis

Visual Basic .NET verwendet als minimalen Arrayindex den Wert 0 und nicht den Wert 1, wie das bisherige Visual Basic und die darauf angepasste OPC-Automation-Schnittstelle.

Feldobjekte des importierten Namensraums *OPCSiemensDAAutomation* beginnen also mit dem Index 0. Im Beispiel werden deshalb spezielle Felder für die Übergabewerte mit Feldgrenzen 1 angelegt:

```
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)
```

Diese Variablen müssen noch initialisiert werden. Das Server-Handle hat seinen Wert 1 beim Hinzufügen des Items erhalten:

```
Serverhandles.SetValue(ItemObj.ServerHandle, 1)
MyErrors.SetValue(0, 1)
```

Dann wird noch der Wert der Text-Eigenschaft des Textfeldes für die Benutzereingabe der ersten Komponente des Feldes *MyValues* zugewiesen:

```
MyValues.SetValue(Edit_WriteVal.Text, 1)
```

Nachdem alle notwendigen lokalen Variablen deklariert und initialisiert wurden, wird die Methode *SyncWrite* der Klasse OPCGroup aufgerufen:

```
GroupObj.SyncWrite(1, Serverhandles, MyValues, MyErrors)
```

## 6.2 OPC-Custom-Schnittstelle (Synchrone Kommunikation) in C++

Das vorliegende Beispiel in Visual C++ nutzt unter Verwendung der Microsoft Foundation Classes (MFC) die Custom-Schnittstelle für Data Access V2.0 von OPC um Daten synchron zu lesen und zu schreiben.

Diese Beschreibung gliedert sich in folgende Abschnitte:

- Aktivieren der Simulationsverbindung
- Bedienung des Beispielprogramms
- Beschreibung des Programmablaufs
- Programmbeschreibung
- Hinweise zum Erstellen eigener Programme

### 6.2.1 Aktivieren der Simulationsverbindung

Damit das vorliegende Programm überhaupt lauffähig ist, müssen Sie eine Simulationsverbindung aktivieren, die die im Programm verwendeten Demo-Variablen verfügbar macht. Gehen Sie dazu folgendermaßen vor:

1. Starten Sie das Konfigurationsprogramm "Kommunikations-Einstellungen" über das Startmenü:  
"Start" > "Alle Programme" > "Siemens Automation" > "SIMATIC" > "SIMATIC NET" > "Kommunikations-Einstellungen".  
Reaktion: Das Konfigurationsprogramm "Kommunikations-Einstellungen" wird geöffnet.
2. Öffnen Sie im linken Navigationsfenster den Eintrag "OPC-Protokollauswahl":  
"SIMATIC NET-Konfiguration" > "OPC-Einstellungen" > "OPC-Protokollauswahl".
3. Klicken Sie auf das Pfeilsymbol neben dem S7-Protokoll und aktivieren Sie das Optionskästchen "virtuelle Baugruppe (DEMO) für die Simulation bereitstellen".
4. Bestätigen Sie mit "Übernehmen".
5. Beenden Sie das Konfigurationsprogramm "Kommunikations-Einstellungen".

---

#### Hinweis

Damit die Änderungen wirksam werden, müssen zuvor alle OPC-Clients beendet werden!

---

## 6.2.2

### Bedienung des Beispielprogramms

Beim Programmstart ist nur die Schaltfläche "Start Sample" aktiviert. Nach dem Betätigen dieser Schaltfläche erzeugt das Programm die notwendigen OPC-Objekte. Danach können auch die anderen Schaltflächen benutzt werden:

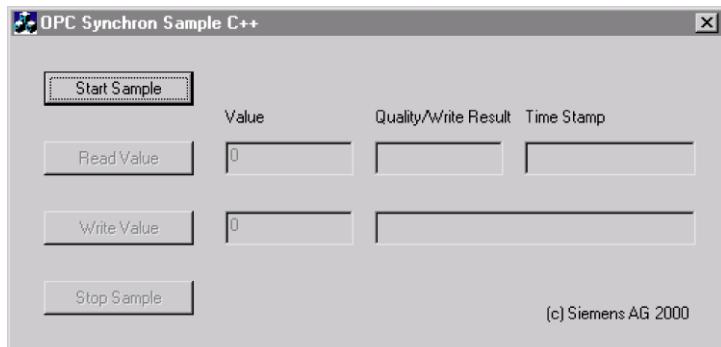


Bild 6-6 Start-Dialog nach Starten des Beispielprogramms für die OPC-Custom-Schnittstelle

Nach dem Drücken der Schaltfläche "Read Value" erscheinen die gelesenen Werte in den entsprechenden Textfeldern. Im Ausgangszustand ist das Beispielprogramm für den Betrieb mit einer Demo-Verbindung ausgelegt. Soll das Beispielprogramm in einer realen Umgebung laufen, müssen Sie die folgende Zeile in der Datei "OpcSyncDlg.cpp" ändern:

```
LPWSTR szItemID = L"S7:[DEMO]MW1";
```

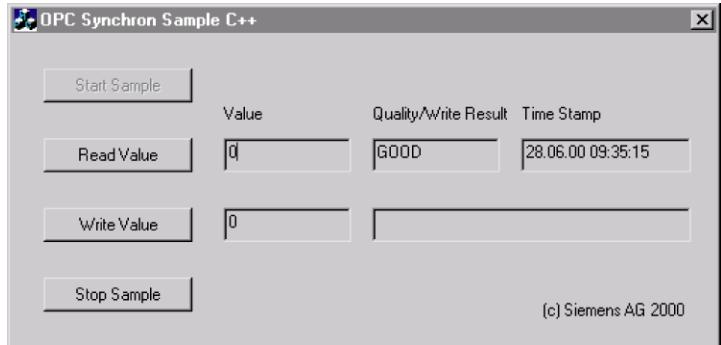


Bild 6-7 Anzeige der gelesenen Werte nach Betätigung der Schaltfläche "Read Value"

Um einen Wert zu schreiben, muss dieser in das entsprechende Textfeld eingetragen werden. Nach dem Betätigen der Schaltfläche "Write Value" gibt das Programm eine Meldung über das Ergebnis dieser Operation aus:

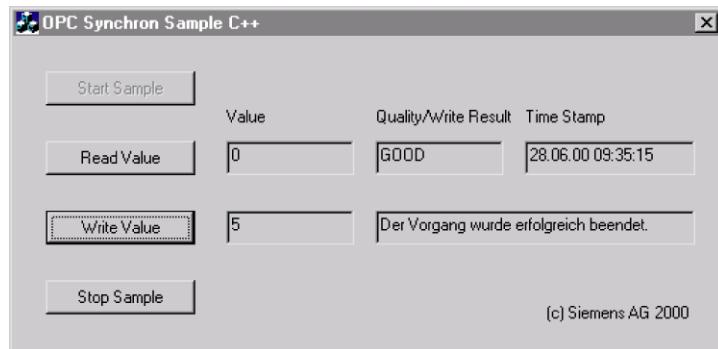


Bild 6-8 Anzeige der Ergebnisse nach Betätigung der Schaltfläche "Write Value"

Mit der Schaltfläche "Stop Sample" wird das Programm beendet, d.h. alle OPC-Objekte werden abgebaut und die zugehörigen Ressourcen werden freigegeben.

### 6.2.3 Beschreibung des Programmablaufs

Bedingt durch das Klassenmodell von OPC muss bei Methodenaufrufen von OPC-Objekten eine bestimmte Reihenfolge eingehalten werden. Um eine Instanz der Klasse *OPCItem* erzeugen zu können, ist ein Objekt der Klasse *OPCGroup* nötig. Das kann aber erst geschehen, wenn eine Instanz der Klasse *OPCServer* vorhanden ist und eine Verbindung zu diesem Server hergestellt wurde.

Die grundsätzliche Abfolge von Befehlen zum Erzeugen und Löschen von OPC-Objekten wird in der folgenden Grafik dargestellt. Dabei werden die Variablennamen des Beispielprogramms verwendet.

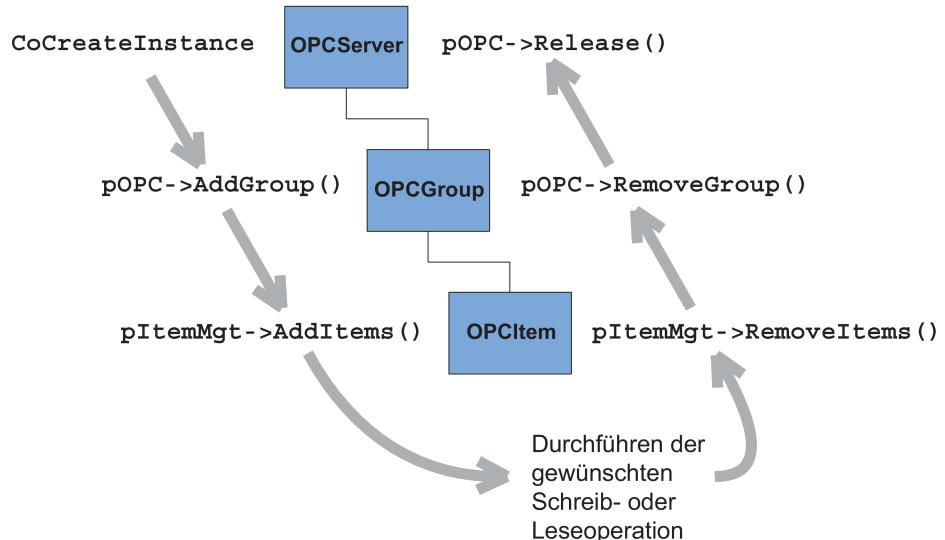


Bild 6-9 Abfolge von Befehlen zum Erzeugen und Löschen von OPC-Objekten

Das Beispielprogramm enthält alle Bestandteile, die in einer typischen Client-Anwendung vorkommen. Dazu gehören der Verbindungsauflauf zum OPC-Server, das Einrichten einer Gruppe mit Variablen sowie das Lesen und Schreiben von Werten für ein Item. Vor der Detailbeschreibung des Quellcodes soll deshalb zunächst der grundlegende Aufbau einer OPC-Anwendung dargestellt werden:

Schritt	Beschreibung
1	Anmelden bei COM
2	Konvertierung der ProgID in eine CLSID
3	Verbindungsauflauf zum OPC-Server
4	Erzeugen einer OPC-Gruppe
5	Hinzufügen von Items
6	Anfordern eines Schnittstellenzeigers für IOPCSyncIO
7	Objekte löschen und Speicher freigeben

### Schritt 1: Anmelden bei COM

Jedes Programm, das Funktionen der COM-Library aufrufen möchte, muss sich zuerst bei COM anmelden. Dafür gibt es die Funktion *CoInitialize()*:

```
HRESULT r1;
r1 = CoInitialize(NULL);
```

### Schritt 2: Konvertierung der ProgID in eine CLSID

Jeder COM-Server besitzt zur Identifizierung eine sogenannte *ProgID*, der eine weltweit eindeutige *CLSID* (128 Bit-Code) zugeordnet ist. Da für die im Folgenden benötigten Funktionen die *CLSID* als Parameter verlangt wird, muss diese mit der Funktion *CLSIDFromProgID()* aus der ProgID ermittelt werden. Die *ProgID* des OPC-Servers von SIMATIC NET ist L"OPC.SimaticNET":

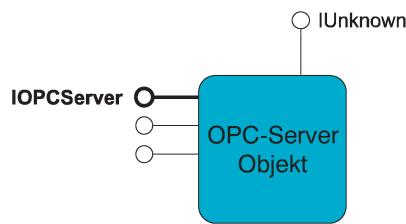
```
r1 = CLSIDFromProgID(L"OPC.SimaticNET", &clsid);
```

### Schritt 3: Verbindungsauflauf zum OPC-Server

Die Funktion *CoCreateInstance()* erzeugt eine Instanz der Klasse, deren *CLSID* vorgegeben wurde:

```
r1 = CoCreateInstance(clsid, NULL, CLSCTX_LOCAL_SERVER ,
IID_IOPCServer, (void**)&m_pIOPCServer);
```

Ergebnis dieses Programmabschnitts ist ein Objekt der Klasse OPC-Server. Außerdem liefert *CoCreateInstance* einen Zeiger auf das Interface *IOPCServer* des Serverobjekts (Variable *m\_pIOPCServer*):

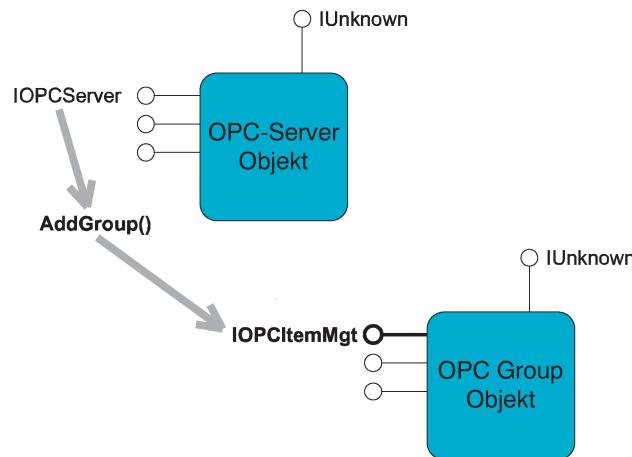
Bild 6-10 Objekt der Klasse OPC-Server mit Zeiger auf die Schnittstelle *IOPCServer*

#### Schritt 4: Erzeugen einer OPC-Gruppe

Die Schnittstelle *IOPCServer* besitzt die Methode *AddGroup()* zum Anlegen von Gruppen:

```
r1 = m_pIOPCServer->AddGroup(L"grp1", TRUE, 500, 1,
                                &TimeBias, &PercentDeadband, LOCALE_ID,
                                &m_GrpSrvHandle, &RevisedUpdateRate,
                                IID_IOPCItemMgt,
                                (LPUNKNOWN*)&m_pIOPCItemMgt);
```

Ergebnis dieses Programmschritts ist eine Gruppe mit dem vorgegebenen Namen und den gewünschten Eigenschaften. Außerdem ist als Rückgabeparameter ein Zeiger auf die angeforderte Schnittstelle des Gruppenobjekts, in diesem Fall *IOPCItemMgt* (Variable *m\_pIOPCItemMgt*), vorhanden:

Bild 6-11 Erzeugen einer OPC-Gruppe mit Zeiger auf die angeforderte Schnittstelle *IOPCItemMgt* des Gruppenobjekts

#### Schritt 5: Hinzufügen von Items

Die Schnittstelle *IOPCItemMgt* besitzt die Methode *AddItems()* zum Anlegen von OPC-Items:

```
r1 = pItemMgt->AddItems(1, m_Items, &m_pItemResult, &m_pErrors);
```

Ergebnis dieses Programmschritts ist die gewünschte Anzahl Items mit den vorgegebenen Eigenschaften. Außerdem werden die Variablen der Ergebnisstruktur *m\_pItemResult* (Serverhandle, Datentyp des Items auf dem Zielsystem usw.) mit Werten belegt.

### Schritt 6: Anfordern eines Schnittstellenzeigers für IOPCSyncIO

Ein Zeiger auf die Schnittstelle *IOPCSyncIO* ist für die Nutzung von Methoden zum synchronen Lesen und Schreiben von Werten notwendig. Er wird mit Hilfe des bereits vorhandenen Zeigers auf die Schnittstelle *IOPCItemMgt* angefordert:

```
r1 = m_pIOPCItemMgt->QueryInterface(IID_IOPCSyncIO,
                                         (void**)&m_pIOPCSyncIO);
```

Mit den Methoden *Read()* und *Write()* dieser Schnittstelle können Werte für Items gelesen bzw. geschrieben werden:

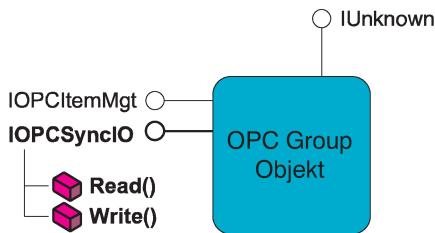


Bild 6-12 OPC-Group-Objekt mit Zeiger auf die Schnittstelle *IOPCSyncIO* und deren Methoden *Read()* und *Write()*

### Schritt 7: Objekte löschen und Speicher freigeben

Vor dem Beenden des Programms müssen die erzeugten OPC-Objekte gelöscht und der angeforderte Speicher freigegeben werden. Die entsprechenden Funktionen sind Bestandteil der bisher benutzten Schnittstellen:

```
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 Programmbeschreibung OPCDA\_SyncDlg.cpp

Die Datei "OPCDA\_SyncDlg.cpp" kann in folgende Abschnitte untergliedert werden:

- Deklaration globaler Variablen
- Methoden der Klasse COPCDA\_SyncDlg

### Globale Variable

Am Anfang dieses Moduls wird eine globale Variable mit der Item-Bezeichnung initialisiert. Diese Variable muss, um die Funktionalität des Beispielprogramms zu gewährleisten, les- und schreibbar sein.

```
const LPWSTR szItemID = L"S7:[DEMO]MW1";
```

Beachten Sie bitte auch einige Beispiele und das Handbuch bezüglich des Aufbaus der ItemIDs.

### Methoden der Klasse COPCDA\_SyncDlg

Hauptbestandteil dieses Moduls sind die Ereignisprozeduren für die einzelnen Schaltflächen des Hauptdialogfelds, die von der MFC aufgerufen werden. In der Methode *OnInitDialog* werden außerdem noch einige Voreinstellungen vorgenommen:

- OnInitDialog
- OnStart
- OnRead
- OnWrite
- OnStop
- DestroyWindow

#### 6.2.4.1 OnInitDialog

Für den Zugriff auf COM-Mechanismen sind mehrere Klassenvariablen vorgesehen, die beim Initialisieren des Hauptdialogfelds mit Anfangswerten belegt werden. Für diese Variablen ist folgende Verwendung vorgesehen:

- m\_pIOPCServer ist ein Zeiger auf die Schnittstelle IOPCServer der Klasse OPC-Server.
- m\_pIOPCItemMgt ist ein Zeiger auf die Schnittstelle IOPCItemMgt der Klasse OPC-Group.
- m\_pOPCSyncIO ist ein Zeiger auf die Schnittstelle IOPCSyncIO der Klasse OPC-Group.

```
m_pIOPCServer = NULL;
m_pIOPCItemMgt = NULL;
m_pOPCSyncIO = NULL;
```

Zum Programmstart kann der Benutzer nur die Schaltfläche "Start Sample" betätigen. Alle anderen Schaltflächen sind deaktiviert. So wird sichergestellt, dass vor dem Durchführen von Schreib- oder Leseoperationen alle notwendigen OPC-Objekte korrekt aufgebaut werden. Dies wird durch folgenden Programmabschnitt erreicht:

```
m_CtrlStop.EnableWindow(FALSE);  
m_CtrlRead.EnableWindow(FALSE);  
m_CtrlWrite.EnableWindow(FALSE);
```

### 6.2.4.2 OnStart

*OnStart* baut die für den Programmablauf notwendigen OPC-Objekte auf. Zunächst werden lokale Variablen deklariert und initialisiert, die für die Methoden der OPC-Objekte benötigt werden:

```
void COpcSyncDlg::OnStart()  
{    HRESULT r1;  
    CLSID clsid;  
    LONG TimeBias = 0;  
    FLOAT PercentDeadband = 0.0;  
    DWORD RevisedUpdateRate;  
    LPWSTR ErrorStr;  
    char str[100];  
    Cstring szErrorText;
```

Danach werden folgende Operationen durchgeführt:

### CoInitialize

*CoInitialize()* initialisiert die COM-Bibliothek. Der Eingangsparameter muss immer *NULL* sein.

```
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;  
        }  
}
```

### CLSIDFromProgID

*CLSIDFromProgID()* ermittelt aus einer vorgegebenen ProgID einen weltweit eindeutigen Klassenbezeichner (den sogenannten *Class Identifier* oder auch *CLSID*). Dieser wird für die Funktion *CoCreateInstance* benötigt.

Diese Methode ist wie folgt deklariert:

```
HRESULT CLSIDFromProgID(LPCOLESTR lpszProgID, LPCLSID pclsid);
```

Parameter	Beschreibung
lpszProgID	Zeiger auf die ProgID [Eingangsparameter]
pclsid	Zeiger auf den ermittelten <i>Class Identifier</i> [Rückgabewert]

Im Beispielprogramm ermittelt diese Methode den Class Identifier des OPC-Servers von SIMATIC NET:

```
r1 = CLSIDFromProgID(L"OPC.SimaticNET", &clsid);
if (r1 != S_OK)
{
    MessageBox("Retrival of CLSID failed",
              "Error CLSIDFromProgID ()",
              MB_OK+MB_ICONERROR);
    CoUninitialize();
    SendMessage(WM_CLOSE);
    return;
}
```

## CoCreateInstance

*CoCreateInstance()* erzeugt eine Instanz der Klasse, deren Klassenbezeichner vorgegeben wurde. Diese Methode ist wie folgt deklariert:

```
STDAPI CoCreateInstance (REFCLSID rclsid,
                        LPUNKNOWN pUnkOuter,
                        DWORD dwClsContext,
                        REFIID riid,
                        LPVOID *ppv);
```

Parameter	Beschreibung
rclsid	Der Klassenbezeichner des gewünschten Objekts
pUnkOuter	NULL-Zeiger, wenn das zu erzeugende Objekt nicht Teil eines aggregierten Objekts ist. Ein Zeiger, der nicht NULL-Zeiger ist, wird als Zeiger auf die IUnknown-Schnittstelle des aggregierten Objekts interpretiert.
dwClsContext	Beschreibt die Ablaufumgebung des zu erzeugenden Objekts. Es wird damit festgelegt, ob der ausführbare Code, der Objekte dieser Klasse erzeugt und verwaltet, auf der lokalen Maschine läuft und ob dafür ein eigener Prozess erzeugt wird. Die hierfür verwendbaren Konstanten sind im Aufzählungstyp CLSCTX festgelegt. Im Beispiel wird CLSCTX_LOCAL_SERVER benutzt, d. h. der ausführbare Code für die Verwaltung der Server-Objekte läuft auf dem gleichen Rechner wie das Beispielprogramm, aber in einem eigenen Prozess.
riid	Bezeichnung der Schnittstelle, über die mit dem Objekt kommuniziert werden soll, im Beispielprogramm <i>IOPCServer</i> .
ppv	Adresse der Zeigervariable, die bei erfolgreichem Methodenaufruf den gewünschten Schnittstellenzeiger enthält.

Der folgende Methodenaufruf erzeugt ein Objekt der Klasse OPC-Server und liefert einen Zeiger auf die Schnittstelle IOPCServer zurück:

```
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

Die Methode *AddGroup()* der Schnittstelle IOPCServer erzeugt eine OPC-Gruppe im Server und ist wie folgt deklariert:

```
HRESULT AddGroup (LPWCSTR szName,  
                  BOOL bActive,  
                  DWORD dwRequestedUpdateRate,  
                  OPCHANDLE hClientGroup,  
                  LONG *pTimeBias,  
                  FLOAT *pPercentDeadband,  
                  DWORD dwLCID,  
                  OPCHANDLE *phServerGroup,  
                  DWORD *pRevisedUpdateRate,  
                  REFIID riid,  
                  LPUNKNOWN *ppUnk);
```

Parameter	Beschreibung
szName	Gruppenname, kann vom Client beliebig vorgegeben werden, jedoch muss innerhalb des Client Eindeutigkeit gewährleistet sein.
bActive	FALSE, wenn die Gruppe zum Zeitpunkt des Erzeugens inaktiv sein soll. TRUE, wenn die Gruppe zum Zeitpunkt des Erzeugens aktiv sein soll.
dwRequestedUpdateRate	Gibt das kürzeste Zeitintervall an, nach dem ein Client über Änderungen der Werte oder Zustände von Items benachrichtigt werden soll. Mit der Festlegung einer geeigneten Zeitdauer wird verhindert, dass einem Client diese Informationen schneller zugesendet werden als er sie verarbeiten kann.
hClientGroup	Vom Client beliebig wählbare Kennzahl, die vom Server bei bestimmten Benachrichtigungen wieder zurückgeliefert wird. Der Client kann dadurch einen Bezug zu seinen Daten herstellen. Mit diesem Handle identifiziert der Client die Gruppe.
pTimeBias	Abweichung der Serverzeit von UTC (Universal Time Convention)

Parameter	Beschreibung
pPercentDeadband	Gibt den Prozentsatz der Bandbreite an, in der eine Wertänderungen nicht zu einer Benachrichtigung führen. Es muss die Ober- und Untergrenze eines Wertes bekannt sein, um die Bandbreite zu bestimmen. Das ist in diesem Beispielprogramm nicht der Fall.
dwLCID	Auswahl der Sprache, die vom Server verwendet wird, wenn Texte zurückgegeben werden
phServerGroup	Vom Server vergebene Handle, das bei manchen Funktionsaufrufen als Parameter angegeben werden muss (z. B. <i>RemoveGroup</i> ). Der Server benötigt es um diese Gruppe zu identifizieren.
pRevisedUpdateRate	Vom Server zurückgeliefertes kürzestes Zeitintervall nach dem ein Client über Änderungen der Werte oder Zustände von Items benachrichtigt wird.
riid	Zeiger auf den Bezeichner einer Schnittstellen des OPC-Group-Objekts, die nach dem Erzeugen der Gruppe zur Verfügung stehen soll. Dieser Parameter erspart einen zusätzlichen Aufruf der Methode <i>QueryInterface</i> .
ppUnk	Zeiger auf die gewünschte Schnittstelle.

Beim Aufruf von *AddGroup* im Beispielprogramm werden folgende Vorgaben gemacht:

- Der Parameter *bActive* wird auf den Wert FALSE gesetzt. Direkt nach dem Erstellen ist die Gruppe inaktiv, d. h. für diese Gruppe werden keine OnDataChange-Callbacks erzeugt.
- Die Dauer des gewünschten Aktualisierungintervalls beträgt 500 Millisekunden.
- Das Client-Handle kann beliebig angegeben werden, da nur eine Gruppe verwendet wird.
- Die Voraussetzungen, bei Wertänderungen innerhalb eines bestimmten Bereichs eine Benachrichtigung zu unterdrücken, sind im Beispielprogramm nicht gegeben. Deshalb wird für den Parameter *pPercentDeadband* der Wert "0.0" eingetragen.
- Als Rückgabewert soll *AddGroup* einen Zeiger auf die Schnittstelle *IOPCItemMgt* liefern.

```
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

Die Methode *AddItems()* der Schnittstelle *IOPCItemMgt* erzeugt OPC-Items und ist wie folgt deklariert:

```

HRESULT AddItems (DWORD dwNumItems,
                  OPCITEMDEF *pItemArray,
                  OPCITEMRESULT **ppAddResults,
                  HRESULT **ppErrors);

```

Parameter	Beschreibung
dwNumItems	Anzahl der Items, die eingefügt werden sollen
pItemArray	Array mit Elementen vom Typ OPCITEMDEF. Strukturvariablen dieses Typs beinhalten alle Informationen, die vom Server für die Erzeugung von Items benötigt werden.
ppAddResults	Array mit Elementen vom Typ OPCITEMRESULT. Die Rückgabewerte liefert der OPC-Server als Strukturvariablen dieses Typs.
ppErrors	Array mit Elementen vom Typ HRESULT. Diese Variablen liefern einen Fehlercode, wenn Items nicht erfolgreich erzeugt werden konnten bzw. eine Information über den erfolgreichen Methodenaufruf.

Vor dem Aufruf von *AddItems* muss erst ein Array mit Elementen vom Typ OPCITEMDEF erzeugt und mit gültigen Werten belegt werden. Dabei werden folgende Gegebenheiten berücksichtigt:

- Ein Zugriffspfad für das Item ist im Beispiel nicht notwendig, deshalb wird hier ein Leerstring angegeben.
- Die Item-Bezeichnung wurde am Anfang des Moduls OPCDA\_SyncDlg.cpp festgelegt und der Variablen szItemID zugewiesen.
- Das Item soll nach der Erzeugung aktiviert sein.
- Das Beispielprogramm verwendet ein Client-Handle von 1.
- Der OPC-Server für SIMATIC NET benötigt keine *BinaryLargeObjects*, deshalb erhält die Strukturkomponente dwBlobSize den Wert "0".
- Für das Item soll der Server den Rückgabewert in einem Typ zurückliefern, der dem ursprünglichen Datentyp des Items entspricht.

```

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;
```

Der Zeiger *m\_pItemResult* ist als Eigenschaft der Klasse *COPCDA\_SyncDlg* vorhanden. Über diese Variable kann auf die vom Server zurückgelieferten Ergebnisse zugegriffen werden. *m\_pErrors* ist ein Zeiger auf den Fehlercode:

```
r1 = m_pIOPCItemMgt->AddItems(1, m_Items, &m_pItemResult, &m_pErrors);
```

Wenn der Aufruf von *AddItems* nicht erfolgreich war, wird das Programm abgebrochen. Vorher muss das Programm allerdings noch die verwendeten Ressourcen freigegeben:

```
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

Wenn der Rückgabewert von *AddItems()* das Auftreten eines Fehlers anzeigt, liefert die Methode *GetErrorString()* der Schnittstelle *IOPCServer* die zugehörige Fehlermeldung. Diese Methode ist wie folgt deklariert:

```
HRESULT GetErrorString (HRESULT dwError,
                        LCID dwLocale,
                        LPWSTR *ppString);
```

Parameter	Beschreibung
dwError	Vom Server zurückgelieferter Fehlercode
dwLocale	Die Sprachkennung für die Fehlermeldung
ppString	Doppelzeiger auf einen nullterminierten String, in den <i>GetErrorString</i> die Fehlermeldung zurückliefert

```
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

Die Methode `QueryInterface()` der Schnittstelle `IUnknown` liefert einen Zeiger auf eine Schnittstelle, deren Bezeichner als Eingangsparameter vorgegeben wird. `QueryInterface()` ist wie folgt deklariert:

```
HRESULT QueryInterface (REFIID iid, void **ppvObject);
```

Parameter	Beschreibung
iid	Bezeichner der gewünschten Schnittstelle
ppvObjekt	Adresse der Zeigervariable, die bei erfolgreichem Methodenaufruf den gewünschten Schnittstellenzeiger enthält. Wenn das Objekt diese Schnittstelle nicht unterstützt, wird ein Fehlercode und der NULL-Zeiger zurückgegeben.

`QueryInterface` ermittelt einen Zeiger auf die Schnittstelle `IOPCSyncIO`, die Methoden zum synchronen Lesen und Schreiben bereitstellt:

```
r1 = m_pIOPCItemMgt->QueryInterface (IID_IOPCSyncIO,
                                         (void**)&m_pIOPCSyncIO);
if (r1 < 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;
}
```

## Schaltflächen aktivieren

Nachdem `OnButtonStart()` alle notwendigen OPC-Objekte aufgebaut hat, deaktiviert es die Schaltfläche "Start Sample". Alle anderen Schaltflächen werden aktiviert. Durch diese Vorgehensweise ist sichergestellt, dass `OnButtonStart()` nur einmal durchlaufen wird. So kann auf zusätzliche Abfragen im Programm verzichtet werden.

```
m_CtrlStop.EnableWindow (TRUE);
m_CtrlRead.EnableWindow (TRUE);
m_CtrlWrite.EnableWindow (TRUE);
m_CtrlStart.EnableWindow (FALSE);
```

## Die Struktur OPCITEMDEF

OPCITEMDEF hat folgenden Aufbau:

```
typedef struct {
    LPWSTR szAccessPath;
```

```

    LPWSTR szItemID;
    BOOL bActive;
    OPCHANDLE hClient;
    DWORD dwBlobSize;
    BYTE *pBlob;
    VARTYP vtRequestedDataType;
    Word wReserved;
} OPCITEMDEF;

```

### Variablen von OPCITEMDEF

Variable	Beschreibung
szAccessPath	optionaler Zugriffspfad für die Items, wird von SIMATIC NET nicht benötigt
szItemID	Vom Client vergebene ItemID
bActive	TRUE, wenn bei Wertänderungen des Items in einer aktiven Gruppe eine Benachrichtigung des Client erfolgen soll; FALSE, wenn eine solche Benachrichtigung nicht stattfinden soll.
hClient	Vom Client vergebene Handle für ein Item. Das Client-Handle übergibt der Server bei Aufrufen an den Client (z.B. <i>OnDataChange</i> ), damit der Client auf die betroffenen Variablen in seinen Strukturen zugreifen kann.
dwBlobSize	Größe eines Speicherbereichs im Server, in dem Zusatzinformationen für einen schnelleren Zugriff auf die Daten eines Items abgelegt werden.
pBlob	Zeiger auf den zuvor beschriebenen Speicherbereich
vtRequestedDataType	Vom Client gewünschter Datentyp

### Die Struktur OPCITEMRESULT

OPCITEMRESULT hat folgenden Aufbau:

```

typedef struct {
    OPCHANDLE hServer;
    VARTYPE vtCanonicalDataType;
    WORD wReserved;
    DWORD dwAccessRights;
    DWORD dwBlobSize;
    BYTE *pBlob;
} OPCITEMRESULT;

```

### Variablen von OPCITEMRESULT

Variable	Beschreibung
hServer	Vom Server vergebene Handle für ein Item. Das Server-Handle übergibt der Client bei Aufrufen an den Server, damit der Server auf die betroffenen Variablen in seinen Strukturen zugreifen kann.
vtCanonicalDataType	Der Datentyp, der vom Server für ein Item verwendet wird

Variable	Beschreibung
dwAccessRights	Information, ob auf ein Item nur lesend, nur schreibend oder lesend und schreibend zugegriffen werden kann.
dwBlobSize	Größe eines Speicherbereichs im Server, in dem Zusatzinformationen für einen schnelleren Zugriff auf die Daten eines Items abgelegt werden.
pBlob	Zeiger auf den zuvor beschriebenen Speicherbereich

#### 6.2.4.3 OnRead

*OnRead* führt einen synchronen Leseauftrag durch. Dazu werden folgende Operationen ausgeführt.

#### Variablen-deklaration

Zunächst werden einige lokale Variablen deklariert:

```
void COpcSyncDlg::OnRead()
{
    OPCHANDLE *phServer;
    OPCITEMSTATE *pItemValue;
    HRESULT *pErrors;
    HRESULT r1;
    LPWSTR ErrorStr;
    char str[100];
    UINT qnr;
```

Das Serverhandle des Items, für das der Wert gelesen werden soll, wird als Parameter für die Methode *Read()* benötigt. Diese vom Server vergebene Kennziffer ist in der Komponente *hServer* der globalen Variablen *pItemResult[0]* vorhanden. In diese Strukturvariable vom Typ OPCITEMRESULT legt die Methode *AddItem()* ihre Rückgabewerte ab.

```
phServer = new OPCHANDLE[1];
phServer[0] = m_pItemResult[0].hServer;
```

#### Read

*Read* liest synchron Werte für OPC-Items. Diese Methode ist wie folgt deklariert:

```
HRESULT Read (OPCDATASOURCE dwSource,
               DWORD dwNumItems,
               OPCHANDLE *phServer,
               OPCITEMSTATE **ppItemValues,
               HRESULT **ppErrors);
```

Parameter	Beschreibung
dwSource	Datenquelle. Bei Verwendung von OPC_DS_CACHE werden die Daten aus dem Cache des OPC-Servers gelesen, bei OPC_DS_DEVICE wird ein Leseauftrag über das Netz durchgeführt.
dwNumItems	Anzahl der Items, für die Werte gelesen werden.
phServer	Array mit Server-Handles.

Parameter	Beschreibung
pplItemValues	Array mit Elementen des Typs OPCITEMSTATE für die gelesenen Werte und Zusatzinformationen über den Lesevorgang.
ppErrors	Array mit Elementen vom Typ HRESULT. Diese Variablen liefern einen Fehlercode, wenn Read() nicht erfolgreich aufgerufen wurde bzw. eine Information über den erfolgreichen Methodenaufruf.

*Read* wird mit den zuvor initialisierten Variablen als Parameter aufgerufen:

```
r1 = m_pIOPCSyncIO->Read(OPC_DS_DEVICE, 1, phServer, &pItemValue,
                           &pErrors);
```

Die Methode *Read* stellt die gelesenen Werte und zusätzliche Informationen im Array *pItemValue* zur Verfügung. Bei erfolgreichem Ablauf des Lesevorgangs werden diese Daten in den entsprechenden Textfeldern des Hauptdialogfelds angezeigt:

```
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);
}
```

Der Rückgabewert *S\_FALSE* zeigt an, dass im Array *pErrors* ein Fehlercode gespeichert wurde. *GetErrorString* ermittelt die zugehörige Fehlermeldung, die in einem Dialogfeld ausgegeben wird:

```
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);
}
```

Wenn keine Fehlermeldungen vorliegen, wird bei einem erfolglosen Methodenaufruf ein passendes Dialogfeld angezeigt. Anschließend werden die von *Read* verwendeten Ressourcen freigegeben:

```
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;
```

### Variablen von OPCITEMSTATE

Variable	Beschreibung
hClient	Das Client-Handle des Items.
ftTimeStamp	Zeitangabe (Zeitpunkt, zu dem die Daten vom OPC-Server empfangen wurden).
wQuality	Information über die Integrität der Daten.
vDataValue	Der gelesene Wert für das Item.

### 6.2.4.4 OnWrite

OnWrite führt einen synchronen Schreibauftrag durch. Dazu werden folgende Operationen ausgeführt.

#### Variablen-deklaration

Zunächst werden einige lokale Variablen deklariert:

```
void COpcSyncDlg::OnWrite()
{
    OPCHANDLE *phServer;
    HRESULT *pErrors;
    VARIANT values[1];
    HRESULT r1;
    LPWSTR ErrorStr;
    CString szOut;
```

Das Serverhandle des Items, für das ein Wert geschrieben werden soll, wird als Parameter für die Methode *Write* benötigt. Diese vom Server vergebene Kennziffer ist in der Komponente *hServer* der globalen Variablen *m\_pItemResult[0]* vorhanden. In diese Strukturvariable vom Typ OPCITEMRESULT legt die Methode *AddItems* ihre Rückgabewerte ab:

```
phServer = new OPCHANDLE[1];
phServer[0] = m_pItemResult[0].hServer;
```

Die Variable *values* ist für die Speicherung des zu schreibenden Wertes vorgesehen. Die Methode *UpdateData* der Klasse *CWnd* überträgt den Inhalt aller Steuerelemente in die entsprechenden Member-Variablen. Damit wird die Komponente *iVa* der Strukturvariablen *values* initialisiert. Als Datentyp wird 2-Byte *Integer* vorgegeben:

```

UpdateData(TRUE);
values[0].vt = VT_I2;
values[0].iVal = m_WriteValue;

```

## Write

*Write* schreibt synchron Werte für OPC-Items. Diese Methode ist wie folgt deklariert:

```

HRESULT Write (DWORD dwNumItems,
               OPCHANDLE *phServer,
               VARIANT *pItemValues,
               HRESULT **ppErrors);

```

Parameter	Beschreibung
dwNumItems	Anzahl der Items, für die Werte geschrieben werden.
phServer	Array mit Serverhandles
pItemValues	Array mit den zu schreibenden Werten
ppErrors	Array mit Elementen vom Typ HRESULT. Diese Variablen liefern einen Fehlercode, wenn <i>Write()</i> nicht erfolgreich aufgerufen wurde bzw. eine Information über den erfolgreichen Methodenaufruf.

*Write* wird mit den zuvor initialisierten Variablen als Parameter aufgerufen:

```
r1 = m_pIOPCSyncIO->Write(1, phServer, values, &pErrors);
```

*GetErrorString* ermittelt die zu dem Rückgabewert *pErrors* gehörende Fehlermeldung, die der Member-Variablen *m\_WriteRes* zugewiesen wird. Allerdings ergänzt *GetErrorString* zwei Sonderzeichen für den Zeilenumbruch, die vor der Ausgabe in einem Textfeld entfernt werden müssen.

Anschließend werden die von *Write* verwendeten Ressourcen freigegeben:

```

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* baut die im Programm verwendeten OPC-Objekte ab und gibt die zugehörigen Ressourcen frei. Diese Methode wird aufgerufen, wenn die Schaltfläche "Stop Sample" betätigt, die Methode *OnDestroy* durchlaufen wird oder die Nachricht WM\_CLOSE gesendet wird (nach Betätigen der Schaltflächen zum Schließen des Dialogfeldes oder explizit durch Aufruf von *SendMessage*).

Die Freigabe der Ressourcen erfolgt in mehreren Schritten:

- Remove Items
- RemoveGroup
- Release
- CoUninitalize

### RemoveItems

Die Methode *RemoveItems* der Schnittstelle IOPCItemMgt löscht OPC-Items und ist wie folgt deklariert:

```
HRESULT RemoveItems (DWORD dwCount,
                     OPCHANDLE *phServer,
                     HRESULT **ppErrors);
```

Parameter	Beschreibung
dwCount	Anzahl der zu löschenen Items
phServer	Array mit den Server-Handles der zu entfernenden Items
ppErrors	Array mit Elementen vom Typ HRESULT. Diese Variablen liefern einen Fehlercode, wenn <i>RemoveItems()</i> nicht erfolgreich aufgerufen wurde bzw. eine Information über den erfolgreichen Methodenaufruf.

```
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_pIOPCServer->GetErrorString(pErrors[0], LOCALE_ID,
                                         &ErrorStr);
        sprintf(str, "%ls\n", ErrorStr);
        MessageBox(str, "Result RemoveItems()",
```

```

        MB_OK+MB_ICONEXCLAMATION);
CoTaskMemFree(ErrorStr);
}

```

## RemoveGroup

*RemoveGroup()* entfernt eine Gruppe aus dem Server und ist wie folgt deklariert:

```
HRESULT RemoveGroup (OPCHANDLE hServerGroup,
                     BOOL bForce);
```

Parameter	Beschreibung
hServerGroup	Serverhandle der Gruppe, die gelöscht werden soll.
bForce	legt fest, ob Gruppen auch dann gelöscht werden können, wenn noch eine Referenz darauf besteht.

```

r1=m_pIOPCServer->RemoveGroup(m_GrpSrvHandle, TRUE);
if (r1 != S_OK)
{
    MessageBox("RemoveGroup failed!",
               "Error Remove Group()",
               MB_OK+MB_ICONERROR);
}

```

## Release

Die Schnittstellen *IOPCServer* und *IMalloc* verfügen über die Methode *Release*, um die von der Schnittstelle belegten Ressourcen freizugeben:

```
m_pIOPCSyncIO->Release();
m_pIOPCSyncIO = NULL;
m_pIOPCItemMgt->Release();
m_pIOPCItemMgt = NULL;
m_pIOPCServer->Release();
m_pIOPCServer = NULL;
```

Außerdem müssen noch alle anderen angeforderten Ressourcen wieder freigegeben werden:

```
delete[] phServer;
CoTaskMemFree(pErrors);
CoTaskMemFree(m_pItemResult);
m_pItemResult=NULL;
CoTaskMemFree(m_pErrors);
m_pErrors = NULL;
m_GrpSrvHandle = NULL;
```

Alle Schaltflächen außer "Start Sample" werden inaktiv gesetzt:

```
m_CtrlStart.EnableWindow(TRUE);
m_CtrlRead.EnableWindow(FALSE);
m_CtrlStop.EnableWindow(FALSE);
m_CtrlWrite.EnableWindow(FALSE);
```

## CoUninitialize

Die Methode *CoUninitialize* schließt die COM-Bibliothek und gibt alle Ressourcen des entsprechenden Threads frei:

```
CoUninitialize();  
}
```

### 6.2.4.6 DestroyWindow

Diese Methode der Klasse CWnd wird überschrieben, damit beim Schließen des Dialogfensters auch alle "Aufräumarbeiten" durchgeführt werden. *OnButtonStop()* baut alle OPC-Objekte ab und gibt die zugehörigen Ressourcen frei. Danach wird *DestroyWindow* der Elternklasse aufgerufen, um den vom Fensterobjekt belegten Speicher freizugeben:

```
BOOL COPCDA_SyncDlg::DestroyWindow()  
{    if (m_pIOPCServer)  
    {        OnStop();  
        return CDialog::DestroyWindow();  
    }    }
```

### 6.2.4.7 GetQualityText

*GetQualityText* liefert zu einem vorgegebenen Fehlercode eine Fehlermeldung als CString-Objekt:

```
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;
}

```

## 6.2.5 Hinweise zum Erstellen eigener Programme

Damit selbsterstellte Programme die Custom-Schnittstelle nutzen können, müssen einige Voraussetzungen erfüllt sein. Gehen Sie bitte folgendermaßen vor:

1. Starten Sie die Entwicklungsumgebung Visual C++.
2. Erstellen Sie ein Projekt des gewünschten Typs mit Hilfe des MFC Class Wizards.
3. Kopieren Sie die Dateien "pre\_opc.h" und "pre\_opc.cpp" aus dem Beispielprogramm in Ihr Projektverzeichnis und fügen Sie diese Dateien dem Projekt hinzu (Menü "Projekt" > "Zum Projekt hinzufügen" > "Dateien ...").

Diese Dateien enthalten OPC-spezifische Definitionen und Include-Anweisungen.

4. Ergänzen Sie in allen Implementierungsdateien Ihres Projekts (Endung "\*.cpp") die folgende Anweisung: `#include pre_opc.h`
5. Ergänzen Sie in den Implementierungsdateien (Endung "\*.cpp"), die die Methode `AddGroup()` oder `GetErrorString()` verwenden, die folgende Anweisung: `#define LOCALE_ID0x409`
6. Ordnen Sie die gewünschten Steuerelemente auf dem Hauptdialogfeld an bzw. erstellen Sie ein geeignetes Anwendungsfenster und programmieren Sie die zugehörigen Ereignisprozeduren.

## 6.3 OPC-Custom-Schnittstelle (Asynchrone Kommunikation) in C++

Das vorliegende Beispiel nutzt die Custom-Schnittstelle für Data Access V2.0 von OPC.

### **6.3.1 Aktivieren der Simulationsverbindung**

Damit das vorliegende Programm überhaupt lauffähig ist, müssen Sie eine Simulationsverbindung aktivieren, die die im Programm verwendeten Demo-Variablen verfügbar macht. Gehen Sie dazu folgendermaßen vor:

1. Starten Sie das Programm "Kommunikations-Einstellungen" über das Startmenü:  
"Start" > "Alle Programme" > "Siemens Automation" > "SIMATIC" > "SIMATIC NET" > "PC-Station einstellen".

Reaktion: Die Konfigurations-Konsole "Kommunikations-Einstellungen" wird geöffnet.

2. Öffnen Sie im linken Navigationsfenster den Eintrag "OPC-Protokollauswahl":  
"PC-Station" > "Applikationen" > "OPC-Einstellungen" > "OPC-Protokollauswahl".

3. Aktivieren Sie die Optionskästchen für das zu simulierende Protokoll.

Das vorliegende Beispiel verwendet das S7-Protokoll.

Aktivieren Sie deshalb unter "Name: S7" die Optionskästchen und klicken Sie auf "Details ändern...".

Reaktion: Das Dialog-Fenster "Protokolldetails" wird geöffnet.

4. Aktivieren Sie das Optionskästchen "virtuelle Baugruppe (DEMO) für die Simulation bereitstellen".

5. Bestätigen Sie mit "Übernehmen".

6. Beenden Sie das Programm "Kommunikations-Einstellungen".

---

#### **Hinweis**

Damit die Änderungen wirksam werden, müssen zuvor alle OPC-Clients beendet werden!

---

### **6.3.2 Bedienung des Beispielprogramms**

Das Programm enthält mehrere Bedienelemente, die jeweils folgende Aktionen auslösen:

<b>Bedienelement</b>	<b>Wirkung</b>
Start Sample	Programm starten
Read bzw. Write Item 0	Werte lesen und schreiben
Group Active	Gruppe aktivieren bzw. deaktivieren
Stop Sample	Programm beenden

### **6.3.3 Programm starten**

Das Programm befindet sich auf Ihrer Festplatte unter:  
"<Installationspfad>\SIEMENS\SIMATIC.NET\opc2\samples\custom\async."

Beim Programmstart ist nur die Schaltfläche "Start Sample" aktiviert. Nach dem Betätigen dieser Schaltfläche erzeugt das Programm die notwendigen OPC-Objekte. Danach können auch die anderen Schaltflächen benutzt werden.

### 6.3.4 Werte lesen und schreiben

Nach dem Betätigen der Schaltfläche "Read Item 0" erscheinen der gelesene Wert sowie Statusinformationen in den entsprechenden Textfeldern. Im Ausgangszustand ist das Beispielprogramm für den Betrieb mit einer Demo-Verbindung ausgelegt. Soll das Beispielprogramm in einer realen Umgebung laufen, müssen Sie die folgenden Zeilen im Programm OPCDA\_AsyncDlg.cpp ändern:

```
const LPWSTR szItemID0 = L"S7:[DEMO]MB1";
const LPWSTR szItemID1 = L"S7:[DEMO]MW1";
```

Um einen Wert zu schreiben, müssen Sie diesen in das Textfeld "Value" eintragen und die Schaltfläche "Write Value" betätigen. Das Programm gibt eine Meldung über das Ergebnis der Schreiboperation aus.

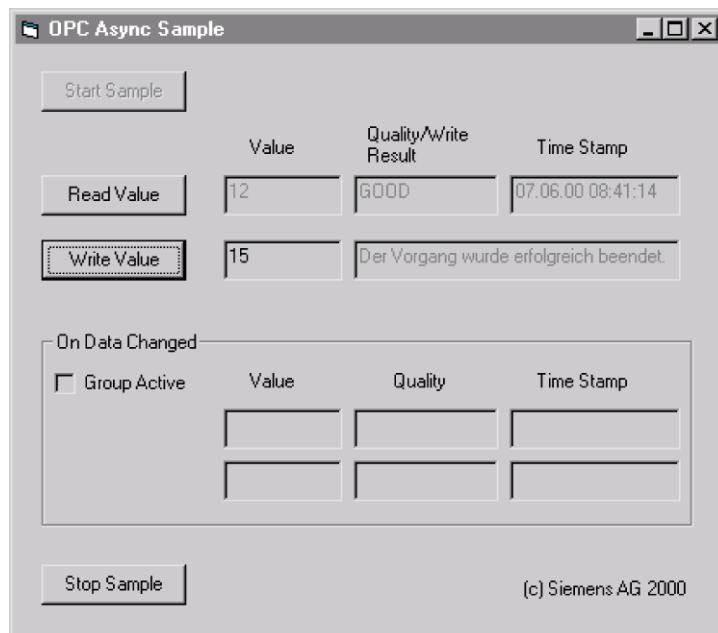


Bild 6-13 Dialog nach Aktivierung der Schaltfläche "Write Value"

### 6.3.5 Gruppe aktivieren

Asynchrone Operationen liefern ein Ergebnis nicht direkt zurück (beispielsweise als Funktionsergebnis oder Rückgabeparameter), sondern mittels Events.

Sie können die OPC-Gruppe im Beispielprogramm aktivieren, indem Sie das Kontrollkästchen "Group Active" markieren. Dann wird u. a. das *OnDataChange-Event*

erzeugt. Eine Prozedur, die beim Auftreten dieses Events abläuft, zeigt Werte und Statusinformationen in den Textfeldern innerhalb des Rahmens "On Data Changed" an.

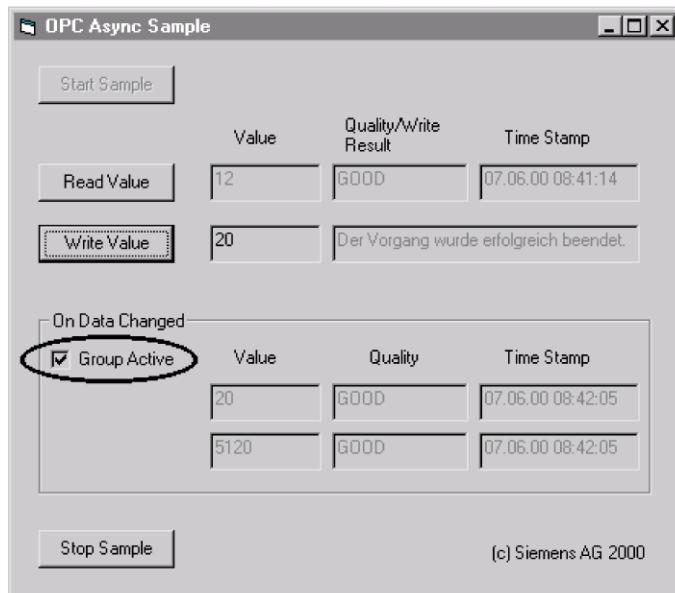


Bild 6-14 Anzeige der Werte und Statusinformationen in den Textfeldern des Feldes "On Data Changed" nach Aktivierung der OPC-Gruppe über das Kontrollkästchen "Group Active"

Events für den Abschluss von Schreib- oder Leseoperationen werden auch von inaktiven Gruppen erzeugt. Deshalb können auch dann Ergebnisse für das Schreiben und Lesen angezeigt werden, wenn das beschriebene Kontrollkästchen nicht markiert ist.

### 6.3.6 Programm beenden

Mit der Schaltfläche "Stop Sample" wird das Programm beendet, d. h. alle OPC-Objekte werden abgebaut und die zugehörigen Ressourcen freigegeben.

### 6.3.7 Beschreibung des Programmablaufs

Bedingt durch das Klassenmodell von OPC muss bei Methodenaufrufen von OPC-Objekten eine bestimmte Reihenfolge eingehalten werden. Um eine Instanz der Klasse *OPCItem* erzeugen zu können, ist ein Objekt der Klasse *OPCGroup* nötig. Das kann aber erst geschehen, wenn eine Instanz der Klasse *OPCServer* vorhanden ist und eine Verbindung zu diesem Server hergestellt wurde.

Die grundsätzliche Abfolge von Befehlen zum Erzeugen und Löschen von OPC-Objekten wird in der folgenden Grafik dargestellt. Dabei werden die Variablennamen des Beispielprogramms verwendet:

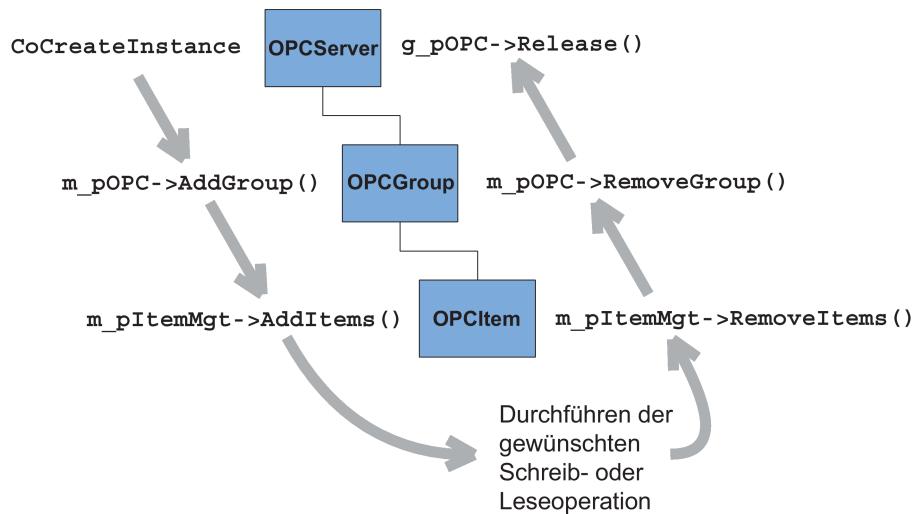


Bild 6-15 Abfolge von Befehlen zum Erzeugen und Löschen von OPC-Objekten

Das Beispielprogramm enthält alle Bestandteile, die in einer typischen Client-Anwendung vorkommen. Dazu gehören der Verbindungsauaufbau zum OPC-Server, das Einrichten einer Gruppe mit Variablen sowie das Lesen und Schreiben von Werten für ein Item. Vor der Detailbeschreibung des Quellcodes soll zunächst der grundlegende Aufbau einer OPC-Anwendung dargestellt werden.

Schritt	Beschreibung
1	Anmelden bei COM
2	Konvertierung der ProgID in eine CLSID
3	Verbindungsauaufbau zum OPC-Server
4	Erzeugen einer OPC-Gruppe
5	Hinzufügen von Items
6	Anfordern eines Schnittstellenzeigers für IOPCGroupStateMgt
7	Anfordern eines Schnittstellenzeigers für IOPCAsyncIO2
8	Callback-Objekt erzeugen
9	OPC-Server und Callback-Objekt des Client verbinden
10	Durchführen der gewünschten Schreib- und Leseoperation
11	Benachrichtigungen des OPC-Servers empfangen
12	Objekte löschen und Speicher freigeben

### Schritt 1: Anmelden bei COM

Jedes Programm, das Funktionen der COM-Library aufrufen möchte, muss sich zunächst bei COM anmelden. Dafür gibt es die Funktion CoInitialize:

```

HRESULT r1;
r1 = CoInitialize(NULL);
  
```

## Schritt 2: Konvertierung der ProgID in eine CLSID

Jeder COM-Server besitzt zur Identifizierung eine sogenannte *ProgID*, die eine weltweit eindeutige *CLSID* zugeordnet ist. Diese wird mit der Funktion *CLSIDFromProgID()* ermittelt. Die ProgID des OPC-Servers von SIMATIC NET ist L"OPC.SimaticNET":

```
r1 = CLSIDFromProgID(L"OPC.SimaticNET", &clsid);
```

## Schritt 3: Verbindungsauftbau zum OPC-Server

Die Funktion *CoCreateInstance()* erzeugt eine Instanz der Klasse, deren *CLSID* vorgegeben wurde:

```
r1 = CoCreateInstance (clsid, NULL, CLSCTX_LOCAL_SERVER,  
                      IID_IOPCServer, (void**)&m_pIOPCServer);
```

Ergebnis dieses Programmschritts ein Objekt der Klasse OPC-Server. Außerdem liefert *CoCreateInstance* einen Zeiger auf das Interface *IOPCServer* des Serverobjekts (Variable *m\_pIOPCServer*):

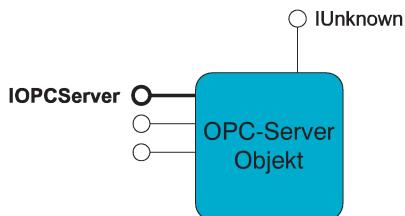


Bild 6-16 Objekt der Klasse OPC-Server mit Zeiger auf die Schnittstelle *IOPCServer*

## Schritt 4: Erzeugen einer OPC-Gruppe

Die Schnittstelle *IOPCServer* besitzt die Methode *AddGroup()* zum Anlegen von Gruppen:

```
HRESULT r1;  
r1 = m_pIOPCServer ->AddGroup(L"grp1", FALSE, 500, 1,  
                                &TimBias, &PercentDeadband, LOCALE_ID, &m_GrpSrvHandle,  
                                &RevisedUpdateRate, IID_IOPCItemMgt,  
                                (LPUNKNOWN*)&m_pIOPCItemMgt);
```

Ergebnis dieser Programmschritte ist eine Gruppe mit dem vorgegebenen Namen und den gewünschten Eigenschaften. Außerdem liefert *AddGroup* als Rückgabeparameter einen Zeiger auf eine angeforderte Schnittstelle des Gruppenobjekts, in diesem Fall *IOPCItemMgt* (Variable *m\_pIOPCItemMgt*):

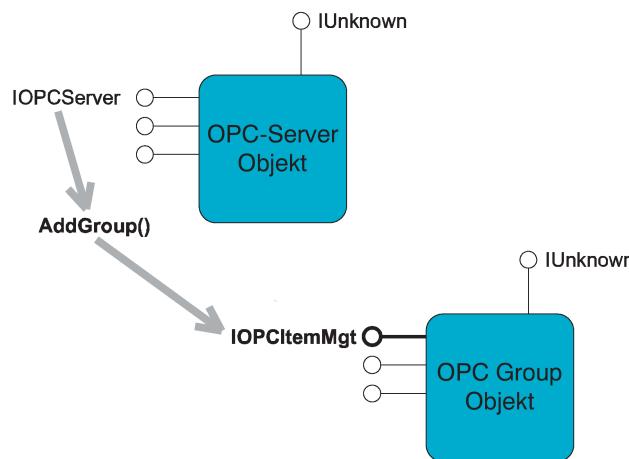


Bild 6-17 Erzeugen einer OPC-Gruppe mit Zeiger auf die angeforderte Schnittstelle *IOPCItemMgt* des Gruppenobjekts

### Schritt 5: Hinzufügen von Items

Die Schnittstelle *IOPCItemMgt* besitzt die Methode *AddItems()* zum Anlegen von OPC-Items:

```

HRESULT r1;
r1 = m_pIOPCItemMgt->AddItems(2, m_Items, &m_pItemResult,
                                &m_pErrors);
  
```

Ergebnis dieses Programmschritts ist, dass der Server zwei Items mit den im Parameter *m\_Items* vorgegebenen Eigenschaften hinzugefügt. Außerdem werden die Variablen der Ergebnisstruktur *m\_pItemResult* (Server-Handle, Datentyp des Items auf dem Zielsystem usw.) mit Werten belegt.

### Schritt 6: Anfordern eines Schnittstellenzeigers für *IOPCGroupStateMgt*

Ein Zeiger auf die Schnittstelle *IOPCGroupStateMgt* ist für die Nutzung der Methoden *SetState* notwendig. Die Methode *SetState* wird im späteren Verlauf zum Aktivieren und Deaktivieren einer Gruppe benötigt. Außerdem wird dieser Schnittstellenzeiger als Parameter für die Methoden *AtAdvise* und *AtUnadvise* benötigt:

```

HRESULT r1;
r1 = m_pIOPCItemMgt->QueryInterface(IID_IOPCGroupStateMgt,
                                         (void**)&m_pIOPCGroupStateMgt);
  
```

Ergebnis dieses Programmschritts ist ein Zeiger auf die Schnittstelle *IOPCGroupStateMgt* (Variable *m\_pIOPCGroupStateMgt*) des Gruppenobjekts:

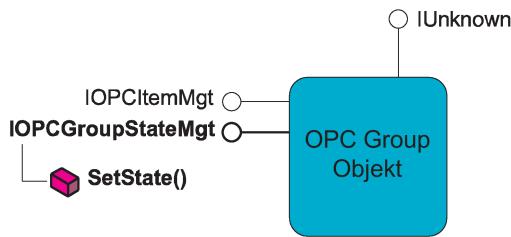


Bild 6-18 OPC-Group-Objekt mit Zeiger auf die Schnittstelle *IOPCGroupStateMgt* und deren Methode *SetState()*

### Schritt 7: Anfordern eines Schnittstellenzeigers für *IOPCAsyncIO2*

Auf gleiche Weise fordert das Programm auch den Zeiger auf die Schnittstelle *IOPCAsyncIO2* (Variable *m\_pIOPCAsyncIO2*) an. Diese Schnittstelle bietet Methoden zum asynchronen Lesen und Schreiben von Werten:

```
r1 = m_pIOPCItemMgt ->QueryInterface(IID_IOPCAsyncIO2, (void**)&m_pIOPCAsyncIO2);
```

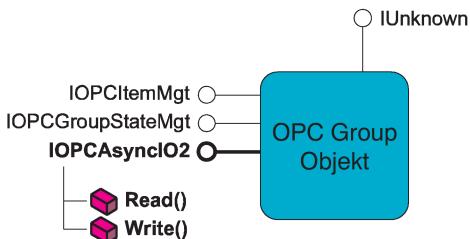


Bild 6-19 OPC-Group-Objekt mit Zeiger auf die Schnittstelle *IOPCAsyncIO2* und deren Methoden zum asynchronen Lesen und Schreiben von Werten

### Schritt 8: Callback-Objekt erzeugen

Damit der OPC-Server bei asynchronen Operationen ein Ergebnis zurückliefern kann, muss im Client die Schnittstelle *IOPCDataCallback* implementiert sein. Im Beispielprogramm übernimmt das die Klasse *COPCDataCallback*, die von *IOPCDataCallback* und *CComObjectRoot* abgeleitet wird.

Die Methode *CreateInstance* erzeugt ein Objekt der Template-Klasse *CComObject*:

```
CComObject<COPCDataCallback>* pCOPCDataCallback;
CComObject<COPCDataCallback>::CreateInstance
&pCOPCDataCallback);
```

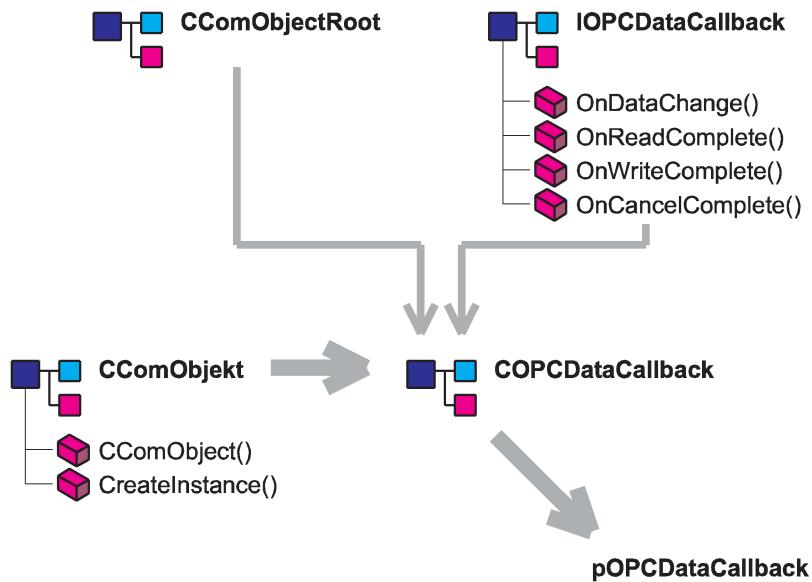


Bild 6-20 Die Schnittstelle `IOPCDataCallback` des Client; die Methode `CreateInstance` erzeugt ein Template-Klassen-Objekt `CComObject`

### Schritt 9: OPC-Server und Callback-Objekt des Client verbinden

Die Methode `AtAdvise()` erzeugt eine Verbindung zwischen dem OPC-Server und dem Callback-Objekt. Zunächst ermittelt die Methode `GetUnknown()` einen Zeiger auf die `IUnknown`-Schnittstelle des Callback-Objekts. `AtAdvise` benötigt diesen Zeiger als Eingangsparameter:

```

LPUNKNOWN pCbUnk;
pCbUnk = pCOPCDataCallback->GetUnknown();
HRESULT hRes = AtlAdvise(m_pIOPCGroupStateMgt, pCbUnk,
                         IID_IOPCDataCallback, &m_dwAdvise);
  
```

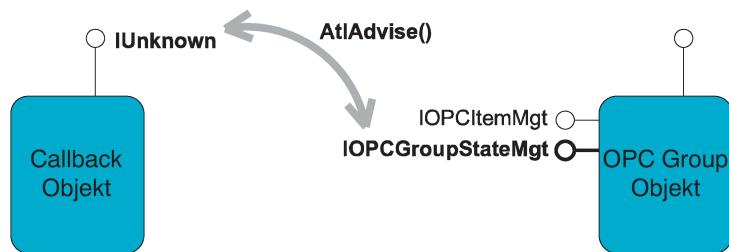


Bild 6-21 Erzeugen einer Verbindung zwischen OPC-Server und dem Callback-Objekt durch die Methode `AtAdvise()`

### Schritt 10: Durchführen der gewünschten Schreib- bzw. Leseoperation

Auf die Methoden `Read()` und `Write()` der Schnittstelle `IOPCAsyncIO2` kann über den Zeiger `m_pIOPCAsyncIO2` zugegriffen werden, der in Schritt 7 erzeugt wurde:

```
r1 = m_pIOPCAsyncIO2->Read(1, phServer, 1, &dwCancelID,  
&pErrors);
```

### Schritt 11: Benachrichtigungen des OPC-Servers empfangen

Bei Änderungen von Daten eines aktiven Items in einer aktiven Gruppe ruft der Server die Methode *OnDataChange* des Callback-Objekts auf. Nach dem Abschluss einer Leseoperation ruft der Server die Methode *OnReadComplete* auf usw. Die Rückgabewerte übergibt der Server als Parameter an die jeweilige Methode.

```
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)
```

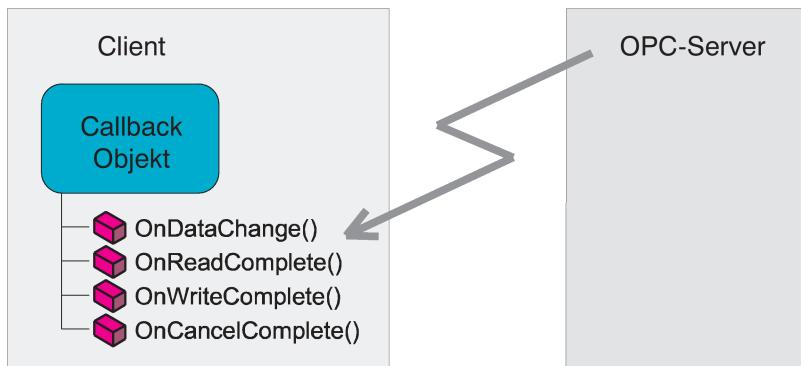


Bild 6-22 Aufruf von Methoden (hier: *OnDataChange()*) des Callback-Objekts durch den OPC-Server

### Schritt 12: Objekte löschen und Speicher freigeben

Vor dem Beenden des Programms müssen die erzeugten OPC-Objekte gelöscht und der angeforderte Speicher freigegeben werden. Die entsprechenden Funktionen sind Bestandteil der bisher benutzten Schnittstellen. *AtlUnadvise* beendet die Verbindung zwischen dem OPC-Server und dem Callback-Objekt:

```
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;
. . .
. . .

```

### Freigabe von Speicher

Bei der Verwendung von COM muss unter bestimmten Umständen der Client Speicher freigeben, der vom Server angefordert wurde.

Die Regeln dazu lauten:

- [out]: Der Speicher für einen Parameter mit diesem Attribut sollte von dem Server angefordert werden.
- [in, out]: Der Speicher für einen solchen Parameter sollte von dem Client angefordert werden. Der Server kann den Speicher wieder freigeben und neu allokieren.
- [in]: Der Client sollte den Speicher anfordern. Der Server ist nicht für die Freigabe des Speichers verantwortlich.

In allen drei Fällen sollte der reservierte Speicher letztendlich von dem Client freigegeben werden.

## 6.3.8

### Beschreibung der Programmstruktur

Diese Beschreibung erklärt den Inhalt der folgenden beiden Dateien:

- "OPCDA\_AsyncDlg.cpp" ist die Implementierungsdatei der programmspezifischen Dialogfeld-Klasse. Hier sind die Ereignisprozeduren für die einzelnen Schaltflächen zu finden. Deren Rümpfe können vom MFC Class Wizard angelegt worden sein. Außerdem sind hier die Methoden für die Initialisierung des Dialogfeldes und für das Anzeigen der Daten definiert.
- "Callback.cpp" ist die Implementierungsdatei der programmspezifischen Callback-Klasse. In diesem Modul sind alle Methoden definiert, die ein Client zur Verfügung stellen muss, damit er Benachrichtigungen eines OPC-Servers empfangen kann.

## 6.3.9

### Die Datei "OPCDA\_AsyncDlg.cpp"

Die Datei "OPCDA\_AsyncDlg.cpp" kann in folgende Abschnitte untergliedert werden:

- Deklaration globaler Variablen
- Die Klasse CAboutDlg
- Methoden der Klasse COPCDA\_AsyncDlg.cpp

### Deklaration der globalen Variablen zur Speicherung der ItemIDs

Am Anfang dieses Moduls werden zwei Variablen mit den Item-Bezeichnungen initialisiert:

```

const LPWSTR szItemID0 = L"S7:[DEMO]MB1";
const LPWSTR szItemID1 = L"S7:[DEMO]MW1";

```

Für die Funktionalität des Programmes ist es wichtig, dass die erste Variable lesbar und schreibbar ist, wohingegen bei der zweiten Variablen die Lesbarkeit genügt.

Beachten Sie bitte auch einige Beispiele und das Handbuch bezüglich des Aufbaus der ItemIDs.

Wenn Sie zwei Variablen wählen, die sich im Speicherabbild überschneiden, ändern sich beide Variablen, wenn Sie eine der beiden Variablen schreiben.

## Die Klasse *CAboutDlg*

Anschließend wird noch die Klasse *CAboutDlg* definiert, die ein Informationsfeld zur Anzeige bringt. Da in dieser Klasse kein OPC-spezifischer Code vorhanden ist, wird auf eine nähere Beschreibung verzichtet.

## Methoden der Klasse *COPCDA\_AsyncDlg*

Die Klasse *COPCDA\_AsyncDlg* besitzt außer Konstruktor und Destruktor noch folgende Methoden:

- DoDataExchange
- OnInitDialog
- OnSysCommand
- OnPaint
- OnQueryDragIcon
- OnButtonStart()
- OnButtonRead()
- OnButtonWrite()
- OnCheckActivategroup
- OnButtonStop()
- OnDestroy
- DisplayData
- UpdateData

## OnInitDialog

Für den Zugriff auf COM-Mechanismen sind mehrere Klassenvariablen vorgesehen, die beim Initialisieren des Hauptdialogfelds mit Anfangswerten belegt werden. Für diese Variablen ist folgende Verwendung vorgesehen:

- m\_pIOPCServer ist ein Zeiger auf die Schnittstelle IOPCServer der Klasse OPC-Server.
- m\_pIOPCItemMgt ist ein Zeiger auf die Schnittstelle IOPCItemMgt der Klasse OPC-Group.
- m\_pIOPCGroupStateMgt ist ein Zeiger auf die Schnittstelle IOPCGroupStateMgt der Klasse OPC-Group.

```
m_pIOPCServer = NULL;
m_pIOPCItemMgt = NULL;
m_pIOPCGroupStateMgt = NULL;
```

Zum Programmstart kann der Benutzer nur die Schaltfläche "Start Sample" betätigen. Alle anderen Schaltflächen sowie das Kontrollkästchen sind deaktiviert. So wird sichergestellt, dass vor dem Durchführen von Schreib- oder Leseoperationen alle notwendigen OPC-Objekte korrekt aufgebaut werden.

Den betreffenden Steuerelementen wurde eine Membervariable zugewiesen, über die der Status verändert werden kann:

```
m_CtrlStop.EnableWindow(FALSE);
m_CtrlRead.EnableWindow(FALSE);
m_CtrlWrite.EnableWindow(FALSE);
m_CtrlChkActive.EnableWindow(FALSE);
```

### OnButtonStart()

*OnButtonStart()* baut die für den Programmablauf notwendigen OPC-Objekte auf. Zunächst werden einige lokale Variablen deklariert, die für die Methoden der OPC-Objekte benötigt werden:

```
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;
    ...
}
```

Danach werden folgende Operationen durchgeführt:

### CoInitialize

*CoInitialize()* initialisiert die COM-Bibliothek. Der Eingangsparameter muss immer *NULL* sein.

```
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()* ermittelt aus einer vorgegebenen ProgID einen weltweit eindeutigen Klassenbezeichner (den sogenannten *Class Identifier* oder auch CLSID). Dieser wird für die Funktion *CoCreateInstance* benötigt.

Diese Methode ist wie folgt deklariert:

```
HRESULT CLSIDFromProgID (LPCOLESTR lpszProgID,
                         LPCLSID pclsid);
```

Parameter	Beschreibung
lpszProgID	Zeiger auf die ProgID [Eingangsparameter].
pclsid	Zeiger auf den ermittelten <i>Class Identifier</i> [Rückgabewert].

Im Beispielprogramm ermittelt diese Methode den Class Identifier des OPC-Servers von SIMATIC NET:

```
r1 = CLSIDFromProgID(L"OPC.SimaticNET", &clsid);
if (r1 != S_OK)
{
    MessageBox("Retrival of CLSID failed",
              "Error CLSIDFromProgID()", 
              MB_OK+MB_ICONERROR);
    CoUninitialize();
    SendMessage(WM_CLOSE);
    return;
}
```

### CoCreateInstance

*CoCreateInstance()* erzeugt eine Instanz der Klasse, deren Klassenbezeichner vorgegeben wurde. Diese Methode ist wie folgt deklariert:

```
STDAPI CoCreateInstance (REFCLSID rclsid,
                        LPUNKNOWN pUnkOuter,
                        DWORD dwClsContext,
                        REFIID riid,
                        LPVOID *ppv);
```

Parameter	Beschreibung
rclsid	Der Klassenbezeichner des gewünschten Objekts.
pUnkOuter	NULL-Zeiger, wenn das zu erzeugende Objekt nicht Teil eines aggregierten Objekts ist. Ein Zeiger, der nicht NULL-Zeiger ist, wird als Zeiger auf die IUnknown-Schnittstelle des aggregierten Objekts interpretiert.

Parameter	Beschreibung
dwClsContext	Beschreibt die Ablaufumgebung des zu erzeugenden Objekts. Es wird damit festgelegt, ob der ausführbare Code, der Objekte dieser Klasse erzeugt und verwaltet, auf der lokalen Maschine läuft und ob dafür ein eigener Prozess erzeugt wird. Die hierfür verwendbaren Konstanten sind im Aufzählungstyp CLSCTX festgelegt. Im Beispiel wird CLSCTX_LOCAL_SERVER benutzt, d. h. der ausführbare Code für die Verwaltung der Server-Objekte läuft auf dem gleichen Rechner wie das Beispielprogramm, aber in einem eigenen Prozess.
riid	Bezeichnung der Schnittstelle, über die mit dem Objekt kommuniziert werden soll, im Beispielprogramm <i>IOPCServer</i> .
ppv	Adresse der Zeigervariable, die bei erfolgreichem Methodenaufruf den gewünschten Schnittstellenzeiger enthält.

Der folgende Methodenaufruf erzeugt ein Objekt der Klasse OPC-Server und liefert einen Zeiger auf die Schnittstelle *IOPCServer* zurück:

```
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

Die Methode *AddGroup()* der Schnittstelle *IOPCServer* erzeugt eine OPC-Gruppe im Server und ist wie folgt deklariert:

```

HRESULT AddGroup (LPWCSTR szName,
                  BOOL bActive,
                  DWORD dwRequestedUpdateRate,
                  OPCHANDLE hClientGroup,
                  LONG *pTimeBias,
                  FLOAT *pPercentDeadband,
                  DWORD dwLCID,
                  OPCHANDLE *phServerGroup,
                  DWORD *pRevisedUpdateRate,
                  REFIID riid,
                  LPUNKNOWN *ppUnk);

```

Parameter	Beschreibung
szName	Gruppenname, kann vom Client beliebig vorgegeben werden, jedoch muss innerhalb des Client Eindeutigkeit gewährleistet sein.
bActive	FALSE, wenn die Gruppe zum Zeitpunkt des Erzeugens inaktiv sein soll. TRUE, wenn die Gruppe zum Zeitpunkt des Erzeugens aktiv sein soll.
dwRequestedUpdateRate	Gibt das kürzeste Zeitintervall an, nach dem ein Client über Änderungen der Werte oder Zustände von Items benachrichtigt werden soll. Mit der Festlegung einer geeigneten Zeitdauer wird verhindert, dass einem Client diese Informationen schneller zugesendet werden als er sie verarbeiten kann.
hClientGroup	Vom Client beliebig wählbare Kennzahl, die vom Server bei bestimmten Benachrichtigungen wieder zurückgeliefert wird. Der Client kann dadurch einen Bezug zu seinen Daten herstellen. Mit diesem Handle identifiziert der Client die Gruppe.
pTimeBias	Abweichung der Serverzeit von UTC (Universal Time Convention).
pPercentDeadband	gibt die Bandbreite an, in der Wertänderungen nicht zu einer Benachrichtigung führen. Diese Eigenschaft ist jedoch nur bei der Verwendung von Analogwerten wirksam. Außerdem muss die Ober- und Untergrenze eines Wertes bekannt sein. Das ist in diesem Beispielprogramm nicht der Fall.
dwLCID	Auswahl der Sprache, die vom Server verwendet wird, wenn Texte zurückgegeben werden.
phServerGroup	Vom Server vergebenes Handle, das bei manchen Funktionsaufrufen als Parameter angegeben werden muss (z. B. <i>RemoveGroup</i> ). Der Server benötigt es um diese Gruppe zu identifizieren.
pRevisedUpdateRate	Vom Server zurückgeliefertes kürzestes Zeitintervall nach dem ein Client über Änderungen der Werte oder Zustände von Items benachrichtigt wird.
riid	Zeiger auf den Bezeichner einer Schnittstellen des OPC-Group-Objekts, die nach dem Erzeugen der Gruppe zur Verfügung stehen soll. Dieser Parameter erspart einen zusätzlichen Aufruf der Methode <i>QueryInterface</i> .
ppUnk	Zeiger auf die gewünschte Schnittstelle.

Beim Aufruf von *AddGroup* im Beispielprogramm werden folgende Vorgaben gemacht:

- Der Parameter *bActive* wird auf den Wert FALSE gesetzt. Direkt nach dem Erstellen ist die Gruppe inaktiv, d. h. für diese Gruppe werden keine OnDataChange-Callbacks erzeugt.
- Die Dauer des gewünschten Aktualisierungintervalls beträgt 500 Millisekunden.
- Das Client-Handle kann beliebig angegeben werden, da nur eine Gruppe verwendet wird.
- Die Voraussetzungen, bei Wertänderungen innerhalb eines bestimmten Bereichs eine Benachrichtigung zu unterdrücken, sind im Beispielprogramm nicht gegeben. Deshalb wird für den Parameter *pPercentDeadband* der Wert "0.0" eingetragen.
- Als Rückgabewert soll *AddGroup* einen Zeiger auf die Schnittstelle *IOPCItemMgt* liefern.

```
r1 = m_pIOPCServer->AddGroup(L"grp1",
                               FALSE,
                               500,
                               1,
                               &TimBias,
                               &PercentDeadband,
                               LOCALE_ID,
                               &m_GrpSrvHandle,
                               &RevisedUpdateRate,
                               IID_IOPCItemMgt,
                               (LPUNKNOWN*)&m_pIOPCItemMgt);
```

### AddItems

Die Methode *AddItems()* der Schnittstelle *IOPCItemMgt* erzeugt OPC-Items und ist wie folgt deklariert:

```
HRESULT AddItems (DWORD dwNumItems,
                  OPCITEMDEF *pItemArray,
                  OPCITEMRESULT **ppAddResults,
                  HRESULT **ppErrors);
```

Parameter	Beschreibung
<i>dwNumItems</i>	Anzahl der Items, die eingefügt werden sollen.
<i>pItemArray</i>	Array mit Elementen vom Typ <i>OPCITEMDEF</i> . Strukturvariablen dieses Typs beinhalten alle Informationen, die vom Server für die Erzeugung von Items benötigt werden.
<i>ppAddResults</i>	Array mit Elementen vom Typ <i>OPCITEMRESULT</i> . Die Rückgabewerte liefert der OPC-Server als Strukturvariablen dieses Typs.
<i>ppErrors</i>	Array mit Elementen vom Typ <i>HRESULT</i> . Diese Variablen liefern einen Fehlercode, wenn Items nicht erfolgreich erzeugt werden konnten bzw. eine Information über den erfolgreichen Methodenaufruf.

Vor dem Aufruf von *AddItems* muss erst ein Array mit Elementen vom Typ OPCITEMDEF erzeugt und mit gültigen Werten belegt werden. Dabei werden folgende Gegebenheiten berücksichtigt:

- Ein Zugriffspfad für das Item ist im Beispiel nicht notwendig, deshalb wird hier ein Leerstring angegeben.
- Die Item-Bezeichnungen wurden am Anfang des Moduls OPCDA\_AsyncDlg.cpp festgelegt und den Variablen szItemID1 und szItemID2 zugewiesen.
- Das Item soll nach der Erzeugung aktiviert sein.
- Das Beispielprogramm verwendet ein Client-Handle von 0 und 1.
- Der OPC-Server für SIMATIC NET benötigt keine *BinaryLargeObjects*, deshalb erhält die Strukturkomponente dwBlobSize den Wert "0".
- Für das erste Item soll der Server den Rückgabewert in einem Typ zurückliefern, der dem ursprünglichen Datentyp des Items entspricht.
- Für das zweite Item soll der Server den Rückgabewert als Zwei-Byte-Integer-Zahl zurückliefern.

```
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;
```

Der Zeiger *m\_pItemResult* ist als Eigenschaft der Klasse *COPCDA\_AsyncDlg* vorhanden. Über diese Variable kann auf die vom Server zurückgelieferten Ergebnisse zugegriffen werden. *m\_pErrors* ist ein Zeiger auf den Fehlercode:

```
r1 = m_pIOPCItemMgt->AddItems(2, m_Items, &m_pItemResult, &m_pErrors);
```

Wenn der Aufruf von *AddItems* nicht erfolgreich war, wird das Programm abgebrochen. Vorher muss das Programm allerdings noch die verwendeten Ressourcen freigegeben:

```
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

Wenn der Rückgabewert von *AddItems()* das Auftreten eines Fehlers anzeigt, liefert die Methode *GetErrorString()* der Schnittstelle *IOPCServer* die zugehörige Fehlermeldung. Diese Methode ist wie folgt deklariert:

```

HRESULT GetErrorString (HRESULT dwError,
                        LCID dwLocale,
                        LPWSTR *ppString);

```

Parameterbeschreibung:

Parameter	Beschreibung
dwError	Vom Server zurückgelieferter Fehlercode.
dwLocale	Die Sprachkennung für die Fehlermeldung.
ppString	Doppelzeiger auf einen nullterminierten String, in den <i>GetErrorString</i> die Fehlermeldung zurückliefert.

```

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

Die Methode *QueryInterface()* der Schnittstelle *IUnknown* liefert einen Zeiger auf eine Schnittstelle, deren Bezeichner als Eingangsparameter vorgegeben wird. *QueryInterface()* ist wie folgt deklariert:

```

HRESULT QueryInterface (REFIID iid, void **ppvObject);

```

Parameterbeschreibung:

Parameter	Beschreibung
iid	Bezeichner der gewünschten Schnittstelle.
ppvObjekt	Adresse der Zeigervariable, die bei erfolgreichem Methodenaufruf den gewünschten Schnittstellenzeiger enthält. Wenn das Objekt diese Schnittstelle nicht unterstützt, wird ein Fehlercode und der NULL-Zeiger zurückgegeben.

*QueryInterface* ermittelt einen Zeiger auf die Schnittstelle *IOPCAsyncIO2*, die Methoden zum asynchronen Lesen und Schreiben bereitstellt:

```
r1 = m_pIOPCItemMgt ->QueryInterface(IID_IOPCAsyncIO2,
                                         (void**)& m_pIOPCAsyncIO2);

if (r1 < 0)
{   MessageBox("No IOPCAsyncIO found!",
            "Error QueryInterface()", MB_OK+MB_ICONERROR);
    . .
    return;
}
```

### Callback-Objekt erzeugen

Im Client-Programm muss die Schnittstelle *IOPCDataCallback* implementiert sein, damit Benachrichtigungen des OPC-Servers empfangen werden können. Im Beispielprogramm wird diese Implementierung mit Hilfe der Active Template Library (ATL) durchgeführt. Die beispieldspezifische Klasse *COPCDataCallback* wird dabei als Parameter für die Template-Klasse *CComObject* verwendet.

Die Methode *CreateInstance* erzeugt ein Objekt der Klasse *CComObject*, auf das über den Zeiger *pOPCDataCallback* zugegriffen wird:

```
CComObject<COPCDataCallback>* pCOPCDataCallback;
CComObject<COPCDataCallback>::CreateInstance(&pCOPCDataCallback);
```

### Verbindung zwischen Dialogfeld und Callback-Objekt herstellen

*OnButtonStart()* ruft die Methode *InformAboutDialog* der Dialogfeldklasse auf und übergibt ihr einen Zeiger auf das aktuelle Dialogfeld (den this-Zeiger). *InformAboutDialog* (im Modul *Callback.cpp*) legt diesen Zeiger in der Member-Variablen *m\_pCDlgClass* ab.

*OnReadComplete()*, *OnWriteComplete()* und *OnDataChange* verwenden diesen Zeiger, um die *DisplayData*-Methode des Dialogfeldes aufzurufen. Ohne die Information in *m\_pCDlgClass* würde der Bezug zum Dialogfeld fehlen, in dem die Daten angezeigt werden sollen.

```
pCOPCDataCallback->InformAboutDialog(this);
```

Der Zusammenhang zwischen den Klassenmethoden von *COPCDA\_AsyncDlg* und *IOPCDataCallback* ist in der folgenden Grafik dargestellt:

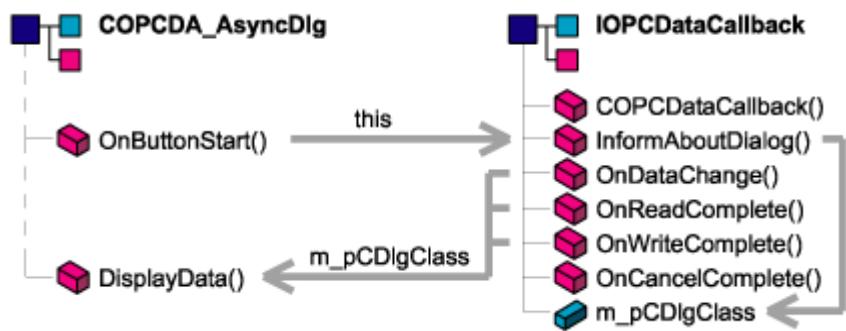


Bild 6-23 Das Zusammenspiel der Klassenmethoden von *COPCDA\_AsyncDlg* und *IOPCDataCallback* zur Verbindung von Dialogfeld und Callback-Objekt

### OPC-Server und Callback-Objekt des Client verbinden

Die Methode *GetUnknown()* ermittelt einen Zeiger auf die *IUnknown*-Schnittstelle des Callback-Objekts:

```
LPUNKNOWN pCbUnk;
pCbUnk = pCOPCDataCallback->GetUnknown();
```

Die Methode *AtlAdvise()* erzeugt eine Verbindung zwischen dem OPC-Server und dem Callback-Objekt. *AtlAdvise* ist wie folgt deklariert:

```
HRESULT AtlAdvise (IUnknown *pUnkCP,
                    IUnknown *pUnk,
                    const IID& iid,
                    LPDWORD pdw)
```

Parameter	Beschreibung
pUnkCP	Zeiger auf die <i>IUnknown</i> -Schnittstelle, zu der der Client eine Verbindung aufbauen möchte.
pUnk	Zeiger auf die <i>IUnknown</i> -Schnittstelle des Callback-Objekts.
iid	Die Bezeichnung des Verbindungspunktes. Üblicherweise wird die Bezeichnung der Schnittstelle angegeben, die die Callback-Funktionalität im Client implementiert.
pdw	Rückgabeparameter. Zeiger auf eine Kennzahl für die Identifizierung der Verbindung. Dieser Wert wird benötigt, wenn mit <i>AtlUnadvise</i> die Verbindung abgebaut wird.

Falls *AtlAdvise* nicht erfolgreich ablaufen konnte, wird dies in einem Dialogfeld angezeigt:

```
HRESULT hRes = AtlAdvise(m_pIOPCGroupStateMgt, pCbUnk,
                           IID_IOPCDataCallback, &m_dwAdvise);
if (hRes != S_OK)
{
    AfxMessageBox("Advise failed!");
    .
    .
    return;
}
```

### Schaltflächen aktivieren

Nachdem *OnButtonStart()* alle notwendigen OPC-Objekte aufgebaut hat, deaktiviert es die Schaltfläche *Start Sample*. Alle anderen Schaltflächen werden aktiviert. Durch diese Vorgehensweise ist sichergestellt, dass *OnButtonStart()* nur einmal durchlaufen wird. So kann auf zusätzliche Abfragen im Programm verzichtet werden.

```
m_CtrlStop.EnableWindow(TRUE);  
m_CtrlRead.EnableWindow(TRUE);  
m_CtrlWrite.EnableWindow(TRUE);  
m_CtrlChkActive.EnableWindow(TRUE);  
m_CtrlStart.EnableWindow(FALSE);
```

### Die Struktur OPCITEMDEF

*OPCITEMDEF* hat folgenden Aufbau:

```
typedef struct {  
    LPWSTR szAccessPath;  
    LPWSTR szItemID;  
    BOOL bActive;  
    OPCHANDLE hClient;  
    DWORD dwBlobSize;  
    BYTE *pBlob;  
    VARTYP vtRequestedDataType;  
    Word wReserved;  
} OPCITEMDEF;
```

Variablen von *OPCITEMDEF*:

Variable	Beschreibung
szAccessPath	optionaler Zugriffspfad für die Items, wird von SIMATIC NET nicht benötigt.
szItemID	Vom Client vergebene ItemID.
bActive	TRUE, wenn bei Wertänderungen des Items in einer aktiven Gruppe eine Benachrichtigung des Client erfolgen soll; FALSE, wenn eine solche Benachrichtigung nicht stattfinden soll.
hClient	Vom Client vergebenes Handle für ein Item. Das Client-Handle über gibt der Server bei Aufrufen an den Client (z.B. <i>OnDataChange</i> ), damit der Client auf die betroffenen Variablen in seinen Strukturen zugreifen kann.
dwBlobSize	Größe eines Speicherbereichs im Server, in dem Zusatzinformationen für einen schnelleren Zugriff auf die Daten eines Items abgelegt werden.
pBlob	Zeiger auf den zuvor beschriebenen Speicherbereich.
vtRequestedDataType	Vom Client gewünschter Datentyp.

### Die Struktur OPCITEMRESULT

*OPCITEMRESULT* hat folgenden Aufbau:

```
typedef struct {  
    OPCHANDLE hServer;  
    VARTYPE vtCanonicalDataType;  
    WORD wReserved;
```

```

    DWORD dwAccessRights;
    DWORD dwBlobSize;
    BYTE *pBlob;
} OPCITEMRESULT;

```

Variablen von *OPCITEMRESULT*:

Variable	Beschreibung
hServer	Vom Server vergebene Handle für ein Item. Das Server-Handle über gibt der Client bei Aufrufen an den Server, damit der Server auf die betroffenen Variablen in seinen Strukturen zugreifen kann.
vtCanonicalDataType	Der Datentyp, der vom Server für ein Item verwendet wird.
dwAccessRights	Information, ob auf ein Item nur lesend, nur schreibend oder lesen und schreibend zugegriffen werden kann.
dwBlobSize	Größe eines Speicherbereichs im Server, in dem Zusatzinformationen für einen schnelleren Zugriff auf die Daten eines Items abgelegt werden.
pBlob	Zeiger auf den zuvor beschriebenen Speicherbereich.

### OnButtonRead()

*OnButtonRead()* startet einen asynchronen Leseauftrag. Dazu werden folgende Operationen durchgeführt.

#### Variablen Deklaration und Plausibilitätsüberprüfung

Nach der Deklaration einiger lokaler Variablen prüft das Programm, ob ein Item vorhanden ist. Nach dem erfolgreichen Aufruf von AddItems in der Methode *OnButtonStart* ist die Variable *m\_pItemResult* mit Werten belegt. Die Array-Komponente *m\_pErrors[0]* muss den Wert *S\_OK* haben, andernfalls wurde das Item nicht korrekt erzeugt:

```

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* liest asynchron Werte für OPC-Items und ist wie folgt deklariert:

```

HRESULT Read (DWORD dwCount,
               OPCHANDLE * phServer,

```

```
DWORD dwTransactionID,
DWORD *pdwCancelID,
HRESULT ** ppErrors);
```

Parameter	Beschreibung
dwCount	Anzahl der Items, für die Werte gelesen werden.
phServer	Array mit den Server-Handles der Items, für die Werte gelesen werden.
dwTransactionID	Vom Client vergebene Kennziffer, um die asynchrone Transaktion zu identifizieren. Sie wird vom Server bei Aufrufen von <i>OnReadComplete</i> zurückgegeben.
pdwCancelID	Vom Server erzeugte Kennziffer, die angegeben werden muss, wenn die Transaktion abgebrochen werden soll.
ppErrors Array	Array mit Elementen vom Typ HRESULT. Diese Variablen liefern einen Fehlercode, wenn <i>Read()</i> nicht erfolgreich aufgerufen wurde bzw. eine Information über den erfolgreichen Methodenaufruf.

Das Programm legt Speicher für ein Array mit einem Element vom Typ OPCHANDLE an und initialisiert das einzige Arrayelement mit der Strukturkomponente *hServer* der Variablen *m\_pItemResult*. Dort hat der OPC-Server beim Aufruf von *AddItems* ein Server-Handle für das Item abgelegt:

```
phServer = new OPCHANDLE[1];
phServer[0] = m_pItemResult [0].hServer;
r1 = m_pIOPCAsyncIO2 ->Read(1, phServer, 1, &dwCancelID,
&pErrors);
```

Der Rückgabewert *S\_FALSE* zeigt an, dass ein oder mehrere Items nicht gelesen werden konnten. In diesem Fall ermittelt die Methode *GetErrorString* die zugehörige Fehlermeldung und zeigt diese in einem Dialogfeld an:

```
if (r1 == S_FALSE)
{   m_pIOPCServer ->GetErrorString(pErrors[0], LOCALE_ID,
&ErrorStr);
szOut.Format ("%s", ErrorStr);
MessageBox(szOut, "Error Reading Item0",
MB_OK+MB_ICONERROR);
CoTaskMemFree(ErrorStr);
}
```

Vor dem Programmende wird der von *OnButtonRead()* reservierte Speicher freigegeben. Beim Array *phServer* erfolgt dies mit *delete*, weil dieser Speicher mit *new* beschafft wurde. Der Speicherbereich des Arrays *pErrors* wurde von *CoTaskMemAlloc* organisiert, deshalb wird er mit *CoTaskMemFree* freigegeben:

```
delete[] phServer;
CoTaskMemFree(pErrors);
}
```

## OnButtonWrite()

*OnButtonWrite()* startet einen asynchronen Schreibauftrag. Dazu werden folgende Operationen durchgeführt.

### Variablen-deklaration und Plausibilitätsüberprüfung

Die Variablen-deklaration entspricht der Methode *OnButtonRead()*. Zusätzlich wird ein Array vom Typ VARIANT für das Ergebnis der Leseoperation deklariert. Für die Überprüfung, ob ein Item vorhanden ist, wird wie in *OnButtonRead()* der Inhalt der Strukturkomponente *hServer* herangezogen:

```
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* schreibt asynchron Werte für OPC-Items und ist wie folgt deklariert:

```
HRESULT Write (DWORD dwCount,
               OPCHANDLE *phServer,
               VARIANT *pItemValues,
               DWORD dwTransactionID,
               DWORD *pdwCancelID
               HRESULT **ppErrors);
```

Parameter	Beschreibung
dwCount	Anzahl der Items, für die Werte geschrieben werden.
phServer	Array mit den Server-Handles der Items, für die Werte geschrieben werden.
pItemValues	Array mit den zu schreibenden Werten.
dwTransactionID	Vom Client vergebene Kennziffer, um die asynchrone Transaktion zu identifizieren.
pdwCancelID	Vom Server erzeugte Kennziffer, die angegeben werden muss, wenn die Transaktion abgebrochen werden soll.
ppErrors	Array mit Elementen vom Typ HRESULT. Diese Variablen liefern einen Fehlercode, wenn <i>Write()</i> nicht erfolgreich aufgerufen wurde bzw. eine Information über den erfolgreichen Methodenaufruf.

Das Beispiel verwendet den Parameter *dwTransactionID* nicht, deshalb wird ein beliebiger Wert vorgegeben:

```
r1 = m_pIOPCAsyncIO2 ->Write(1, phServer, values, 2,
                                &dwCancelID, &pErrors);
```

Falls Write *S\_FALSE* zurückgibt, war die Ausführung fehlerhaft. Das Programm ermittelt die zugehörige Fehlermeldung. Wenn *Write* nicht ausgeführt werden konnte, wird eine entsprechende Fehlermeldung in einer Dialogbox angezeigt.

Mit *delete[] phServer* wird der von *OnButtonWrite()* reservierte Speicher freigegeben:

```
delete[] phServer;
if (r1 == S_FALSE)
{
    m_pOPCServer->GetErrorString(pErrors[0], LOCALE_ID,
                                   &ErrorStr);

    m_WriteResult = ErrorStr;
    UpdateData(FALSE);
    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);
}
```

#### Werte aus dem Dialogfeld übernehmen

Der Aufruf der Methode *UpdateData* mit dem Parameter TRUE überträgt den Inhalt aller Steuervariablen in die entsprechenden Member-Variablen. Der Wert der Membervariable *m\_WriteVal* des Textfeldes IDC\_EDIT\_WRITEVAL wird in der Komponente *iVa* des einzigen Elements von *values[]* gespeichert. Die Komponente *v* bekommt die Typangabe VT\_I2 zugewiesen, d. h. *iVa* soll als Integer-Zahl der Länge zwei Byte interpretiert werden:

```
UpdateData(TRUE);
values[0].vt = VT_I2;
values[0].ival = m_WriteVal;
```

#### OnCheckActivategroup

*OnCheckActivategroup* wird aufgerufen, wenn das Kontrollkästchen "Group Active" betätigt wird. Es aktiviert bzw. deaktiviert die Gruppe, abhängig vom Status des Kontrollkästchens.

Dazu werden folgende Operationen durchgeführt:

#### Wert aus dem Dialogfeld übernehmen

Der Aufruf der Methode *UpdateData* mit dem Parameter TRUE überträgt den Inhalt aller Steuerelemente in die entsprechenden Member-Variablen:

```
void COPCDA_AsyncDlg::OnCheckActivategroup()
{
    UpdateData(TRUE);
```

#### SetState

*SetState* legt mehrere Eigenschaften einer Gruppe fest und ist wie folgt deklariert:

```

HRESULT SetState (DWORD * pRequestedUpdateRate,
                  DWORD * pRevisedUpdateRate,
                  BOOL *pActive,
                  LONG * pTimeBias,
                  FLOAT * pPercentDeadband
                  DWORD * pLCID,
                  OPCHANDLE *phClientGroup) ;

```

Parameter	Beschreibung
pRequestedUpdateRate	Vom Client gewünschtes Aktualisierungsintervall für Daten und Zustandsänderungen von Items.
pRevisedUpdateRate	Das kürzeste vom Server realisierbare Aktualisierungsintervall für die Gruppe.
pActive	Status der Gruppe: TRUE, um die Gruppe zu aktivieren. FALSE, um die Gruppe zu deaktivieren.
pTimeBias	Abweichung der Serverzeit von UTC (Universal Time Convention).
pPercentDeadband	gibt die Bandbreite an, in der Wertänderungen nicht zu einer Benachrichtigung führen. Diese Eigenschaft ist jedoch nur bei der Verwendung von Analogwerten wirksam. Außerdem muss die Ober- und Untergrenze eines Wertes bekannt sein. Das ist in diesem Beispielprogramm nicht der Fall.
pLCID	Kennziffer für die lokale Anpassung der Gruppe (Sprache usw.).
phClientGroup	Neues ClientHandle für die Gruppe.

Es soll lediglich die Gruppe aktiviert werden, deshalb wird auch nur der Parameter *pActive* mit einem Wert belegt. Außerdem muss für den Rückgabeparameter *pRevisedUpdateRate* eine entsprechende Variable zur Verfügung stehen:

```

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()* baut die im Programm verwendeten OPC-Objekte ab und gibt die zugehörigen Ressourcen frei. Diese Methode wird aufgerufen, wenn die Schaltfläche "Stop Sample" betätigt, die Methode *OnDestroy* durchlaufen wird oder die Nachricht WM\_CLOSE gesendet wird (nach Betätigen der Schaltflächen zum Schließen des Dialogfeldes oder explizit durch Aufruf von *SendMessage*).

Es werden folgende Operationen durchgeführt:

### AtlUnadvise

*AtlUnadvise* beendet die Verbindung zwischen dem OPC-Server und dem Callback-Objekt und ist wie folgt deklariert:

```
HRESULT AtlUnadvise (IUnknown*pUnkCP,
                      const IID &iid,
                      DWORD dw);
```

Parameter	Beschreibung
pUnkCP	Zeiger auf die IUnknown-Schnittstelle des Objekts, mit dem der Client verbunden ist.
iid	Die Bezeichnung des Verbindungspunktes. Üblicherweise wird die Bezeichnung der Schnittstelle angegeben, die die Callback-Funktionalität im Client implementiert.
dw	Kennziffer, mit der die Verbindung identifiziert werden kann.

```
void COPCDA_AsyncDlg::OnButtonStop()
{
    HRESULT r1;
    OPCHANDLE *phServer;
    HRESULT *pErrors;
    HRESULT hRes = AtlUnadvise(m_pIOPCGroupStateMgt,
                               IID_IOPCDataCallback,
                               m_dwAdvise);
```

### Release

Jede COM-Schnittstelle verfügt über die Methode *Release*, mit der die von der Schnittstelle belegten Ressourcen freigegeben werden. Genau genommen entfernt *Release* noch keine Objekte aus dem Speicher, es dekrementiert lediglich den Referenzzähler der Schnittstelle. Erst wenn die Schnittstelle nicht mehr referenziert wird, werden die entsprechenden Objekte tatsächlich gelöscht.

```
m_pIOPCGroupStateMgt ->Release();
```

### RemoveItems

Die Methode *RemoveItems* der Schnittstelle *IOPCItemMgt* löscht OPC-Items und ist wie folgt deklariert:

```
HRESULT RemoveItems (DWORD dwCount,
                     OPCHANDLE *phServer,
                     HRESULT **ppErrors );
```

Parameter	Beschreibung
dwCount	Anzahl der zu löschenen Items.
phServer	Array mit den Server-Handles der zu entfernenden Items.
ppErrors	Array mit Elementen vom Typ HRESULT. Diese Variablen liefern einen Fehlercode, wenn <i>RemoveItems()</i> nicht erfolgreich aufgerufen wurde bzw. eine Information über den erfolgreichen Methodenaufruf.

Dieser Programmabschnitt legt Speicher für ein Array mit einem Element vom Typ OPCHANDLE an und initialisiert die Array-Element mit der Strukturkomponente *hServer* der Variablen *m\_pItemResult*. Dort hat der OPC-Server beim Aufruf von *AddItems* die Server-Handles für die Items abgelegt. Nach dem Aufruf von *RemoveItems* wird der Speicher für dieses Array wieder freigeben:

```
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;
```

Die Speicherbereiche für *pErrors* und *m\_pItemResult* werden mit der Methode *CoTaskMemFree* freigegeben, weil sie über COM-Mechanismen beschafft wurden. Die Ressourcen von *m\_pIOPCItemMgt* und *m\_pIOPCAsyncIO2* werden von Release freigegeben:

```
CoTaskMemFree(pErrors);
CoTaskMemFree(m_pItemResult);
m_pItemResult =NULL;
CoTaskMemFree(m_pErrors);
m_pErrors =NULL;
m_pIOPCAsyncIO2->Release();
m_pIOPCAsyncIO2 = NULL;
m_pIOPCItemMgt->Release();
m_pIOPCItemMgt = NULL;
```

## RemoveGroup

*RemoveGroup()* entfernt eine Gruppe aus dem Server und ist wie folgt deklariert:

```
HRESULT RemoveGroup (OPCHANDLE hServerGroup, BOOL bForce);
```

Parameter	Beschreibung
hServerGroup	Serverhandle der Gruppe, die gelöscht werden soll.
bForce	Legt fest, ob Gruppen auch dann gelöscht werden können, wenn noch eine Referenz darauf besteht.

```
r1 = m_pOPCServer->RemoveGroup(m_GrpSrvHandle, TRUE);
if (r1 != S_OK)
{   MessageBox("RemoveGroup failed!",
            "Error RemoveGroup()", MB_OK+MB_ICONERROR);
}
m_GrpSrvHandle = NULL;
```

Die Ressourcen des OPC-Serverobjekts werden mit Release freigegeben, *CoUninitialize* schließt die COM-Bibliothek:

```
m_pIOPCServer->Release();  
m_pIOPCServer = NULL;  
CoUninitialize();
```

Abschließend deaktiviert das Programm alle Steuerelemente bis auf die Schaltfläche "Start Sample":

```
m_CtrlStart.EnableWindow(TRUE);  
m_CtrlStop.EnableWindow(FALSE);  
m_CtrlRead.EnableWindow(FALSE);  
m_CtrlWrite.EnableWindow(FALSE);  
m_CtrlChkActive.EnableWindow(FALSE);  
}
```

## OnDestroy

Diese Methode der Klasse CWnd wird überschrieben, damit beim Schließen des Dialogfensters auch alle "Aufräumarbeiten" durchgeführt werden. *OnButtonStop()* baut alle OPC-Objekte ab und gibt die zugehörigen Ressourcen frei. Danach wird *OnDestroy* der Elternklasse aufgerufen, um den vom Fensterobjekt belegten Speicher freizugeben:

```
void COPCDA_AsyncDlg::OnDestroy()  
{  
    if (m_pIOPCServer)  
        OnButtonStop();  
    CDialog::OnDestroy();  
}
```

## DisplayData

Die Dialogfeldklasse verfügt über drei Methoden, um nach einer Meldung des OPC-Servers Daten anzuzeigen. Es sind unterschiedliche Methoden notwendig, weil die verschiedenen Benachrichtigungen des Servers unterschiedliche Daten zurückliefern.

Es stehen folgende Methoden zur Verfügung:

- Nach der Änderung von Daten (je eine Methode pro Item):

```
void DisplayData0 (CString ReadVal, CString ReadQuality, CString ReadTS) void  
DisplayData1 (CString ReadVal, CString ReadQuality, CString ReadTS)
```

- Nach dem Abschluss einer Leseoperation:

```
void DisplayData (long Value, CString Quality, CString TimeStamp)
```

- Nach dem Abschluss einer Schreiboperation:

```
void DisplayData (HRESULT ErrorCode)
```

## DisplayData0 (CString ReadVal, CString ReadQuality, CString ReadTS)

Diese Version von *DisplayData* wird von der Methode *OnDataChange* der Callback-Klasse aufgerufen und aktualisiert die Anzeige in den Textfeldern, die sich innerhalb des Rahmens "On Data Changed" befinden.

---

### Hinweis

In diesem Fall werden die Textfelder von Item0 aktualisiert. Um die Textfelder von Item1 zu aktualisieren wird analog die Methode *DisplayData1* implementiert.

---

```
void COPCDA_AsyncDlg::DisplayData0(CString ReadVal,
                                    CString ReadQuality,
                                    CString ReadTS)
```

Parameter	Beschreibung
ReadVal	Gelesener Wert.
ReadQuality	Information über die Integrität der Daten.
ReadTS	Zeitangabe (Der Zeitpunkt, zu dem die Daten vom OPC-Server empfangen wurden).

Die Parameterwerte werden den Member-Variablen des Dialogfeldes zugewiesen. *UpdateData* überträgt die neuen Daten aus den Member-Variablen auf die zugehörigen Steuerelemente:

```
{   m_Quality0 = ReadQuality;
    m_TimeStamp0 = ReadTS;
    m_Value0 = ReadVal;
    UpdateData(FALSE);
}
```

## DisplayData (long Value, CString Quality, CString TimeStamp)

Die Methode *OnReadComplete()* der Callback-Klasse ruft diese Methode auf, um die Anzeige in den Textfeldern rechts neben der Schaltfläche "Read Value" zu aktualisieren:

```
void COPCDA_AsyncDlg::DisplayData(long Value,
                                    CString Quality,
                                    CString TimeStamp)
```

Parameter	Beschreibung
Value	Der gelesene Wert.
Quality	Eine Angabe über die Integrität der Daten.
TimeStamp	Der Zeitpunkt, zu dem die Daten vom OPC-Server empfangen wurden.

Die Parameter werden den entsprechenden Member-Variablen des Dialogfeldes zugewiesen. *UpdateData* überträgt die neuen Daten aus den Member-Variablen auf die zugehörigen Steuerelemente:

```
{    m_ReadValue = Value;
    m_QualityRead = Quality;
    m_TimeStampRead =TimeStamp;
    UpdateData(FALSE);
}
```

### DisplayData (HRESULT ErrorCode)

Die Methode *OnWriteComplete()* der Callback-Klasse ruft diese Methode auf, um eine Statusmeldung über das Ergebnis einer Schreiboperation in einem Textfeld rechts neben der Schaltfläche "Write Value" anzuzeigen:

```
void COPCDA_AsyncDlg::DisplayData( HRESULT ErrorCode)
```

Parameter	Beschreibung
ErrorCode	Der vom OPC-Server zurückgelieferte Fehlercode.

Die Methode *GetErrorString()* der Schnittstelle *IOPCServer* liefert die zum Fehlercode gehörende Fehlermeldung. Diese Meldung wird der Member-Variablen *m\_WriteResult* zugewiesen. Bevor *UpdateData* diese Meldung im Textfeld anzeigt, müssen mit *Remove()* noch die Zeilenendkennzeichen entfernt werden:

```
{    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

In bestimmten Situationen kann ein Thread-Wechsel während eines Ressourcen-Zugriffs zu inkonsistenten Daten oder undefinierten Zuständen führen. In solchen Fällen muss ein Thread-Wechsel explizit verhindert werden. Dafür steht die Klasse *CCriticalSection* zur Verfügung. Nach dem Erzeugen eines Objekts dieser Klasse sind Thread-Wechsel erst wieder möglich, wenn die Methode *Unlock()* aufgerufen wird.

Das Beispielprogramm nutzt dieses Verfahren, um eine fehlerfreie Aktualisierung der Steuerelemente durchzuführen. Dazu wird nach dem Programmstart ein Objekt der Klasse *CCriticalSection* erzeugt. Unmittelbar nach dem Aufruf der Methode *UpdateData* der Elternklasse gibt *Unlock* die Ressourcen auch für andere Threads wieder frei:

```
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 und Callback.h

Im Beispielprogramm ist die Klasse *COPCDataCallback* die Implementierung der Schnittstelle *IOPCDataCallback*. Diese Klasse stellt die folgenden Methoden zur Verfügung, um Benachrichtigungen des OPC-Servers zu empfangen:

- *OnDataChange()*
- *OnReadComplete()*
- *OnWriteComplete()*
- *InformAboutDialog*

Weiterhin befindet sich in der Datei die Hilfsfunktion *GetQualityText*

#### OnDataChange

Der OPC-Server ruft diese Methode auf, wenn sich Daten ändern:

```
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)
```

Parameter	Beschreibung
dwTransid	Die Kennziffer für die Art des Methodenaufrufs ist Null, wenn eine Wertänderung die Ursache für den Methodenaufruf war und ungleich Null, wenn ein Refresh der Auslöser war.
hGroup	Das Client-Handle der Gruppe zur Identifikation der Gruppe durch den Client
hrMasterquality	Information über die Integrität der Daten: S_OK, wenn OPC_QUALITY_MASK bei alle Items den Wert OPC_QUALITY_GOOD besitzt, andernfalls S_FALSE.
hrMastererror	S_OK, wenn die Fehlermeldung bei allen Items S_OK ist, andernfalls S_FALSE.
dwCount	Anzahl der Items, für die Werte vorliegen.
phClientItems	Array mit den Client-Handles der Items, deren Werte sich geändert haben.
pvValues	Array vom Typ VARIANT, das die Werte der Items enthält, die sich geändert haben.
pwQualities	Array mit Informationen über die Integrität der Werte der einzelnen Items
pftTimeStamps	Array mit Zeitangaben für die einzelnen Items
pErrors	Array mit Fehlermeldungen für die Items

Es ist die Aufgabe dieser Methode, die vom OPC-Server als Parameter übergebenen Werte

in Objekte vom Typ `CString` umzuwandeln. Die Methode `DisplayData` erwartet Eingangsparameter dieses Typs.

Das Programm erzeugt drei Arrays für `CString`-Objekte, um die Werte der Items, eine Informationen über die Qualität dieser Werte und eine Zeitangabe zu speichern:

```
{   CString szReadVal;
    CString szReadQuality;
    CString szReadTS;
```

`CComVariant` ist eine Wrapper-Klasse für den Datentyp VARIANT. In einer Schleife wird für jedes Item ein `CComVariant`-Objekt erzeugt. Dafür wird der Konstruktor verwendet, der einen VARIANT-Wert als Parameter erhält:

```
CComVariant *pvarOut;
DWORD i;
for (i = 0; i<dwCount; i++)
{   pvarOut = new CComVariant(pvValues[i]);
```

Im nächsten Schritt wandelt die Methode `ChangeType` den gelesenen Wert in einen String um. Da es sich aber nach wie vor um ein Objekt der Klasse `CComVariant` handelt, kann über die Komponente `bstrVal` von `pvarOut` auf den String zugegriffen werden. Das Ergebnis der Methode `Format` ist das gewünschte `CString`-Objekt:

```
pvarOut->ChangeType (VT_BSTR);
szReadVal.Format ("%ls", pvarOut->bstrVal);
```

Da nur die Variable `szReadVal` von Interesse ist, wird das Objekt `pvarOut` unmittelbar nach seiner Verwendung wieder abgebaut:

```
delete pvarOut;
```

`GetQualityText` liefert eine Qualitätsangabe für den gelesenen Wert als `CString`-Objekt. Der Parameter für diese Methode ist eine Kennziffer des OPC-Servers:

```
szReadQuality = GetQualityText (pwQualities[i]);
```

Der OPC-Server liefert die Zeitangabe für die Items als Variable des Typs FILETIME. Das Ergebnis der Methode `Format` der Klasse `COleDateTime` ist die Zeitangabe als `CString`-Objekt:

```
szReadTS = COleDateTime (pftTimeStamps[i]).Format ();
}
```

Nachdem alle Ergebnisse in `CString`-Objekte umgesetzt wurden, ruft das Programm entweder die Methode `DisplayData0` oder die Methode `DisplayData1` der Klasse `COPCDA_AsyncDlg` auf:

```
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()

Der OPC-Server ruft diese Methode auf, wenn ein asynchroner Lesevorgang abgeschlossen wurde. Die Parameter von *OnReadComplete()* stimmen mit denen von *OnDataChange* überein:

```

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)

```

Parameter	Beschreibung
dwTransid	Die Kennziffer für die Art des Methodenaufrufs ist Null, wenn eine Werteänderung die Ursache für den Methodenaufruf war und ungleich Null, wenn ein Refresh der Auslöser war.
hGroup	Das Client-Handle der Gruppe zur Identifikation der Gruppe durch den Client
hrMasterquality	Information über die Integrität der Daten: S_OK, wenn OPC_QUALITY_MASK bei allen Items den Wert OPC_QUALITY_GOOD besitzt, andernfalls S_FALSE.
hrMastererror	S_OK, wenn die Fehlermeldung bei allen Items S_OK ist, andernfalls S_FALSE.
dwCount	Anzahl der Items, für die Werte vorliegen.
phClientItems	Array mit den Client-Handles der Items, deren Werte sich geändert haben.
pvValues	Array vom Typ VARIANT, das die Werte der Items enthält, die sich geändert haben.
pwQualities	Array mit Informationen über die Integrität der Werte der einzelnen Items
pftTimeStamps	Array mit Zeitangaben für die einzelnen Items
pErrors	Array mit Fehlermeldungen für die Items

Vor der Bearbeitung der vom OPC-Server gelieferten Daten prüft das Programm die Variable *pErrors*. Wenn fehlerfrei gelesen wurde, erfolgt die Auswertung der Daten wie im

Programm *OnDataChange*. Allerdings wird der Wert des Items in den Typ *Long Integer* umgesetzt:

```
if (pErrors[0] == S_OK)
{
    CComVariant * pvarOut;
    pvarOut = new CComVariant(pvValues[0]);
    pvarOut->ChangeType(VT_I4);
    CString readQuality = GetQualityText(pwQualities[0]);
    CString readTS = COleDateTime(pftTimeStamps[0]).Format();
    m_pCDlgClass->DisplayData(pvarOut->intVal, readQuality, readTS);
    delete pvarOut;
}
```

Wenn ein Fehler aufgetreten ist, liefert die Methode *GetQualityText* die zum Fehlercode gehörende Meldung:

```
else
{
    CString readQuality = GetQualityText(pErrors[0]);
    m_pCDlgClass->DisplayData(0, readQuality, "");
}
return S_OK;
};
```

### OnWriteComplete()

Der OPC-Server ruft diese Methode auf, wenn ein asynchroner Schreibvorgang abgeschlossen wurde:

```
STDMETHODIMP COPCDataCallback::OnWriteComplete(
    DWORD dwTransid,
    OPCHANDLE hGroup,
    HRESULT hrMastererr,
    DWORD dwCount,
    OPCHANDLE __RPC_FAR *pClienthandles,
    HRESULT __RPC_FAR *pErrors)
```

Parameter	Beschreibung
dwTransid	Eine Kennziffer für die asynchrone Schreiboperation
hGroup	Das Client-Handle der Gruppe
hrMastererr	S_OK, wenn die Fehlermeldung bei allen Items S_OK ist, andernfalls S_FALSE.
dwCount	Anzahl der Items, für die Werte geschrieben wurden.
pClienthandles	Array mit den Client-Handles der Items, für die Werte geschrieben wurden.
pErrors	Array mit Fehlermeldungen für die Items

*OnWriteComplete()* ruft die Methode *DisplayData* der Klasse *COPCDA\_AsyncDlg* auf, um die Fehlermeldung des OPC-Servers anzuzeigen:

```
{    m_pCDlgClass->DisplayData(pErrors[0]);
    return S_OK;
};
```

## InformAboutDialog

InformAboutDialog liefert eine Referenz auf die Dialog Box:

```
void InformAboutDialog (COPCDA_AsyncDlg *pCDlgClass)
{   m_pCDlgClass = pCDlgClass;
}
```

## GetQualityText

GetQualityText liefert zu einem vorgegebenen Fehlercode eine Fehlermeldung als CString-Objekt:

```
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 Hinweise zum Erstellen eigener Programme**

Damit selbsterstellte Programme die Custom-Schnittstelle nutzen können, müssen einige Voraussetzungen erfüllt sein. Gehen Sie bitte folgendermaßen vor:

1. Starten Sie die Entwicklungsumgebung Visual C++.
2. Erstellen Sie ein Projekt des gewünschten Typs mit Hilfe des MFC Class Wizards.
3. Kopieren Sie die Dateien "pre\_opc.h" und "pre\_opc.cpp" aus dem Beispielprogramm in Ihr Projektverzeichnis und fügen Sie diese Dateien dem Projekt hinzu (Menü "Projekt" > "Zum Projekt hinzufügen" > "Dateien ...").

Diese Dateien enthalten OPC-spezifische Definitionen und Include-Anweisungen.

4. Ergänzen Sie in allen Implementierungsdateien Ihres Projekts (Endung "\*.cpp") die folgende Anweisung: `#include pre_opc.h`
5. Ergänzen Sie in den Implementierungsdateien (Endung "\*.cpp"), die die Methode `AddGroup()` oder `GetErrorString()` verwenden, die folgende Anweisung: `#define LOCALE_ID 0x409`
6. Ordnen Sie die gewünschten Steuerelemente auf dem Hauptdialogfeld an bzw. erstellen Sie ein geeignetes Anwendungsfenster und programmieren Sie die zugehörigen Ereignisprozeduren.
7. Implementieren Sie ein Callback-Objekt, wenn Sie asynchrone Operationen über ATL nutzen wollen. Am zweckmäßigsten ist es, eine eigene Klasse mit den notwendigen Methoden zu definieren.

## **6.4 OPC-Custom-Schnittstelle (Asynchrone Kommunikation) in VB.NET**

### **Voraussetzungen für die Verwendung des Beispielprogramms**

Nach der Installation der SIMATIC NET-Software finden Sie dieses Programm auf Ihrer Festplatte in folgendem Verzeichnis:

"<Installationspfad>\SIEMENS\SIMATIC.NET\opc2\samples\dotnet\vb\async.net"

Auf dem Rechner, auf dem das Beispielprogramm ablaufen soll, muss ein .NET-Framework ab Version 4.0 installiert sein. Außerdem müssen Sie eine Simulationsverbindung aktivieren, damit die im Programm verwendeten Demo-Variablen verfügbar sind (siehe Kapitel "Aktivieren der Simulationsverbindung (Seite 662)").

## Runtime Callable Wrapper

Das vorliegende Beispiel nutzt die Custom-Schnittstelle für OPC Data Access V2.0 mit dem .NET Framework. Der Zugriff auf die COM-Schnittstelle aus verwaltetem Code von .NET-Anwendungsprogrammen erfolgt über den Runtime Callable Wrapper (RCW) der OPC Foundation.

Die primäre Funktion von Runtime Callable Wrappern besteht im Marshallen von Aufrufen zwischen einem .NET-Client und einem unverwalteten COM-Objekt. Unter Marshallen versteht man das Verfügbarmachen von Speicherbereichen.

### 6.4.1 Bedienung des Beispielprogramms

Das Programm enthält mehrere Bedienelemente, die jeweils folgende Aktionen auslösen:

Bedienelement	Wirkung
Start Sample	Programm starten
Read Value bzw. Write Value	Werte lesen bzw. schreiben
Group Active	Gruppe aktivieren bzw. deaktivieren
Stop Sample	Programm beenden

Beim Programmstart ist nur die Schaltfläche "Start Sample" aktiviert. Nach dem Betätigen dieser Schaltfläche erzeugt das Programm die notwendigen OPC-Objekte. Danach können auch die anderen Schaltflächen benutzt werden.

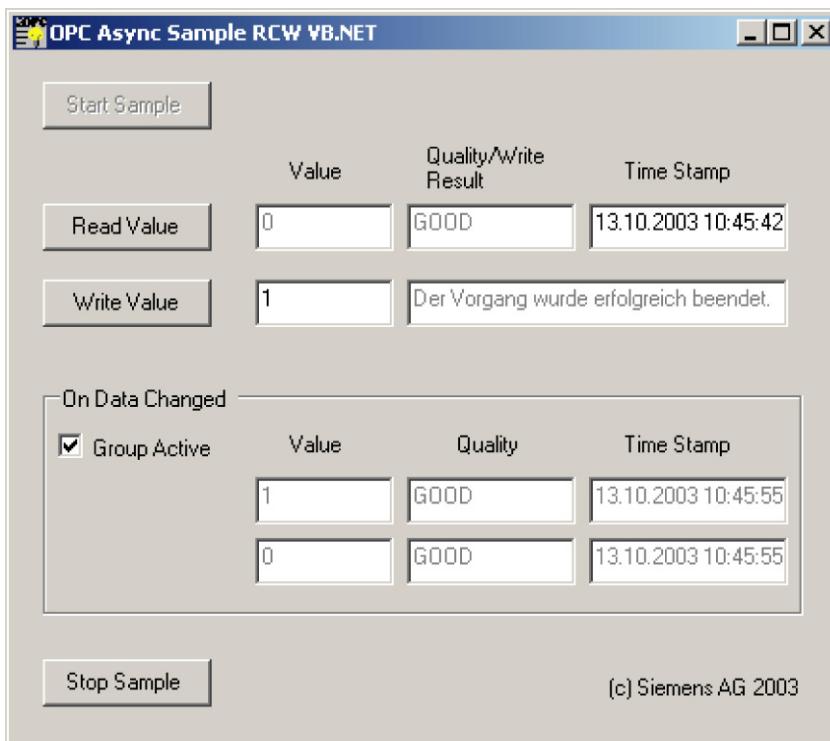


Bild 6-24 Startdialog des Beispielprogramms für die OPC-Custom-Schnittstelle (Asynchrone Kommunikation) in VB.NET mit dem .NET-Framework nach Betätigung der Schaltfläche "Start Sample"

## 6.4.2 Programmbeschreibung

### Einleitung

Die grundsätzliche Abfolge der einzelnen Programmschritte entspricht dem vorangegangenen Beispiel "OPC-Custom-Schnittstelle (Asynchrone Kommunikation) in C++ (Seite 661)". Dazu gehören der Verbindungsaufbau zum OPC-Server, das Einrichten einer Gruppe mit Variablen sowie das Lesen und Schreiben von Werten für ein Item. Die folgende Tabelle zeigt die vom Programm durchzuführenden Schritte:

Schritt	Beschreibung
1	Auswahl der .NET-Komponenten.
2	Konvertierung der ProgID in eine CLSID.
3	Verbindungsaufbau zum OPC-Server.
4	Erzeugen einer OPC-Gruppe.
5	Hinzufügen von Items.
6	Anfordern der Schnittstelle IConnectionPointContainer.
7	Anfordern der Schnittstelle IOPCAsyncIO2.
8	Callback-Objekt erzeugen.
9	OPC-Server und Callback-Objekt des Client verbinden.

Schritt	Beschreibung
10	Durchführen der gewünschten Schreib- und Leseoperation.
11	Benachrichtigungen des OPC-Servers empfangen.
12	Objekte löschen und Speicher freigeben.

### Schritt 1: Auswahl der .NET Komponenten

Um die Custom-Schnittstelle für OPC Data Access unter .NET nutzen zu können, müssen Sie in Ihrem Visual Basic-Projekt die bereits beschriebenen Runtime Callable Wrapper (RCW) einsetzen. Im Beispielprojekt wurden bereits die Komponenten *OpcRcw.Da* und *OpcRcw.Comn* hinzugefügt.

Falls diese nicht in der Auswahl erscheinen, können Sie die entsprechenden Einträge auch über die Schaltfläche *Durchsuchen* aus dem Verzeichnis "*<Installationspfad>\SIEMENS\SIMATIC.NET\opc2\bin*" auswählen.

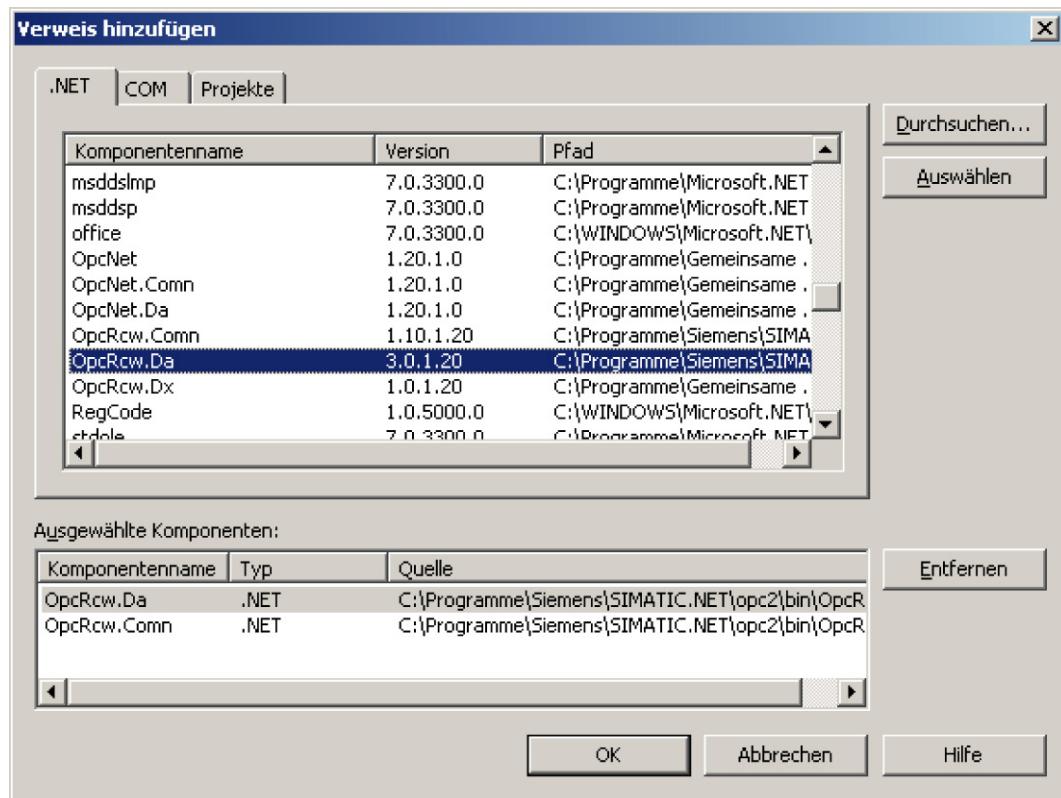


Bild 6-25 Auswahl der "Runtime Callable Wrapper" (RCW) im Visual Basic-Projekt

### Mit den Befehlen

```
Imports OpcRcw.Comn
Imports OpcRcw.Da
```

Können auf einfache Weise der Namensraum und die Methoden der OPC-Custom-Schnittstelle unter dem .NET Framework genutzt werden.

## Schritt 2: Konvertierung der ProgID in einen Typ

Jeder COM-Server besitzt zur Identifizierung eine sogenannte *ProgID*, der ein weltweit eindeutiger Typ zugeordnet ist. Dieser wird mit der Funktion *GetTypeFromProgID()* ermittelt. Die ProgID des OPC-Servers von SIMATIC NET ist L"OPC.SimaticNET":

```
Dim typeofOPCserver As Type =  
    Type.GetTypeFromProgID(L"OPC.SimaticNET")
```

## Schritt 3: Verbindungsaufbau zum OPC-Server

Die Funktion *CreateInstance()* der Klasse *Activator* erzeugt eine Instanz der Klasse, deren Typ vorgegeben wurde:

```
m_server = Activator.CreateInstance(typeofOPCserver)
```

Ergebnis dieses Programmschritts ist eine Interfacevariable *m\_server* vom Typ *IOPCSERVER*.

## Schritt 4: Erzeugen einer OPC-Gruppe

Die Schnittstelle *IOPCSERVER* besitzt die Methode *AddGroup()* zum Anlegen von Gruppen:

```
m_server.AddGroup("MyOPCGroup",  
    0,  
    250,  
    1,  
    pTimeBias,  
    pDeadband,  
    LOCALE_ID,  
    ServerGroup,  
    RevisedUpdateRate,  
    GetType(IOPCGroupStateMgt).GUID,  
    m_group)
```

Ergebnis dieses Programmschritts ist eine Gruppe mit dem vorgegebenen Namen und den gewünschten Eigenschaften. Außerdem liefert *AddGroup()* als Rückgabeparameter eine Variable *m\_group*, ein Interface auf ein Gruppenobjekt, in diesem Fall *IOPCGroupStateMgt*. Die Schnittstelle *IOPCGroupStateMgt* ist für die Nutzung der Methoden *SetState()* notwendig. Die Methode *SetState()* wird im späteren Verlauf zum Aktivieren und Deaktivieren einer Gruppe benötigt.

Aus dieser Interfacevariablen *m\_group* kann durch einfache Zuweisung die Variable *m\_item* vom Typ *IOPCItemMgt* abgeleitet werden.

```
m_item = m_group
```

## Schritt 5: Hinzufügen von Items

Die Schnittstelle *IOPCItemMgt* besitzt die Methode *AddItems()* zum Anlegen von OPC-Items:

```
m_item.AddItems(2, itemdefs, Results, pErrors)
```

Ergebnis dieses Programmschritts ist, dass der Server zwei Items mit den im Parameter *itemdefs* vorgegebenen Eigenschaften hinzugefügt. Außerdem werden die Variablen der Ergebnisstruktur *Results* (Server-Handle, Datentyp des Items auf dem Zielsystem usw.) mit Werten belegt.

Da die Ergebnisse von der unterlagerten COM Schnittstelle in unverwalteten Speicher geschrieben werden, muss der Zugriff aus verwaltetem .NET Code über die Marshal-Funktionen gemacht werden:

```
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
```

Auch die Freigabe des unverwalteten Speichers muss der Client leisten:

```
Marshal.FreeCoTaskMem(Results)
Marshal.FreeCoTaskMem(pErrors)
```

## Schritt 6: Anfordern der Schnittstelle IConnectionPointContainer

Aus der Variable *m\_group* kann die Variable *m\_ConnectionContainer* vom Typ *IConnectionPointContainer* abgeleitet werden.

```
m_ConnectionContainer = m_group
```

Die Schnittstelle wird für das Auffinden der Schnittstelle *IConnectionPoint* benötigt.

## Schritt 7: Anfordern der Schnittstellen IOPCAsyncIO2

Aus der Variable *m\_group* kann die Variable *m\_asyncIO2* vom Typ *IOPCAsyncIO2* abgeleitet werden.

```
m_asyncIO2 = m_group
```

Diese Schnittstelle bietet Methoden zum asynchronen Lesen und Schreiben von Werten.

## Schritt 8: Callback-Objekt erzeugen

Damit der OPC-Server bei asynchronen Operationen ein Ergebnis zurückliefern kann, muss im Client die Schnittstelle *IOPCDataCallback* implementiert sein.

```
Implements IOPCDataCallback
```

Die Verbindung der Schnittstelle *IConnectionPoint* und *IOPCDataCallback* des OPC-Servers wird über die Methode *FindConnectionPoint()* der Schnittstelle *IConnectionPointContainer* hergestellt. Damit wird ein Callback-Objekt für asynchrone Aufrufe erzeugt.

```
m_ConnectionContainer.FindConnectionPoint(  
    GetType(IOPCDataCallback).GUID,  
    m_ConnectionPoint)
```

### Schritt 9: OPC-Server und Callback-Objekt des Client verbinden

Die Verbindung zwischen dem Callback-Objekt des OPC-Servers und der Client-Anwendung wird durch die Methode *Advise()* hergestellt. Die RückgabevARIABLE *m\_dwCookie* ist eine Kennung für die Verbindung.

```
m_ConnectionPoint.Advise(Me, m_dwCookie)
```

### Schritt 10: Durchführen der gewünschten Schreib- bzw. Leseoperation

Auf die Methoden *Read()* und *Write()* kann über die Schnittstelle *IOPCAsyncIO2* zugegriffen werden, die in Schritt 7 erzeugt wurde:

```
m_asyncIO2.Read(1, pServer, Transaction, CancelID, pErrors)
```

### Schritt 11: Benachrichtigungen des OPC-Servers empfangen

Bei Änderungen von Daten eines aktiven Items in einer aktiven Gruppe ruft der Server die Methode *OnDataChange()* des Callback-Objekts auf. Nach dem Abschluss einer Leseoperation ruft der Server die Methode *OnReadComplete()* auf, nach dem Abschluss einer Schreiboperation wird die Methode *OnWriteComplete()* aufgerufen. Die Rückgabewerte übergibt der Server als Parameter an die jeweilige Methode.

```
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
```

### Schritt 12: Objekte löschen und Speicher freigeben

Vor dem Beenden des Programms müssen die erzeugten OPC-Objekte gelöscht und der angeforderte Speicher freigegeben werden. Die entsprechenden Funktionen sind Bestandteil der bisher benutzten Schnittstellen. *Unadvise()* beendet die Verbindung zwischen dem OPC-Server und dem Callback-Objekt:

```
m_ConnectionPoint.Unadvise(m_dwCookie)  
Marshal.ReleaseComObject(m_ConnectionPoint)  
m_item.RemoveItems(2, pItems, pErrors)  
....
```

```

Marshal.FreeCoTaskMem(pErrors)
m_server.RemoveGroup(ServerGroup, True)

```

### Freigabe von Speicher

Bei der Verwendung von COM muss unter bestimmten Umständen der Client Speicher freigeben, der vom Server angefordert wurde. Die Regeln dazu lauten:

- [out]: Der Speicher für einen Parameter mit diesem Attribut sollte von dem Server angefordert werden.
- [in, out]: Der Speicher für einen solchen Parameter sollte von dem Client angefordert werden. Der Server kann den Speicher wieder freigeben und neu allokieren.
- [in]: Der Client sollte den Speicher anfordern. Der Server ist nicht für die Freigabe des Speichers verantwortlich.

In allen drei Fällen sollte der reservierte Speicher von unverwalteten COM Objekten letztendlich vom Client freigegeben werden.

## 6.5 OPC-Custom-Schnittstelle (Synchrone Kommunikation) in C#

### Voraussetzungen für die Verwendung des Beispielprogramms

Nach der Installation der SIMATIC NET-Software finden Sie dieses Programm auf Ihrer Festplatte in folgendem Verzeichnis:

"<Installationspfad>\SIEMENS\SIMATIC.NET\opc2\samples\dotnet\C#\sync.net"

Auf dem Rechner, auf dem das Beispielprogramm ablaufen soll, muss ein .NET-Framework ab Version 4.0 installiert sein. Außerdem müssen Sie eine Simulationsverbindung aktivieren, damit die im Programm verwendeten Demo-Variablen verfügbar sind (siehe Kapitel "Aktivieren der Simulationsverbindung (Seite 662)").

### Runtime Callable Wrapper

Das vorliegende Beispiel nutzt die Custom-Schnittstelle für OPC Data Access V2.0 mit dem .NET Framework. Der Zugriff auf die COM-Schnittstelle aus verwaltetem Code von .NET-Anwendungsprogrammen erfolgt über den Runtime Callable Wrapper (RCW) der OPC Foundation.

Die primäre Funktion von Runtime Callable Wrappern besteht im Marshallen von Aufrufen zwischen einem .NET-Client und einem unverwalteten COM-Objekt. Unter Marshallen versteht man das Verfügarmachen von Speicherbereichen.

### 6.5.1 Bedienung des Beispielprogramms

Die Anwendung enthält mehrere Bedienelemente, die jeweils folgende Aktionen auslösen:

Bedienelement	Wirkung
Start Sample	Anwendung starten
Read Item bzw. Write Value	Item lesen bzw. Wert schreiben
Stop Sample	Anwendung beenden

Beim Programmstart ist nur die Schaltfläche "Start Sample" aktiviert. Nach dem Betätigen dieser Schaltfläche erzeugt die Anwendung die notwendigen OPC-Objekte. Danach werden die übrigen Schaltflächen freigegeben.

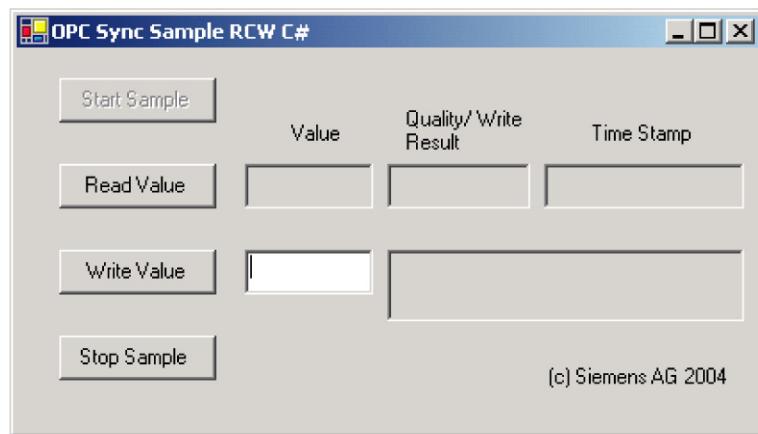


Bild 6-26 Startdialog des Beispielprogramms für die OPC-Custom-Schnittstelle (Synchrone Kommunikation) in C# mit dem .NET-Framework nach Betätigung der Schaltfläche "Start Sample"

### 6.5.2 Programmbeschreibung

#### Einleitung

Die grundsätzliche Abfolge der einzelnen Programmschritte entspricht dem vorangegangenen Beispiel "OPC-Custom-Schnittstelle (Synchrone Kommunikation) in C++ (Seite 639)". Dazu gehören der Verbindungsaufbau zum OPC-Server, das Einrichten einer Gruppe mit Variablen sowie das Lesen und Schreiben von Werten für ein Item. Die folgende Tabelle zeigt die vom Programm durchzuführenden Schritte:

Schritt	Beschreibung
1	Auswahl der .NET-Komponenten
2	Konvertierung der ProgID in eine CLSID
3	Verbindungsaufbau zum OPC-Server
4	Erzeugen einer OPC-Gruppe

Schritt	Beschreibung
5	Hinzufügen von Items
6	Anfordern der Schnittstelle IOPCSyncIO
7	Durchführen der gewünschten Schreib- und Leseoperation
8	Objekte löschen und Speicher freigeben

### Schritt 1: Auswahl der .NET Komponenten

Siehe Kapitel "Programmbeschreibung (Seite 700)" Schritt 1

### Schritt 2: Konvertierung der ProgID in eine CLSID

Jeder COM-Server besitzt zur Identifizierung eine sogenannte *ProgID*, der ein weltweit eindeutiger Typ zugeordnet ist. Dieser wird mit der Funktion *GetTypeFromProgID()* ermittelt. Die ProgID des OPC-Servers von SIMATIC NET ist L"OPC.SimaticNET":

```
Type svrComponenttyp = Type.GetTypeFromProgID("OPC.SimaticNET");
```

### Schritt 3: Verbindungsauftbau zum OPC-Server

Die Funktion *CreateInstance()* der Klasse Activator erzeugt eine Instanz der Klasse, deren Typ vorgegeben wurde:

```
pIOPCServer =
    (IOPCServer)Activator.CreateInstance(svrComponenttyp);
```

Ergebnis dieses Programmschritts ist eine Interfacevariable *pIOPCServer* vom Typ IOPCServer.

### Schritt 4: Erzeugen einer OPC-Gruppe

Die Schnittstelle IOPCServer besitzt die Methode *AddGroup()* zum Anlegen von Gruppen:

```
pIOPCServer.AddGroup(GROUP_NAME,
    0,
    dwRequestedUpdateRate,
    hClientGroup,
    hTimeBias..AddrOfPinnedObject(),
    hDeadband.AddrOfPinnedObject(),
    dwLCID,
    out pSvrGroupHandle,
    out pRevUpdateRate,
    ref out pobjGroup1)
```

#### Hinweis

Die Speicherzuordnung für hDeadband und hTimeBias erfolgt durch die Funktion GCHandle.Alloc():

```
GCHandle hTimeBias;
hTimeBias = GCHandle.Alloc(timebias, GCHandleType.Pinned);
```

```
GCHandle hDeadband;  
hDeadband = GCHandle.Alloc(deadband,GCHandleType.Pinned);
```

Mit diesen Funktionen wird der unverwaltete Speicher der Übergabevariablen timebias und deadband (Percent Deadband) behandelt.

Die Verwaltungs-Handles müssen folgendermaßen freigegeben werden, wenn sie nicht mehr benötigt werden:

```
if (hTimeBias.IsAllocated) hTimeBias.Free();  
if (hDeadband.IsAllocated) hDeadband.Free();
```

Ergebnis dieses Programmschritts ist eine Gruppe mit dem vorgegebenen Namen und den gewünschten Eigenschaften. Außerdem liefert *AddGroup()* als Rückgabeparameter eine Variable *pobjGroup1*, ein Interface auf ein Gruppenobjekt.

Durch den Typanpassungsauftruf *IOPCGroupStateMgt* kann in einfacher Form die Schnittstelle aus der zurückgegebenen Gruppenschnittstelle *pobjGroup1* erhalten werden.

Dieser Aufruf entspricht vereinfacht der COM Methode *QueryInterface()*.

Die Methode *SetState()* der Schnittstelle *IOPCGroupStateMgt* wird im späteren Verlauf zum Aktivieren und Deaktivieren einer Gruppe benötigt.

```
pIOPCGroupStateMgt = (IOPCGroupStateMgt)pobjGroup1;
```

## Schritt 5: Hinzufügen von Items

Die Schnittstelle *IOPCItemMgt* besitzt die Methode *AddItems()* zum Anlegen von OPC-Items:

```
((IOPCItemMgt)pobjGroup1).AddItems(1, ItemDefArray, out pResults, out  
pErrors);
```

Ergebnis dieses Programmschritts ist, dass der Server ein Item mit den im Parameter *itemdefs* vorgegebenen Eigenschaften hinzugefügt. Außerdem werden die Variablen der Ergebnisstruktur *OPCITEMRESULT* (Server-Handle, Datentyp des Items auf dem Zielsystem usw.) mit Werten belegt.

Da die Ergebnisse von der unterlagerten COM Schnittstelle in den unverwalteten Speicher geschrieben werden, muss der Zugriff aus verwaltetem .NET Code über die Marshal-Funktionen gemacht werden:

```
OPCITEMRESULT result =  
    OPCITEMRESULT)Marshal.PtrToStructure(pResults,  
    typeof(OPCITEMRESULT));
```

Auch die Freigabe des unverwalteten Speichers muss der Client leisten:

```
Marshal.FreeCoTaskMem(Results)  
Marshal.FreeCoTaskMem(pErrors)
```

## Schritt 6: Anfordern der Schnittstelle *IOPCSyncIO*

Aus der Variable *pIOPCSyncIO* kann die Variable *pobjGroup1* vom Typ *pIOPCSyncIO* abgeleitet werden.

```
pIOPCSyncIO2 = (IOPCSyncIO2)pobjGroup1;
```

Die Schnittstelle bietet Methoden für synchrones Schreiben und Lesen von Werten.

### Schritt 7: Durchführen der gewünschten Schreib- bzw. Leseoperation

Mittels der Methoden *Read()* und *Write()* kann über die Schnittstelle IOPCSyncIO auf Werte von OPC-Items zugegriffen werden:

```
pIOPCSyncIO.Read(OPCDataSource.OPC_DS_DEVICE, 1, nItemSvrID, out
                  pItemValues, out pErrors);

pIOPCSyncIO.Write(1, nItemSvrID, values, out pErrors);
```

Da nach dem Lesen die Ergebnisse von der unterlagerten COM Schnittstelle in den unverwalteten Speicher geschrieben werden, muss der Zugriff aus veraltetem .NET Code über die Marshal-Funktionen durchgeführt werden:

```
OPCITEMSTATE pItemState =
    (OPCITEMSTATE)Marshal.PtrToStructure(pItemValues,
                                         typeof(OPCITEMSTATE));
```

Auch die Freigabe des unverwalteten Speichers muss der Client leisten:

```
Marshal.FreeCoTaskMem(pItemValues);
Marshal.FreeCoTaskMem(pErrors);
```

### Schritt 8: Objekte löschen und Speicher freigeben

Vor dem Beenden des Programms oder falls Stop betätigt wurde, müssen die erzeugten OPC-Objekte gelöscht und der angeforderte Speicher freigegeben werden.

```
Marshal.ReleaseComObject(pIOPCSyncIO);
pIOPCServer.RemoveGroup(nSvrGroupID, 0);
Marshal.ReleaseComObject(pobjGroup1);
Marshal.ReleaseComObject(pIOPCServer);
```

#### Freigabe von Speicher

Bei der Verwendung von COM muss unter bestimmten Umständen der Client Speicher freigeben, der vom Server angefordert wurde. Die Regeln dazu lauten:

- [out]: Der Speicher für einen Parameter mit diesem Attribut sollte von dem Server angefordert werden.
- [in, out]: Der Speicher für einen solchen Parameter sollte von dem Client angefordert werden. Der Server kann den Speicher wieder freigeben und neu allokieren.
- [in]: Der Client sollte den Speicher anfordern. Der Server ist nicht für die Freigabe des Speichers verantwortlich.

In allen drei Fällen sollte der reservierte Speicher von unverwalteten COM Objekten letztendlich vom Client freigegeben werden.

## **6.6        OPC-Custom-Schnittstelle (Asynchrone Kommunikation) in C#**

### **Voraussetzungen für die Verwendung des Beispielprogramms**

Nach der Installation der SIMATIC NET-Software finden Sie dieses Programm auf Ihrer Festplatte in folgendem Verzeichnis:

"<Installationspfad>\SIEMENS\SIMATIC.NET\opc2\samples\dotnet\C#\async.net"

Auf dem Rechner, auf dem das Beispielprogramm ablaufen soll, muss ein .NET-Framework ab Version 4.0 installiert sein. Außerdem müssen Sie eine Simulationsverbindung aktivieren, damit die im Programm verwendeten Demo-Variablen verfügbar sind (siehe Kapitel "Aktivieren der Simulationsverbindung (Seite 662)").

### **Runtime Callable Wrapper**

Das vorliegende Beispiel nutzt die Custom-Schnittstelle für OPC Data Access V2.0 mit dem .NET Framework. Der Zugriff auf die COM-Schnittstelle aus verwaltetem Code von .NET-Anwendungsprogrammen erfolgt über den Runtime Callable Wrapper (RCW) der OPC Foundation.

Die primäre Funktion von Runtime Callable Wrappern besteht im Marshallen von Aufrufen zwischen einem .NET-Client und einem unverwalteten COM-Objekt. Unter Marshallen versteht man das Verfügbarmachen von Speicherbereichen.

### **6.6.1      Bedienung des Beispielprogramms**

Das Programm enthält mehrere Bedienelemente, die jeweils folgende Aktionen auslösen:

<b>Bedienelement</b>	<b>Wirkung</b>
Start Sample	Programm starten
Read Item bzw. Write Value	Item lesen bzw. Wert schreiben
Group Active	Gruppe aktivieren bzw. deaktivieren
Stop Sample	Programm beenden

Beim Programmstart ist nur die Schaltfläche "Start Sample" aktiviert. Nach dem Betätigen dieser Schaltfläche erzeugt das Programm die notwendigen OPC-Objekte. Danach werden die übrigen Schaltflächen freigegeben.

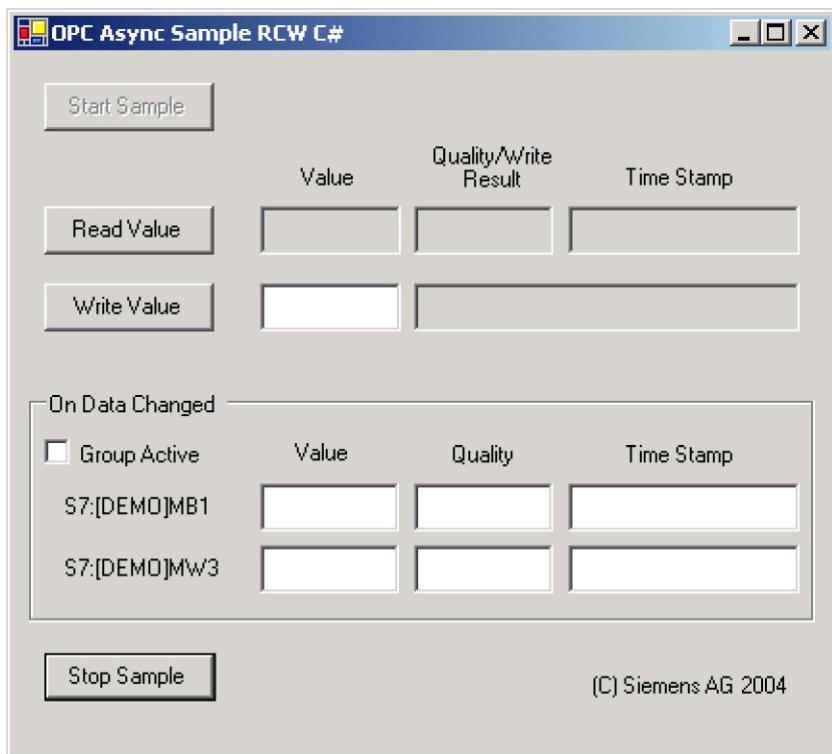


Bild 6-27 Startdialog des Beispielprogramms für die OPC-Custom-Schnittstelle (Asynchrone Kommunikation) in C# mit dem .NET-Framework nach Betätigung der Schaltfläche "Start Sample"

## 6.6.2 Programmbeschreibung

### Einleitung

Die grundsätzliche Abfolge der einzelnen Programmschritte entspricht dem vorangegangenen Beispiel "OPC-Custom-Schnittstelle (Asynchrone Kommunikation) in VB.NET (Seite 698)". Dazu gehören der Verbindungsaufbau zum OPC-Server, das Einrichten einer Gruppe mit Variablen sowie das Lesen und Schreiben von Werten für ein Item. Die folgende Tabelle zeigt die vom Programm durchzuführenden Schritte:

Schritt	Beschreibung
1	Auswahl der .NET-Komponenten
2	Konvertierung der ProgID in eine CLSID
3	Verbindungsaufbau zum OPC-Server
4	Erzeugen einer OPC-Gruppe
5	Hinzufügen von Items
6	Anfordern der Schnittstelle IConnectionPointContainer
7	Anfordern der Schnittstelle IOPCAsyncIO2
8	Callback-Objekt erzeugen

Schritt	Beschreibung
9	OPC-Server und Callback-Objekt des Client verbinden
10	Durchführen der gewünschten Schreib- und Leseoperation
11	Benachrichtigungen des OPC-Servers empfangen
12	Objekte löschen und Speicher freigeben

### Schritt 1: Auswahl der .NET Komponenten

Siehe Kapitel "Programmbeschreibung (Seite 700)" Schritt 1

### Schritt 2: Konvertierung der ProgID in eine CLSID

Jeder COM-Server besitzt zur Identifizierung eine sogenannte *ProgID*, der ein weltweit eindeutiger Typ zugeordnet ist. Dieser wird mit der Funktion *GetTypeFromProgID()* ermittelt. Die ProgID des OPC-Servers von SIMATIC NET ist L"OPC.SimaticNET":

```
Type svrComponenttyp = Type.GetTypeFromProgID(L"OPC.SimaticNET");
```

### Schritt 3: Verbindungsaufbau zum OPC-Server

Die Funktion *CreateInstance()* der Klasse Activator erzeugt eine Instanz der Klasse, deren Typ vorgegeben wurde:

```
pIOPCServer =
    (IOPCServer)Activator.CreateInstance(svrComponenttyp);
```

Ergebnis dieses Programmschritts ist eine Interfacevariable *pIOPCServer* vom Typ *IOPCServer*.

### Schritt 4: Erzeugen einer OPC-Gruppe

Die Schnittstelle IOPCServer besitzt die Methode *AddGroup()* zum Anlegen von Gruppen:

```
pIOPCServer.AddGroup(GROUP_NAME,
    0,
    dwRequestedUpdateRate,
    hClientGroup,
    hTimeBias..AddrOfPinnedObject(),
    hDeadband.AddrOfPinnedObject(),
    dwLCID,
    out pSvrGroupHandle,
    out pRevUpdateRate,
    ref out pobjGroup1)
```

#### Hinweis

Die Speicherzuordnung für hDeadband und hTimeBias erfolgt durch die Funktion GCHandle.Alloc():

```
GCHandle hTimeBias;
hTimeBias = GCHandle.Alloc(timebias,GCHandleType.Pinned);
```

```
GCHandle hDeadband;
hDeadband = GCHandle.Alloc(deadband, GCHandleType.Pinned);
```

Mit diesen Funktionen wird der unverwaltete Speicher der Übergabevariablen timebias und deadband (Percent Deadband) behandelt.  
Die Verwaltungs-Handles müssen folgendermaßen freigegeben werden, wenn sie nicht mehr benötigt werden:

```
if (hTimeBias.IsAllocated) hTimeBias.Free();
if (hDeadband.IsAllocated) hDeadband.Free();
```

Ergebnis dieses Programmschritts ist eine Gruppe mit dem vorgegebenen Namen und den gewünschten Eigenschaften. Außerdem liefert *AddGroup()* als Rückgabeparameter eine Variable *pobjGroup1*, ein Interface auf ein Gruppenobjekt.

Durch den Typanpassungsauftrag *IOPCGroupStateMgt* kann in einfacher Form die Schnittstelle aus der zurückgegebenen Gruppenschnittstelle *pobjGroup1* erhalten werden. Dieser Aufruf entspricht vereinfacht der COM Methode *QueryInterface()*. Die Methode *SetState()* der Schnittstelle *IOPCGroupStateMgt* wird im späteren Verlauf zum Aktivieren und Deaktivieren einer Gruppe benötigt.

```
pIOPCGroupStateMgt = (IOPCGroupStateMgt)pobjGroup1;
```

## Schritt 5: Hinzufügen von Items

Die Schnittstelle *IOPCItemMgt* besitzt die Methode *AddItems()* zum Anlegen von OPC-Items:

```
((IOPCItemMgt)pobjGroup1).AddItems(2, ItemDefArray, out
pResults, out pErrors);
```

Ergebnis dieses Programmschritts ist, dass der Server zwei Items mit den im Parameter *itemdefs* vorgegebenen Eigenschaften hinzugefügt. Außerdem werden die Variablen der Ergebnisstruktur *OPCITEMRESULT* (Server-Handle, Datentyp der Items auf dem Zielsystem usw.) mit Werten belegt.

Da die Ergebnisse von der unterlagerten COM Schnittstelle in den unverwalteten Speicher geschrieben werden, muss der Zugriff aus verwaltetem .NET Code über die Marshal-Funktionen gemacht werden:

```
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;
```

Auch die Freigabe des unverwalteten Speichers muss der Client leisten:

```
Marshal.FreeCoTaskMem(Results);  
Marshal.FreeCoTaskMem(pErrors);
```

### Schritt 6: Anfordern der Schnittstelle IConnectionPointContainer

Aus der Variable *m\_group1* kann die Variable *pIConnectionPointContainer* vom Typ *IConnectionPointContainer* abgeleitet werden.

```
pIConnectionPointContainer =  
(IConnectionPointContainer)pobjGroup1;
```

Die Schnittstelle wird für das Auffinden der Schnittstelle *IConnectionPoint* benötigt.

### Schritt 7: Anfordern der Schnittstellen IOpcAsyncIO2

Aus der Variable *pobjGroup1* kann die Variable *pIOPCAsyncIO2* vom Typ *pIOPCAsyncIO2* abgeleitet werden.

```
pIOPCAsyncIO2 = (IOpcAsyncIO2)pobjGroup1;
```

Diese Schnittstelle bietet Methoden zum asynchronen Lesen und Schreiben von Werten.

### Schritt 8: Callback-Objekt erzeugen

Damit der OPC-Server bei asynchronen Operationen ein Ergebnis zurückliefern kann, muss im Client die Schnittstelle *IOPCDataCallback* implementiert sein.

```
public class OPCAsync : System.Windows.Forms.Form ,  
    IOpcDataCallback
```

Die Verbindung der Schnittstelle *IConnectionPoint* und *IOPCDataCallback* des OPC-Servers wird über die Methode *FindConnectionPoint()* der Schnittstelle *IConnectionPointContainer* hergestellt. Damit wird ein Callback-Objekt für asynchrone Aufrufe erzeugt.

```
Guid iid = typeof(IOpcDataCallback).GUID;  
pIConnectionPointContainer.FindConnectionPoint(ref iid,  
                                                out pIConnectionPoint);
```

### Schritt 9: OPC-Server und Callback-Objekt des Client verbinden

Die Verbindung zwischen dem Callback-Objekt des OPC-Servers und der Client-Anwendung wird durch die Methode *Advise()* hergestellt. Die RückgabevARIABLE *dwCookie* ist eine Kennung für die Verbindung.

```
pIConnectionPoint.Advise(this,out dwCookie);
```

### Schritt 10: Durchführen der gewünschten Schreib- bzw. Leseoperation

Auf die Methoden *Read()* und *Write()* kann über die Schnittstelle *IOpcAsyncIO2* zugegriffen werden, die in Schritt 7 erzeugt wurde:

```
piOPCAsyncIO2.Read(1, ItemSrvHandleArray, nTransactionID+1,
                    out nCancelid, out pErrors);
```

### Schritt 11: Benachrichtigungen des OPC-Servers empfangen

Bei Änderungen von Daten eines aktiven Items in einer aktiven Gruppe ruft der Server die Methode *OnDataChange()* des Callback-Objekts auf. Nach dem Abschluss einer Leseoperation ruft der Server die Methode *OnReadComplete()* auf, nach dem Abschluss einer Schreiboperation wird die Methode *OnWriteComplete()* aufgerufen. Die Rückgabewerte übergibt der Server als Parameter an die jeweilige Methode.

```
virtual void OnDataChange(
    Int32 dwTransid,
    Int32 hGroup,
    Int32 hrMasterquality,
    Int32 hrMastererror,
    Int32 dwCount,
    int[] phClientItems,
    object[] pvValues,
    short[] pwQualities,
    FILETIME[] pftTimeStamps,
    int[] pErrors,
)
```

### Schritt 12: Objekte löschen und Speicher freigeben

Vor dem Beenden des Programms oder falls Stop betätigt wurde, müssen die erzeugten OPC-Objekte gelöscht und der angeforderte Speicher freigegeben werden.

```
pIConnectionPoint.Unadvise(dwCookie);
Marshal.ReleaseComObject(pIConnectionPoint);
Marshal.ReleaseComObject(pIConnectionPointContainer);
Marshal.ReleaseComObject(piOPCAsyncIO2);
Marshal.ReleaseComObject(piOPCGroupStateMgt);
Marshal.ReleaseComObject(pobjGroup1);
Marshal.ReleaseComObject(piOPCServer);
```

#### Freigabe von Speicher

Bei der Verwendung von COM muss unter bestimmten Umständen der Client Speicher freigeben, der vom Server angefordert wurde. Die Regeln dazu lauten:

- [out]: Der Speicher für einen Parameter mit diesem Attribut sollte von dem Server angefordert werden.
- [in, out]: Der Speicher für einen solchen Parameter sollte von dem Client angefordert werden. Der Server kann den Speicher wieder freigeben und neu allokieren.
- [in]: Der Client sollte den Speicher anfordern. Der Server ist nicht für die Freigabe des Speichers verantwortlich.

In allen drei Fällen sollte der reservierte Speicher von unverwalteten COM Objekten letztendlich vom Client freigegeben werden.

## **6.7 OPC-XML-Schnittstelle in C#**

Das vorliegende Beispiel in der Programmiersprache C# nutzt die XML-Schnittstelle für Data Access von OPC. Es enthält Methoden, um eine Verbindung zu einem Web-Dienst herzustellen und um Lese- und Schreibaufträge durchzuführen.

### **Voraussetzungen für die Verwendung des Beispielprogramms**

Auf dem Rechner, auf dem das Beispielprogramm ablaufen soll, müssen Sie zunächst die S7-Simulationsbaugruppe und das S7-Protokoll aktivieren.

Weiterhin müssen Sie den Web-Dienst einrichten.

### **6.7.1 Bedienung des Beispielprogramms**

#### **Programm starten**

Nach dem Aufruf des Programms sind alle Schaltflächen bis auf "Start Sample" deaktiviert. Klicken Sie diese Schaltfläche, um eine Verbindung zum OPC XML-Web-Dienst herzustellen. Wenn der Web-Dienst verfügbar ist, zeigt ein Dialogfeld Informationen über den Server. Andernfalls wird eine Fehlermeldung angezeigt.

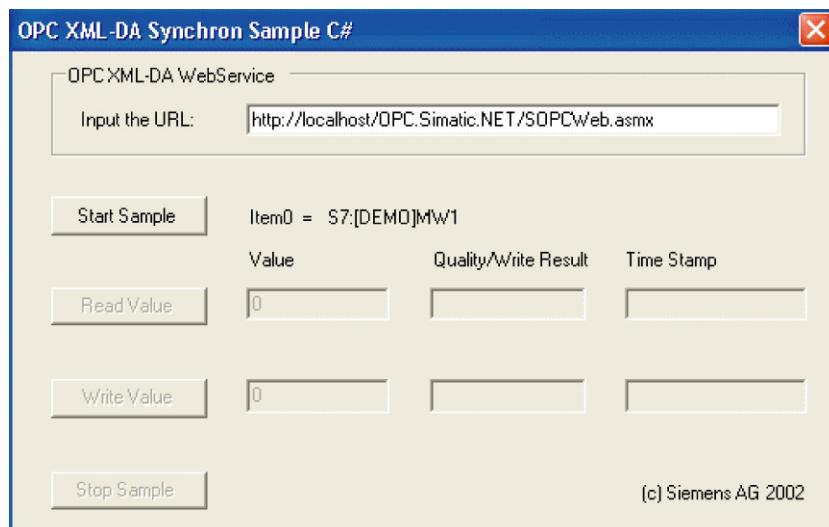


Bild 6-28 Startdialog des Beispielprogramms (C#) für den OPC Data Access über die XML-Schnittstelle vor Betätigung der Schaltfläche "Start Sample"

## Werte lesen

Klicken Sie die Schaltfläche "Read Value", um den Wert des Items *S7:[DEMO]MW1* zu lesen. Falls der Leseauftrag erfolgreich durchgeführt werden konnte, zeigt das Programm in den Textfeldern neben der Schaltfläche "Read Value" den Wert des Items, eine Angabe für die Qualität und einen Zeitstempel an.

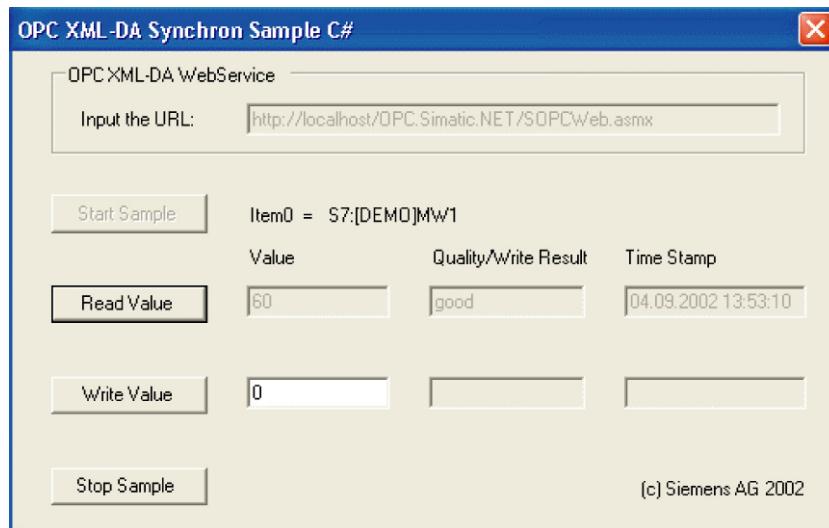


Bild 6-29 Anzeige der gelesenen Werte nach Betätigung der Schaltfläche "Read Value"

## Werte schreiben

Tragen Sie im Textfeld neben der Schaltfläche "Write Value" den zu schreibenden Wert ein und klicken Sie die Schaltfläche "Write Value". Falls der Schreibauftrag erfolgreich durchgeführt werden konnte, zeigt das Programm im Textfeld "Quality/Write Result" eine Angabe über die Qualität des Werts und im Textfeld "Time Stamp" den Zeitstempel.

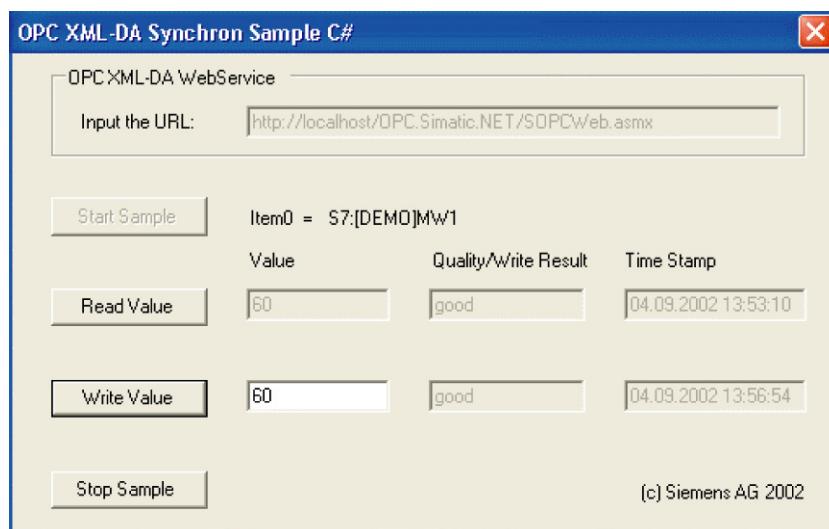


Bild 6-30 Anzeigen zu Qualität des Werts und Zeitstempel der geschriebenen Werte nach Betätigung der Schaltfläche "Write Value"

## **Programm beenden**

Beenden Sie das Programm, indem Sie die Schaltfläche "Stop Sample" anklicken. Dadurch werden die Schaltflächen "Read Value", "Write Value" und "Stop Sample" deaktiviert. Die Schaltfläche "Start Sample" sowie das Textfeld für die URL werden wieder aktiviert.

### **6.7.2 Web-Dienst zum Projekt hinzufügen**

#### **Einstellungen in Visual Studio .NET**

Damit das Beispielprogramm den Web-Dienst der OPC XML-Schnittstelle nutzen kann, wurde im entsprechenden Projekt in Visual Studio .NET eine "Web Reference" eingebunden.

Wenn Sie selbst Programme mit OPC XML-Anbindung erstellen wollen, müssen Sie dazu folgende Arbeitsschritte durchführen:

1. Rufen Sie in "Microsoft Visual Studio" das Menü "Projekt" > "Dienstverweis hinzufügen" auf.
2. Klicken Sie im Dialogfeld "Dienstverweis hinzufügen" auf die Schaltfläche "Erweitert".
3. Klicken Sie im Dialogfeld "Dienstverweiseinstellungen" auf die Schaltfläche "Webverweis hinzufügen"

4. Geben Sie im Eingabefeld "URL" die Webadresse des SIMATIC NET OPC XML-Webdienstes an. Diese URL hat folgende Form:  
"http://<Rechneradresse>/<Name des OPC SIMATIC NET Webdienstes>/SOPCWeb.asmx?wsdl"
5. Klicken Sie auf die Schaltfläche "Verweis hinzufügen".

Im Fenster "OPCXML\_DataAccess Beschreibung" sind die Methoden aufgeführt, die Ihnen zur Verfügung stehen.

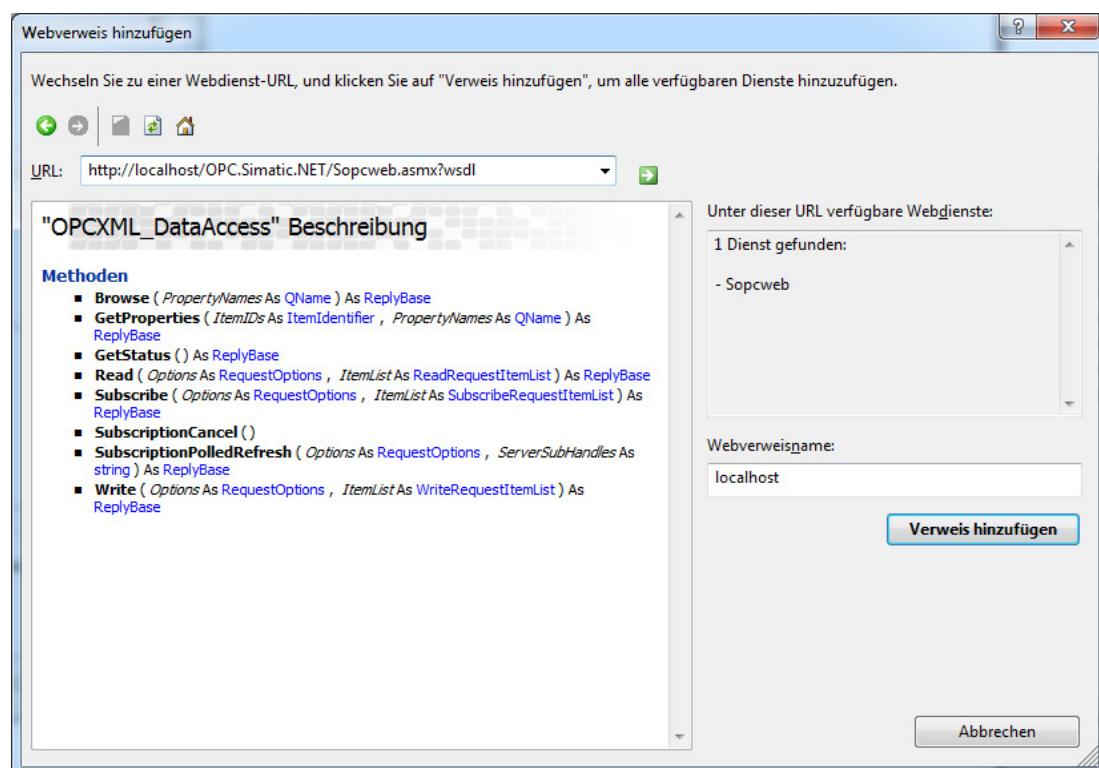


Bild 6-31 Das Fenster "Webverweis hinzufügen" mit der Auflistung der in der WSDL definierten Methoden

### 6.7.3 Die Klasse MainForm

#### Programmcode

Alle Methoden des Beispielprogramms sind in der Klasse *MainForm* definiert. Am Anfang werden die Elemente des Dialogfeldes deklariert.

```
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;
```

Der Web-Dienst ist eine Instanz der Klasse OPCXML\_DataAccess:

```
// proxy class for the webservice
private OPCXML_DataAccess m_OPCTML_DataAccess;
```

Der Name des Item wird in der Variablen m\_strItemName abgelegt:

```
// ItemID used in this sample
private string m_strItemName = "S7:[DEMO]MW1";
```

## 6.7.4 Der Konstruktor von MainForm und die Methode Dispose

### Programmcode

Der Konstruktor der Dialogfeldklasse ruft die Methode *InitializeComponent()* auf. Dort werden sämtliche Elemente des Dialogfelds erzeugt. Der Inhalt von *InitializeComponent()* wird von Visual Studio .NET automatisch erzeugt, wenn das Dialogfeld mit dem Form Designer erstellt wird.

```
public MainForm()
{
    // Required for Windows Form Designer support
    //
    InitializeComponent();
    //
    // TODO: Add any constructor code after
    // InitializeComponent call
```

Der Item-Name erscheint im Dialogfeld, weil eine Instanz der Klasse *System.Windows.Forms.Label* mit dem Inhalt der Variablen *m\_strItemName* initialisiert wird:

```
    Label_Item_Value.Text = m_strItemName;
}
```

Die Methode *Dispose()* gibt beim Programmende alle verwendeten Ressourcen frei:

```
/// <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 Erzeugen der Dialogfeldelemente

### Programmcode

Der folgende Abschnitt wird automatisch von Visual Studio .NET erzeugt, wenn der Anwender das Hauptdialogfeld der Anwendung mit dem Form Designer erstellt. Hier sind Informationen zu alle Elementen des Dialogfeldes hinterlegt, wie beispielsweise Größe, Lage und Beschriftung von Schaltflächen. Außerdem ist hier festgelegt, welche Ereignisprozeduren beim Betätigen einer Schaltfläche ausgeführt werden.

```
#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();
    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 = ";  
//  
// Label_Item_Value  
//
```

```
this.Lable_Item_Value.Location = new System.Drawing.
    Point(192, 94);
this.Lable_Item_Value.Name = "Lable_Item_Value";
this.Lable_Item_Value.Size = new System.Drawing. Size(296, 23);
this.Lable_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";
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.Lable_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);
}
#endif
```

## 6.7.6 Die Methode Main

### Programmcode

Die Methode Main ist der Startpunkt für das Beispielprogramm. Es wird eine Instanz der Dialogfeldklasse erzeugt und über den Aufruf von Application.Run gestartet.

```
/// <summary>
/// The main entry point for the application.
```

```
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new MainForm());
}
```

## 6.7.7 Die Methode Button\_Start\_Sample\_Click

### Programmcode

Diese Methode wird ausgeführt, wenn Sie die Schaltfläche "Start Sample" anklicken. Der OPC XML-Web-Dienst wird durch eine Instanz der Klasse OPCXML\_DataAccess repräsentiert.

```
/*
 | Name:  Button_Start_Sample_Click
 | Desc:  Handler is being called, when button "Start Sample"
 |        has been 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_DataAccess == null)
            { m_OPCTML_DataAccess = new OPCTML_DataAccess ();
              m_OPCTML_DataAccess.Timeout = 10000; //Timeout 10 sec.
            }
    }
```

Das Programm prüft die Verbindung zum Web-Dienst, indem es Methoden der Klasse *WebRequest* aufruft. Wenn der Web-Dienst nicht verfügbar ist, wird eine Exception ausgelöst.

```
// 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();
```

Mit der Methode *GetStatus* fragt das Programm den Serverstatus ab. Hier wird zum ersten Mal eine Methode der XML-Schnittstelle verwendet.

```
// Assiging the url to the proxy class for the
// web service
m_OPCTML_DataAccess.Url = Edit_URL.Text;

//Checking the webservice status
```

```
ServerStatus Status;
ReplyBase replay = m_OPCTXML_DataAccess.GetStatus ("en", "1", out
Status);
```

Wenn der Server ordnungsgemäß läuft, gibt das Programm einen Dialog mit verschiedenen Informationen aus.

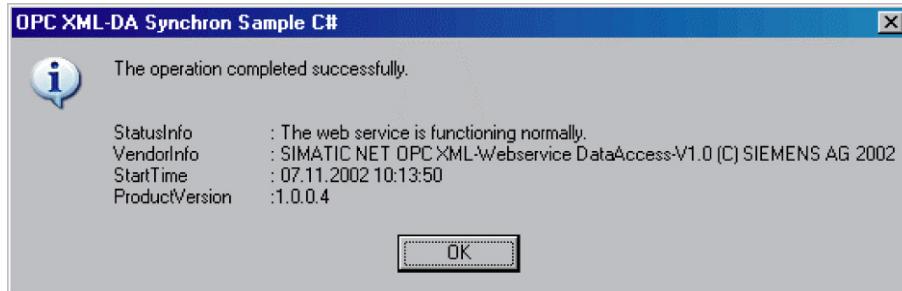


Bild 6-32 Informations-Dialog nach erfolgreichem Anlauf eines Web-Dienstes

```
if (replay.ServerState == serverState.running)
{
    string strText =
        ""The operation completed successfully.\n\n";
    if (Status.StatusInfo != null)
        { strText+= "\nStatusInfo\t: " + Status.StatusInfo;
        }
    if (Status.VendorInfo != null)
        { strText+= "\nVendorInfo\t: " + Status.VendorInfo;
        }
    strText+= "\nStartTime\t\t: " +
    Status.StartTime.ToString();
    if (Status.ProductVersion != null)
        { strText+= "\nProductVersion\t:"+
            + Status.ProductVersion;
        }
    MessageBox.Show (this, strText, this.Text,
    MessageBoxButtons.OK, MessageBoxIcon.Information);
```

Das Textfeld zur Eingabe der URL und die Schaltfläche "Start Sample" werden deaktiviert. Die restlichen Schaltflächen werden aktiviert.

```
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;
}
```

Falls beim Web-Dienst ein Fehler ausgetreten ist, wird er in einem Dialogfeld angezeigt:

```
        else
        {
            string strText =
                "The operation is not completed successfully.\n\n";
            MessageBox.Show (this, strText, this.Text,
                MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        }
    }
```

Wenn beim Ablauf des Programms ein Fehler ausgetreten ist, wird dieser innerhalb eines Catch-Blocks behandelt.

```
    catch (Exception excep)
    {
        MessageBox.Show (this, excep.Message, this.Text,
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
    }
}
```

## 6.7.8 Die Methode Button\_Stop\_Sample\_Click

### Programmcode

Diese Methode wird ausgeführt, wenn Sie die Schaltfläche "Stop Sample" anklicken. Es wird lediglich das Aussehen des Diagfeldes geändert, das Textfeld für die URL ist wieder editierbar und die Schaltfläche "Start Sample" kann angeklickt werden. Alle anderen Schaltflächen werden deaktiviert.

```
/*
| 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 Die Methode Button\_Read\_Value\_Click

### Programmcode

Diese Methode wird ausgeführt, wenn Sie die Schaltfläche "Read Value" anklicken. Zunächst erzeugt das Programm alle für den Leseauftrag notwendigen Objekte, ein Array des Typs *ReadRequestItemList*s und ein Array des Typs *ReadRequestItem*, jeweils mit einem Element. Außerdem sind je eine Instanz der Klasse *RequestOptions* sowie der Klasse *OPCError* als Rückgabeparameter vorzusehen.

```
-----  
| Name: Button_Read_Value_Click  
| Desc: Handler is being called, when button "Read Value"  
|       has been pressed  
| Notes: initiates an 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;
```

Die Methode *Read* der XML-Schnittstelle wird mit den zuvor definierten Parametern aufgerufen. Wenn ein Wert gelesen werden konnte, zeigt das Dialogfeld diesen im Textfeld "Value" an. Andernfalls wird "0" angezeigt.

```
m_OPCTextDataAccess.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";  
    }
```

Auch die gelesenen Werte für den Zeitstempel und die Qualität der Werte erscheinen im Dialogfeld in den entsprechenden Textfeldern.

```
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);
}
```

Programmfehler werden in einem Catch-Block behandelt.

```
catch (Exception excep)
{
    MessageBox.Show (this, excep.Message, this.Text,
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
}
```

### 6.7.10 Die Methode Button\_Write\_Value\_Click

#### Programmcode

Diese Methode wird ausgeführt, wenn Sie die Schaltfläche "Write Value" anklicken. Zunächst erzeugt das Programm alle für den Schreibauftrag notwendigen Objekte, ein Array des Typs *WriteRequestItemList* und ein Array des Typs *ItemValue*, jeweils mit einem Element. Der gelesene Wert wird in einem Parameter des Typs *ReplyItemList* zurückgegeben. Außerdem sind je eine Instanz der Klasse *RequestOptions* sowie der Klasse *OPCError* als Rückgabeparameter vorzusehen.

```
/*-----
 | Name: Button_Write_Value_Click
 | Desc: Handler is being called, when button "Write Value"
 |       has been pressed
 | Notes: initiates an 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;
```

Die Methode *Write* der XML-Schnittstelle wird mit den zuvor definierten Parametern aufgerufen. Wenn der Wert geschrieben werden konnte, zeigt das Dialogfeld diesen im Textfeld "Value" an. Andernfalls wird "0" angezeigt.

```
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";
}
```

Auch die zurückgegebenen Werte für den Zeitstempel und die Qualität der Werte erscheinen im Dialogfeld in den entsprechenden Textfeldern.

```
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,
                        MessageBoxIcon.Exclamation);
    }
}
}

```

Programmfehler werden in einem Catch-Block behandelt.

```

catch(Exception excep)
{
    MessageBox.Show (this, excep.Message, this.Text,
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
}
}

```

## 6.8 OPC Alarms & Events Custom-Schnittstelle in C++

### Einleitung

Dieses Kapitel stellt die grundlegenden Schritte dar, die ein Client-Programm ausführen muss, um Events verarbeiten zu können.

### Schritt 1: Initialisierung der COM-Bibliothek

Jedes Programm, das die COM-Bibliothek nutzen möchte, muss diese erst initialisieren. Für diese Aufgabe gibt es die Funktion *CoInitializeEx*, die gegenüber *CoInitialize* über einen zusätzlichen Parameter für das gewünschte Thread-Modell verfügt. Damit kann festgelegt werden, ob das COM-Objekt im Single-Thread-Modus oder im Multi-Thread-Modus angelegt wird.

```
HRESULT hr = CoInitializeEx(NULL, COINIT_MULTITHREADED);
```

## Schritt 2: Konvertieren der ProgID in eine CLSID

Jeder COM-Server besitzt zur Identifizierung eine sogenannte *ProgID*, die eine weltweit eindeutige *CLSID* zugeordnet ist. Diese wird mit der Funktion *CLSIDFromProgID()* ermittelt. Die ProgID des OPC Alarms & Events-Servers von SIMATIC NET ist L"OPC.SimaticNETAlarms":

```
hr = CLSIDFromProgID(L"OPC.SimaticNETAlarms",
                      &clsidOPCEventServer);
```

## Schritt 3: Erzeugen eines Server-Objekts

Die Funktion *CoCreateInstance()* erzeugt eine Instanz der Klasse, deren *CLSID* vorgegeben wurde:

```
hr = CoCreateInstance (clsidOPCEventServer,
                      NULL,
                      CLSCTX_LOCAL_SERVER,
                      IID_IOPCEventServer,
                      (void**)&gpIOPCEventServer);
```

Ergebnis dieses Programmschritts ein Objekt der Klasse OPC-Server. Außerdem liefert *CoCreateInstance* einen Zeiger auf das Interface IOPCServer des Serverobjekts (Parameter *gpIOPCEventServer*).

## Schritt 4: Registrierung beim Event-Server

Der Client muss beim Server registriert sein, um Benachrichtigungen über Ereignisse zu erhalten. Die Methode *CreateEventSubscription* der Klasse OPCEventServer fügt dem Server ein Objekt der Klasse *OPCEventSubscription* hinzu und liefert einen Zeiger auf die Schnittstelle *IOPCEventSubscriptionMgt* (Parameter *pgIOPCEventSubscrMgt*).

```
hr = gpIOPCEventServer->CreateEventSubscription
    (TRUE,
     dwBufferTime,
     dwMaxSize,
     hClientSubscription,
     IID_IOPCEventSubscriptionMgt,
     (LPUNKNOWN*) &pgIOPCEventSubscrMgt,
     &dwRevisedBufferTime,
     &dwRevisedMaxSize );
```

## Schritt 5: Callback-Objekt erzeugen

Damit der Event-Server bei Ereignissen eine Benachrichtigung an den Client senden kann, muss im Client die Schnittstelle *OPCEventCallback* implementiert sein. Im Beispielprogramm gibt es dafür ein Objekt der Klasse *COPCEventCallback*, die von *CObjectRoot* und *IOPCEventSink* abgeleitet wurde.

```
CComObject<COPCEventCallback>::CreateInstance
    (&pgCOPCEventCallback);
```

Der Parameter *pgCOPCEventCallback* ist ein Zeiger auf das Callback-Objekt.

### Schritt 6: OPC-Event-Server und Callback-Objekt verbinden

Die Methode *AtAdvise()* erzeugt eine Verbindung zwischen dem OPC-Server und dem Callback-Objekt. Der erste Parameter ist ein Zeiger auf die IUnknown-Schnittstelle des Objekts, mit dem sich der Client verbinden möchte. Der zweite Parameter ist ein Zeiger auf die IUnknown-Schnittstelle des Callback-Objekts.

```
hr = AtlAdvise(pgIOPCEventSubscrMgt,
                pgCOPCEventCallback->GetUnknown(),
                IID_IOPCEventSink,
                &dwAdviseEvent);
```

### Schritt 7: Benachrichtigungen des Alarms & Events-Server empfangen

Beim Auftreten eines Alarms ruft der Server die Methode *OnEvent* des Callback-Objekts auf. Im Beispielprogramm ist *OnEvent* so implementiert, dass die Methode *displayEvent* der Dialogfeldklasse *CReceiveAnAlarm* Detailinformationen über das Event anzeigt.

```
STDMETHODIMP COPCEventCallback::OnEvent(
    OPCHANDLE hClientSubscription,
    BOOL bRefresh,
    BOOL bLastRefresh,
    DWORD dwCount,
    ONEVENTSTRUCT __RPC_FAR *pEvents)
```

Der Parameter *pEvents* ist ein Zeiger auf eine Struktur, in der die Informationen über die Events abgelegt sind.

### Schritt 8: Ein empfangenes Ereignis quittieren

Der Client quittiert die Benachrichtigung des Servers mit der Methode *AckCondition*. Der Parameter *pszSource* enthält die Ereignisquelle und *pszConditionName* die Bedingung, deren Statusänderung bestätigt werden soll. Beide Parameter sind Arrayadressen. Im Beispiel wird nur ein Ereignis quittiert. Ein Aufruf von *AckCondition* kann aber auch mehrere Ereignisse quittieren.

Die Werte für *pszSource* und *pszConditionName* liefert der Alarms & Events-Server bei seinem Aufruf von *OnEvent* als Bestandteile *szSource* und *szConditionName* der Struktur *ONEVENTSTRUCT*.

```
hr = gpIOPCEventServer->AckCondition(1,
                                         L"Me",
                                         L"NoComment",
                                         &pszSource,
                                         &pszConditionName,
                                         &ftActiveTime,
                                         &dwCookie,
                                         &pErrors);
```

### Schritt 9: Objekte löschen und Speicher freigeben

Vor dem Beenden des Programms müssen die erzeugten OPC-Objekte gelöscht und der angeforderte Speicher freigegeben werden. *AtlUnadvise* beendet die Verbindung zwischen dem OPC-Server und dem Callback-Objekt.

```
hr = AtlUnadvise(pgiOPCEventSubscrMgt,  
                  IID_IOPCEventSink,  
                  dwAdviseEvent);
```

Jede COM-Schnittstelle verfügt über die Methode *Release*, mit der der Referenzzähler der Schnittstelle dekrementiert wird. Wenn die Schnittstelle nicht mehr referenziert wird, werden die belegten Ressourcen freigegeben.

```
hr=pgIOPCEventSubscrMgt->Release();
```

Die Methode *CoUninitialize* schließt die COM-Bibliothek für den betreffenden Thread und gibt die von ihm belegten Ressourcen frei. Für jeden erfolgreichen Aufruf von *CoInitializeEx* muss es einen korrespondierenden Aufruf von *CoUninitialize* geben.

```
CoUninitialize();
```

## 6.9

## OPC-UA-Schnittstelle in C

Die vorliegenden Beispiele nutzen die OPC-UA-Schnittstelle in einfachem C.

Sie finden diese unter:

- "<Installationspfad>\SIEMENS\SIMATIC.NET\opc2\samples\ua\c\read.c\"
- "<Installationspfad>\SIEMENS\SIMATIC.NET\opc2\samples\ua\c\alarm.c\"
- "<Installationspfad>\SIEMENS\SIMATIC.NET\opc2\samples\ua\c\publish.c\"

Das Programm "Read.exe" zeigt die Funktion "synchrone Lesen". Es wird nur unverschlüsselte Kommunikation verwendet.

Das Programm "Alarm.exe". zeigt die Funktion "asynchrone Beobachten von Daten-Variablen und S7-Alarmen". Es wird verschlüsselte Kommunikation verwendet.

Das Programm "Publish.exe". zeigt die Funktionen "Lesen, Schreiben und asynchrones Beobachten von Daten-Variablen und S7-Alarmen". Es kann unverschlüsselte und verschlüsselte Kommunikation verwendet werden. Auch Benutzer-Authentifizierung ist möglich.

Die weitere Beschreibung erfolgt für das weiterreichende Beispiel "Publish", das auch die Funktionalität der einfacheren Beispiele "Read" und "Alarm" umfasst.

## 6.9.1

### Aktivieren der Simulationsverbindung

Damit das vorliegende Programm funktionsfähig ist, müssen Sie eine Simulationsverbindung aktivieren, welche die im Programm verwendete Demo-Variable verfügbar macht. Gehen Sie folgendermaßen vor:

1. Starten Sie das Programm "Kommunikations-Einstellungen" über das Startmenü.  
"Start" > "Siemens Automation" > "SIMATIC" > "SIMATIC NET" > "Kommunikations-Einstellungen"
2. Öffnen Sie im linken Navigationsfenster die Eigenschaftsseite für die OPC-Protokollauswahl, indem Sie die Ebenen "SIMATIC NET Konfiguration" > "OPC Protokollauswahl" öffnen.
3. Aktivieren Sie das Kontrollkästchen für das zu simulierende Protokoll.  
Das vorliegende Beispiel verwendet das S7-Protokoll.  
Aktivieren Sie deshalb beim S7-Protokoll die DEMO-Verbindungen.
4. Beenden Sie das Programm "Kommunikations-Einstellungen".

---

#### Hinweis

Damit die Änderungen wirksam werden, müssen zuvor alle OPC-Clients beendet und der OPC-Server neu gestartet werden!

---

## 6.9.2

### Importieren des Client-Zertifikates

Damit eine sichere Verbindung zum Server aufgebaut werden kann, muss der Server beim Zertifikataustausch das ihm übergebene Client-Zertifikat akzeptieren. Um dies zu gewährleisten, wird das Zertifikat in den Zertifikatspeicher des Servers importiert.

Gehen Sie folgendermaßen vor, um ein Client-Zertifikat zu importieren:

1. Starten Sie die Konfigurationskonsole "Kommunikations-Einstellungen" über das Startmenü.  
"Start" > "SIMATIC" > "SIMATIC NET" > "Einstellungen" > "Kommunikations-Einstellungen"
2. Öffnen Sie im linken Navigationsfenster die Eigenschaftsseite für die Zertifikate, indem Sie die Ebenen "SIMATIC NET Konfiguration" > "Applikationen" > "OPC-Einstellungen" öffnen und auf das Symbol "OPC-UA-Zertifikate" klicken.
3. Wählen Sie im Kontextmenü den Menübefehl "Client-Zertifikat importieren ..." aus.  
Dieses befindet sich auf Ihrer Festplatte unter  
<Installationspfad>\SIEMENS\SIMATIC.NET\opc2\samples\ua\c\pki\ca\certs\Opc.Publish.C.Sample.der und klicken Sie anschließend auf "Öffnen".
4. Beenden Sie das Programm "Kommunikations-Einstellungen".

### 6.9.3

### Bedienung des Beispielprogramms

Das Programm enthält zwei Menüs. Das erste ist das Hauptmenü (mainmenu), welches beim Starten des Programms erscheint. Ist eine Verbindung zum Server aufgebaut erscheint das Verbindungs menü (connectionmenu). Durch die Menüs werden folgende Aktionen ausgelöst:

Menüpunkt	Aktion
<b>mainmenu</b>	
c (connect)	Eine ungesicherte Verbindung zum Server aufbauen.
s (secure connect)	Eine gesicherte Verbindung zum Server aufbauen.
q (quit)	Das Programm beenden.
<b>connectionmenu</b>	
r (read)	Den Wert einer Variablen lesen.
w (write)	Einen Wert in eine Variable schreiben.
m (monitor)	Eine Variable auf Wertänderungen und Alarne beobachten.
d (disconnect)	Die Verbindung zum Server abbauen.
q (quit)	Die Verbindung zum Server abbauen und das Programm beenden.

### 6.9.4

### Programm starten

Das Programm befindet sich auf Ihrer Festplatte unter:

"<Installationspfad>\SIEMENS\SIMATIC.NET\opc2\samples\ua\c\publish.c"

Beim Programmstart erscheint das Hauptmenü (mainmenu). Durch Drücken der Taste "c" auf der Tastatur wird eine ungesicherte Verbindung zum Server aufgebaut. Wird stattdessen "s" gedrückt findet ein Zertifikataustausch zwischen Client und Server statt, bei dem beide Seiten das jeweils andere Zertifikat bestätigen müssen. Danach ist eine sichere Verbindung zum Server aufgebaut.

Nach dem Verbindungs aufbau wird das Verbindungs menü (connectionmenu) angezeigt, welches die Optionen zum Lesen, Schreiben und Beobachten von Variablen bereitstellt.

### 6.9.5

### Werte lesen und schreiben

Nach dem Auswählen der Aktion "read" durch Drücken der Taste "r" auf der Tastatur wird der Wert der Variablen gelesen und auf der Konsole ausgegeben. Anschließend erscheint das Verbindungs menü (connectionmenu).

Durch Drücken der Taste "w" auf der Tastatur und anschließendem Eingeben eines Integer-Wertes zwischen 0 und 255 wird dieser in die Variable geschrieben. Auch hier erscheint nach Beendigung der Aktion das Verbindungs menü (connectionmenu).

Im Ausgangszustand ist das Beispielprogramm für den Betrieb mit einer Demo-Verbindung ausgelegt. Soll das Beispielprogramm in einer realen Umgebung laufen, müssen Sie den Code an die realen Variablen anpassen (siehe Kapitel "Hinweise zum Umstellen auf reale Variablen (Seite 742)").

## 6.9.6

### Variablen beobachten

Durch das Drücken der Taste "m" auf der Tastatur werden Wertänderungen und Alarme einer Variablen kontinuierlich auf der Konsole ausgegeben. Mit der Taste "q" wird das Beobachten beendet und das Verbindungs menü aufgerufen.

Im Ausgangszustand ist das Beispielprogramm für den Betrieb mit einer Demo-Verbindung ausgelegt. Um Werteänderungen auf der Konsole zu sehen, müssen diese daher manuell oder über einen Generiermodus erzeugt werden.

Gehen Sie folgendermaßen vor:

1. Starten Sie das Programm OPC Scout V10 über das Startmenü.  
"Start" > "Siemens Automation" > "SIMATIC" > "SIMATIC NET" > "OPC Scout V10"
2. Öffnen Sie im linken Navigationsfenster die Ebenen.  
"UA-Server" > "opc.tcp://<hostname>:55101" > "Objects" > "Server" >  
"S7:" > "DEMO" > "blocks" > "db" > "db20"
3. Ziehen Sie die Variable "db20.0,b" per Drag and Drop in die DA-Ansicht 1.
4. Tragen Sie den neuen Wert in das Feld "Neuer Wert" ein, um den Wert manuell zu ändern und klicken anschließend auf die Schaltfläche "Schreiben". Durch Klicken der Schaltfläche "Lesen oder Beobachten EIN" erscheint im Feld "Wert" der aktuelle Wert der Variablen.
5. Tragen Sie im Feld "Generiermodus" die gewünschte Generierung ein, um das Generieren von Werten für die Variable einzustellen zu können, zum Beispiel [0...255]+1 für ein kontinuierliches Hochzählen der Werte von 0 bis 255.
6. Aktivieren Sie das Generieren des Wertes mit der Schaltfläche "Werte generieren EIN".

---

#### Hinweis

Soll das Beispielprogramm in einer realen Umgebung laufen, müssen Sie den Code an die realen Variablen anpassen (siehe Kapitel "Hinweise zum Umstellen auf reale Variablen (Seite 742)").

---

## 6.9.7

### Programm beenden

Das Beispielprogramm wird durch Drücken der Taste "q" im Haupt- oder Verbindungs menü beendet. Befindet sich das Programm im Verbindungs menü wird zuerst die Verbindung zum Server abgebaut.

## **6.9.8 Beschreibung des Programmablaufs**

### **6.9.8.1 Verbindungsauftbau**

Eine Verbindung zum Server kann über zwei Möglichkeiten aufgebaut werden. Zum Einen über einen sicheren Verbindungsauftbau mit Zertifikataustausch und sicherem Endpunkt und zum Anderen über einen nicht sicheren Endpunkt.

#### **Die Schritte des sicheren Verbindungsauftbaus**

1. Erstellen Sie einen Kanal.  
Um eine Verbindung zum Server aufzubauen, muss zuerst ein Kanal (engl. channel) erstellt werden. Hierfür wird die Funktion "OpcUa\_Channel\_Create()" verwendet.
2. Öffnen Sie einen Kanal zu einem Endpunkt mit Sicherheitsmodus "None".  
Um eine Verbindung zu einem Endpunkt herzustellen, wird die Funktion "OpcUa\_Channel\_Connect()" verwendet. Dieser wird die Sicherheitsanforderung des Endpunktes übergeben, in diesem Fall "None". Diese gibt an, dass keine Sicherheitsfunktionen vom Server gefordert werden. Damit ist diese Verbindung unsicher.
3. Lesen Sie die Endpunkte des Servers aus.  
Über die Funktion "OpcUa\_ClientApi\_GetEndpoints()" werden dem Client die Endpunkte des Servers mitgeteilt. Außerdem wird ihm das Serverzertifikat übergeben, das für den Verbindungsauftbau mit einem Endpunkt mit Sicherheitsmodus "Sign&Encrypt" benötigt wird.
4. Akzeptieren Sie das Serverzertifikat.  
Das Serverzertifikat muss vom Client akzeptiert werden. Danach wird mit der Funktion "ValidateCertificate()" geprüft, ob sich das Zertifikat bereits im Zertifikatspeicher des Client befindet. Ist dies nicht der Fall, wird es mit "SaveCertificate()" in den Speicher des Client kopiert.
5. Schließen Sie den unsicheren Kanal.  
Um eine Verbindung zu einem Endpunkt zu schließen, wird die Funktion "OpcUa\_ClientApi\_Disconnect()" verwendet.
6. Öffnen Sie einen gesicherten Kanal.  
Wie in Schritt 2 wird auch hier die Funktion "OpcUa\_Channel\_Connect()" verwendet, um eine Verbindung zu einem Endpunkt aufzubauen. Der hier verwendete Endpunkt hat den Sicherheitsmodus "Sign&Encrypt", was bedeutet, dass die Nachrichten zwischen Client und Server verschlüsselt und signiert werden müssen.
7. Erstellen Sie eine Session.  
Um eine Session zu erstellen, wird die Funktion "OpcUa\_ClientApi\_CreateSession()" verwendet. Die Rückgabe des Servers besteht aus einer eindeutigen Nummer, der sogenannten "ServerNonce", die für die Aktivierung der Session in Schritt 8 erforderlich ist.
8. Aktivieren Sie die Session.  
Um die in Schritt 7 erstellte Session zu aktivieren, wird die Funktion "OpcUa\_ClientApi\_ActivateSession()" verwendet. Dieser wird eine Signatur übergeben, die sich aus der in Schritt 7 erhaltenen "ServerNonce" und dem Serverzertifikat zusammensetzt.

## Die Schritte des ungesicherten Verbindungsaufbaus

1. Erstellen Sie einen Kanal.  
Um eine Verbindung zum Server aufzubauen, muss zuerst ein Kanal (engl. channel) erstellt werden. Hierfür wird die Funktion "OpcUa\_Channel\_Create()" verwendet.
2. Öffnen Sie einen Kanal zu einem Endpunkt mit Sicherheitsmodus "None"  
Um eine Verbindung zu einem Endpunkt herzustellen, wird die Funktion "OpcUa\_Channel\_Connect()" verwendet. Dieser wird die Sicherheitsanforderung des Endpunktes übergeben, in diesem Fall "None". Diese gibt an, dass keine Sicherheitsfunktionen vom Server gefordert werden. Damit ist diese Verbindung unsicher.
3. Erstellen Sie eine Session.  
Um eine Session zu erstellen, wird die Funktion "OpcUa\_ClientApi\_CreateSession()" verwendet. Die Rückgabe des Servers besteht aus einer eindeutigen Nummer, der sogenannten "ServerNonce", die für die Aktivierung der Session in Schritt 8 erforderlich ist.
4. Aktivieren Sie die Session.  
Um die in Schritt 7 erstellte Session zu aktivieren, wird die Funktion "OpcUa\_ClientApi\_ActivateSession()" verwendet. Dieser wird eine Signatur übergeben, die sich aus der in Schritt 7 erhaltenen "ServerNonce" und dem Serverzertifikat zusammensetzt.

### 6.9.8.2 Lesen und Schreiben von Variablen

#### Lesen des Wertes einer Variablen

Um den Wert einer Variablen zu lesen, wird die Funktion "OpcUa\_ClientApi\_Read()" verwendet. Diese gibt neben dem Wert auch einen Statuscode zurück, welcher angibt, ob die Funktion erfolgreich beendet werden konnte.

#### Einen Wert in eine Variable schreiben

Zum Schreiben eines Wertes in eine Variable wird die Funktion "OpcUa\_ClientApi\_Write()" verwendet. Ausgabe ist ein Statuscode, der angibt, ob die Funktion erfolgreich war.

### 6.9.8.3 Beobachten von Variablen und Alarmen

#### Gehen Sie so vor, um Variablen und Alarne beobachten zu können:

1. Erstellen Sie eine Subscription.  
Um eine Subscription zu erstellen, wird die Funktion "OpcUa\_CreateSubscription()" verwendet. Der Funktion wird als Übergabewert unter anderem das Publish-Intervall übergeben.
2. Erstellen Sie die MonitoredItems für Werte- und Alarmbeobachtung.  
Mit der Funktion "OpcUa\_ClientApi\_CreateMonitoredItems()" werden die MonitoredItems zum Beobachten von Datenänderungen und Alarmen erstellt. Für das MonitoredItem zum Beobachten von Alarmen müssen Sie einen Eventfilter setzen.

3. Senden Sie eine Publish-Anfrage an den Server.  
Um Alarne bzw. Datenänderungen vom Server zu bekommen, muss der Client die Funktion "OpcUa\_ClientApi\_BeginPublish()" aufrufen. Dieser wird die Callbackfunktion übergeben, mit dessen Aufruf der Server seine Antwort sendet.
4. Werten Sie die Antwort auf die Publish-Anfrage aus.  
Die Callbackfunktion "pfnUaServer\_ClientChannelRequestComplete()" empfängt die vom Server gesendete Antwort auf die Publish-Anfrage. Hier werden die dort enthaltenen Meldungen sortiert und die jeweilige Bildschirmausgabe erzeugt. Anschließend wird eine weitere Publish-Anfrage an den Server gesendet.
5. Löschen Sie die MonitoredItems  
Mit der Funktion "OpcUa\_ClientApi\_DeleteMonitoredItems()" werden die MonitoredItems gelöscht.
6. Löschen Sie die Subscription  
Die Funktion "OpcUa\_ClientApi\_DeleteSubscription()" löscht die Subscription. Dabei kann die Callbackfunktion mit einem Fehler aufgerufen werden, wenn eine ausstehende Publish-Antwort erst nach dem Löschen der Subscription erscheint.

#### **6.9.8.4 Verbindungsabbau**

**Gehen Sie so vor, um die Verbindung abzubauen:**

1. Schließen Sie die Session.  
Mit der Funktion "OpcUa\_ClientApi\_CloseSession()" wird die Session geschlossen.
2. Schließen Sie den Kanal.  
Die Funktion "OpcUa\_Channel\_Close()" schließt den Kanal.
3. Löschen Sie den Kanal.  
Der Kanal wird mit der Funktion "OpcUa\_Channel\_Delete()" gelöscht.

#### **6.9.9 Hinweise zum Umstellen auf reale Variablen**

Folgende Zeilen müssen an die Variablen angepasst werden, um das Beispielprogramm auf eine reale Variable umzustellen:

- g\_Nodeld\_Variable.NamespacelIndex = NAMESPACE\_S7;
- g\_Nodeld\_Variable.IdentifierType = OpcUa\_IdentifierType\_String;
- g\_Nodeld\_Variable.Identifier.String.strContent = NODE\_IDENTIFIER\_STRING;
- g\_Nodeld\_Variable.Identifier.String.uLength = OpcUa\_StrLenA(NODE\_IDENTIFIER\_STRING);

---

#### **Hinweis**

Weitere Information zu Nodeld und den Namensräumen eines Servers finden Sie im Kapitel "S7-Kommunikation mit OPC UA (Seite 161)".

---

## 6.10 OPC-UA-Schnittstelle (Asynchrone Kommunikation) in C#

Ausführlich beschriebene Beispiele zu Programmierung der OPC-UA-Schnittstelle in C# können Sie über unsere Internetseiten herunterladen:

Programmierung eines OPC UA .NET Client mit C# für den SIMATIC NET OPC-UA-Server (<http://support.automation.siemens.com/WW/view/de/42014088>)

Der OPC-UA-Client in der PC Station ist hier in zwei Komplexitätsstufen realisiert. Ein sehr einfach gestalteter Client (Simple OPC UA Client) zeigt Ihnen alle Basisfunktionen für den schnellen Einstieg in OPC UA. Ein komplexerer Client (OPC UA .NET Client) mit komfortabler Oberfläche demonstriert Ihnen den professionellen Umgang mit OPC UA, mit wieder verwendbaren Klassen realisiert unter .NET in der Programmiersprache C#.

Folgende Szenarien werden programmtechnisch in den beiden Beispiel-Clients erläutert:

- Anmelden, Abmelden und Authentifizierung am OPC-UA-Server
- Durchsuchen des Namensraums von Variablen
- Lesen, Schreiben und Beobachten von Variablen
- Lesen und Schreiben bei Nutzung von S7-Blockdiensten
- Verwendung von absoluter und symbolischer Adressierung
- einfaches Fehlerhandling



# Referenz Automation-Schnittstelle

## Einleitung

Dieses Kapitel enthält die Spezifikation der OPC-Automation-Schnittstelle.

Es beschreibt die Eigenschaften, die Methoden und - soweit vorhanden - die Ereignisse der einzelnen Objekte.

## 7.1 Allgemeine Informationen

Diese Einführung in die OPC-Automation-Schnittstelle erklärt die wichtigsten Bezeichnungen, die in der Spezifikation immer wieder genannt werden. Sie zu kennen ist wesentlich für das Verständnis der Automation-Schnittstelle.

Folgende Begriffe werden näher erläutert:

- Schnittstelle
- Schnittstellenarten
- COM/OLE-Objekte
- Collection-Objekte
- Objektmodell
- Datensynchronisation
- Ausnahmen
- Ereignisse
- Arrays
- Parameter
- TypeLibrary

Die vorliegende Beschreibung bezieht sich auf die Version 2.02 der Spezifikation für die OPC-Automation-Schnittstelle.

### 7.1.1 Was ist eine Schnittstelle?

Objekte können mehrere Schnittstellen haben und werden durch die Schnittstellen vollständig beschrieben. Nur über die Schnittstellen können die Methoden (Objektfunktionen) ausgeführt werden, und nur über die Schnittstellen kann auf Objektdaten zugegriffen werden. Objekte können von mehreren Anwendungen benutzt werden. Eine Schnittstelle besteht aus einer Tabelle von Zeigern auf die eigentlichen Funktionen.

## Struktur einer Schnittstelle

Ein Zeiger auf eine Schnittstelle ist in OLE 2.0 ein Zeiger auf eine Sprungtabelle mit Funktionszeigern. Der Aufrufer benutzt einen Zeiger auf die gewünschte Schnittstelle. Dahinter verbirgt sich wiederum ein Zeiger auf eine Liste von Funktionszeigern. Diese referenzieren nun die eigentlichen Methoden in COM-/OLE-Objekten.

### 7.1.2 Die zwei Schnittstellenarten von OPC

#### COM-/OLE-Schnittstellen

Eine COM-/OLE-Komponente stellt anderen Komponenten oder Anwendungen Objekte und deren Methoden zur Verfügung. Auf diese Objekte wird über die COM-/OLE-Schnittstellen zugegriffen. Eine Schnittstelle im Sinne von COM/OLE ist eine Gruppe von logisch zusammengehörigen Funktionen.

Der OPC-Server für SIMATIC NET unterstützt zwei verschiedene Arten von COM-/OLE-Schnittstellen, nämlich

- die Automation- und
- die Custom-Schnittstelle.

Beide Schnittstellen dienen der Kommunikation zwischen Objekten. Die Automation- und Custom-Schnittstelle unterscheiden sich darin, wie intern die Methoden einer Schnittstelle aufgerufen werden. Folglich gibt es auch zwei verschiedene Schnittstellenspezifikationen für den OPC-Server.

#### Wann wird welche Schnittstelle benutzt?

Client-Anwendungen, die auf einer Scriptsprache wie Visual Basic oder VBA basieren, müssen die Automation-Schnittstelle benutzen. Grundsätzlich gilt, daß Anwendungen in der Programmiersprache C/C++ für eine maximale Performance auf der Custom-Schnittstelle aufsetzen sollten. Die Anwendung der Automation-Schnittstelle ist jedoch auch in C/C++ möglich.

### 7.1.3 COM-/OLE-Objekte

COM-/OLE-Objekte sind Einheiten in Windows, die anderen Objekten über Ihre Schnittstellen eine definierte Funktionalität anbieten. COM-/OLE-Objekte bieten Ihre Dienste über fest definierte Schnittstellen an. Der Inhalt des Objekts, Daten und Code, bleiben dem Objektbenutzer verborgen. COM-/OLE-Objekte werden durch Ihre Schnittstellen definiert. Der Begriff des Objekts im Sinne von OLE entspricht nicht der Objektdefinition in objektorientierten Programmiersprachen. COM-/OLE-Objekte unterstützen beispielsweise nicht die Vererbung.

## Struktur eines COM-/OLE-Objekts

Der Zugriff auf das Objekt erfolgt nur über eine der Schnittstellen. Auf das eigentliche Objekt als Ganzes, die darin enthaltenen Daten oder den Code, gibt es keine Zugriffsmöglichkeit. Die Schnittstellen verbergen die ihnen zugeordneten Methoden.

### 7.1.4 Collection-Objekte

Ein Collection-Objekt dient der Erzeugung und Verwaltung von Untereinheiten. Erst wenn ein Collection-Objekt besteht, können die dazugehörigen Untereinheiten erzeugt werden. Die Collection-Objekte legen Defaultwerte für die jeweils dazugehörigen Untereinheiten fest.

COM-/OLE-Automation-Collections sind Objekte, die die Methode *Item*, die Eigenschaft *Count*, und die versteckte Eigenschaft *\_NewEnum* unterstützen. Jedes Objekt, das diese Eigenschaften als Teil der Schnittstelle besitzt, ist ein Collection-Objekt.

In VB gibt es zwei Methoden, die Elemente dieser Collections zu durchlaufen.

Die erste Methode gebraucht explizit *Count* und *Item*, um die Collection-Elemente zu identifizieren.

```
For I = 1 To object.Count
element = object.Item ( I )

' oder...

element = object( I )
Next I
```

Bei der zweiten Möglichkeit werden die verfügbaren Items durchlaufen, indem die verborgene Funktion *\_NewEnum* eingesetzt wird:

```
For Each element In object
...
Next element
```

Die Methode mit *For Each* ist zum Durchlaufen einer Collection schneller als die Verwendung der Eigenschaft *Item*.

Mit *Item* kann auch auf einen bestimmten Index wie etwa *Item( 3 )* zugegriffen werden. Die Verwendung dieser Eigenschaft ist also nicht auf Schleifen beschränkt.

## Zwei Arten von Collection-Objekten

Bei OPC gibt es zwei verschiedene Collection-Objekte: Mit dem Collection-Objekt OPCGroups können Objekte der Klasse OPCGroup angelegt und/oder bearbeitet werden. Das Collection-Objekt OPCGroups verfügt über sieben Eigenschaften, sieben Methoden und ein Ereignis.

Mit dem Collection-Objekt OPCItems können Objekte der Klasse OPCItem erzeugt und/oder bearbeitet werden. Das Collection-Objekt OPCItems verfügt über fünf Eigenschaften und neun Methoden, jedoch über keine Ereignisse.

### 7.1.5 Objektmodell

#### Objekte der Klasse OPCServer

Objekte der Klasse OPCServer werden durch den Client erzeugt. Die Eigenschaften eines OPC-Servers enthalten allgemeine Informationen über den Server. Mit der Erzeugung eines OPCServer-Objekts wird auch eine OPCGroup-Collection als Eigenschaft des OPCServer-Objektes angelegt.

#### Objekte der Klasse OPCGroups

Das Objekt OPCGroups ist ein Collection-Objekt zur Erzeugung und Verwaltung von OPCGroup-Objekten. Die Default-Eigenschaften von OPCGroups legen Defaultwerte für die Erzeugung aller OPCGroup-Objekte fest.

---

##### Hinweis

Public Groups werden vom OPC-Server für SIMATIC NET nicht unterstützt.

---

#### Objekte der Klasse OPCGroup

Die Klasse OPCGroup verwaltet die einzelnen Prozessvariablen, die OPC-Items. Mit Hilfe von OPCGroup-Objekten kann ein Client semantisch sinnvolle Einheiten von OPCItem-Objekten bilden und mit diesen Operationen ausführen.

#### Objekte der Klasse OPCItems

Das Objekt OPCItems ist ein Collection-Objekt zur Erzeugung und Verwaltung von Objekten der Klasse OPCItem. Die Default-Eigenschaften von OPCItems legen Defaultwerte für alle zu erzeugenden OPCItem-Objekte fest.

#### Objekte der Klasse OPCItem

Ein Objekt der Klasse OPCItem repräsentiert eine Verbindung zu einer Prozessvariable, zum Beispiel zum Eingabemodul einer speicherprogrammierbaren Steuerung. Eine Prozessvariable ist ein schreibbares und/oder lesbares Datum der Prozessperipherie, wie zum Beispiel die Temperatur eines Kessels. Mit jeder Prozessvariable ist ein Wert (Datentyp VARIANT), eine Qualität und ein Zeitstempel verbunden.

Die folgende Darstellung verdeutlicht die Hierarchie, die dem Objektmodell zu Grunde liegt:

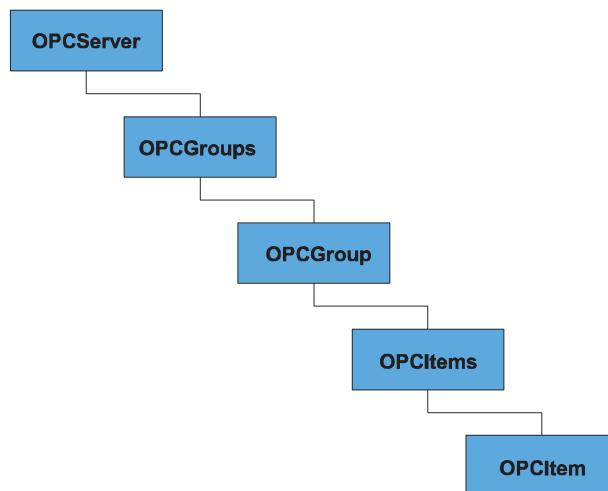


Bild 7-1 Dem Objektmodell zu Grunde liegende Hierarchie

### 7.1.6 Datensynchronisation

Der VB-Client muss in der Lage sein, Daten derart zu lesen und zu empfangen, dass Wert, Qualität und Zeitstempel synchron gehalten werden. Der Client muss sicher sein können, dass Datenqualität und Zeitstempel zu den Werten passen.

Erhält ein Client Werte mittels einer Lesemethode, sind Wert, Qualität und Zeitstempel synchron.

Erhält ein Client Daten mittels des Ereignisses *DataChange*, sind Wert, Zeitstempel und Qualität im Bereich der Ereignisbehandlungsroutine synchron.

Werden diese beiden Methoden der Datenbeschaffung nicht strikt getrennt, kann der Client nicht sicherstellen, dass die Item-Eigenschaften exakt synchron sind; dies kann beispielsweise dann der Fall sein, wenn sich die Eigenschaften der Daten verändern, während der Client auf diese Eigenschaften zugreift.

### 7.1.7 Ausnahmen

Die meisten Eigenschaften und Ereignisse, die hier beschrieben werden, kommunizieren mit einem OPC-Custom-Server. Bei der OLE-Automation ist es nicht einfach, einen Fehler zurückzugeben, wenn auf eine Eigenschaft zugegriffen wird. Im Falle eines Fehlers in der entsprechenden Datenquelle erzeugt der Automation-Server eine Ausnahme. Dies bedeutet, dass der Client eine Ausnahmebehandlung besitzen muss, um mit Fehlern umgehen zu können.

Fehler, die beim Schreiben einer Eigenschaft auftreten, werden mittels des *Err*-Objekts von Visual Basic angezeigt.

### 7.1.8 Ereignisse

Die Automation-Schnittstelle unterstützt den Benachrichtigungsmechanismus für Ereignisse von Visual Basic 5.0.

Der Automation-Server löst Ereignisse aus entsprechend den Anforderungen der Methodenaufrufe *AsyncRefresh*, *AsyncRead* und *AsyncWrite* und auch dann, wenn sich Daten entsprechend den Vorgaben des Clients ändern.

Die Implementierung setzt voraus, dass der Client über eine geeignete Ereignisbehandlung verfügt.

### 7.1.9 Arrays

Die Nummerierung von Arrays beginnt mit 1. Wird in einer Funktion ein Array verwendet, das länger ist als Count- oder NumItems-Parameter, wird nur eine dem Parameter entsprechende Anzahl von Array-Elementen verwendet (wobei der Index bei 1 beginnt).

Dies bezieht sich lediglich auf Parameter für Funktionen und Ereignisse innerhalb der Automation-Schnittstelle; es trifft nicht zu auf Item-Werte, bei denen der Datentyp selbst ein Array ist.

Um Fehler zu vermeiden sollte der VB-Code *Option Base 1* benutzt werden.

### 7.1.10 Parameter

#### Optionale Parameter

Optionale Parameter sind mit *Optional* gekennzeichnet. Optionale Parameter müssen nicht unbedingt in einen Methodenaufruf aufgenommen werden, wenn das Standard-Verhalten ausreichend ist. Bei OLE-Automation müssen die optionalen Parameter als *Variant* deklariert werden, auch wenn sie einen String oder ein Array etc. enthalten.

#### Methodenparameter

Methodenparameter werden als Werte weitergegeben, wenn sie nicht über den Zusatz *ByRef* als Referenz spezifiziert sind. *ByRef*-Parameter werden von Methoden mit Werten belegt und zurückgegeben.

### 7.1.11 Type Library

Bei VB wird die "OPC Automation Type Library" verwendet, um die folgenden Schnittstellen zu definieren. Stellen Sie sicher, dass "OPC Automation 2.0" aktiviert ist.

## 7.2 Das Objekt OPCServer

### Beschreibung

Ein Client generiert das OPCServer-Automation-Objekt und *verbindet* es dann mit der OPC Data Access-Custom-Schnittstelle (s. Methode *Connect*). Mit Hilfe des Objekts OPCServer können jetzt allgemeine Informationen zu einem OPC-Server abgerufen werden und ein Collection-Objekt OPCGroups erstellt und bearbeitet werden.

### Syntax

```
OPCServer
```

### Bemerkungen

Eine Deklaration mit *WithEvents* bewirkt, dass ein Objekt die für diesen Objekttyp spezifizierten Ereignisse unterstützt. Das Objekt OPC-Server besitzt nur ein einziges Ereignis, das Ereignis *ServerShutdown*. Das Objekt OPCGroup verfügt über alle Ereignisse, die mit *DataChange* und der Unterstützung asynchroner Methoden des Objekts OPCGroup zusammenhängen.

### Beispiel

```
Dim WithEvents AnOPCServer As OPCServer  
Set AnOPCServer = New OPCServer
```

### 7.2.1 Eigenschaften des Objekts OPCServer

StartTime  
CurrentTime  
LastUpdateTime  
MajorVersion  
MinorVersion  
BuildNumber  
VendorInfo  
ServerState  
LocaleID  
Bandwidth  
OPCGroups  
PublicGroupNames  
ServerName  
ServerNode  
ClientName

### 7.2.1.1 StartTime

#### Beschreibung

(Nur lesbar) Diese Eigenschaft gibt den Zeitpunkt an, an dem der Server, den der Client ausgewählt hat, gestartet wurde. Mehrere Clients, die mit demselben OPC-Server verbunden sind, lesen auch dieselbe Zeit für diese Eigenschaft ein.

#### Syntax

```
StartTime As Date
```

#### Bemerkungen

Der Automation-Server benutzt die Custom-Schnittstelle *GetStatus()*, um die Werte sowohl für diese Eigenschaft als auch für viele andere Eigenschaften des Objekts OPCServer zu erhalten. Es treten Fehler auf, wenn sich der Client nicht mit der Methode *Connect* mit dem OPC-Server verbunden hat.

#### Beispiel

```
Dim AnOPCServerTime As Date  
AnOPCServerTime = AnOPCServer.StartTime
```

### 7.2.1.2 CurrentTime

#### Beschreibung

(Nur lesbar) Diese Eigenschaft gibt die aktuelle Zeit vom Server zurück. Wenn Sie auf die Eigenschaft *CurrentTime* zugreifen, erhalten Sie den Wert, den der Automation-Server vom Custom-Server über das GetStatus-Interface bekommen hat.

#### Syntax

```
CurrentTime As Date
```

#### Bemerkungen

Es treten Fehler auf, wenn sich der Client nicht mit der Methode *Connect* mit dem OPC-Server verbunden hat.

#### Beispiel

```
Dim AnOPCServerTime As Date  
AnOPCServerTime = AnOPCServer.CurrentTime
```

### 7.2.1.3 LastUpdateTime

#### Beschreibung

(Nur lesbar) Diese Eigenschaft gibt den Zeitpunkt an, an dem der Server das letzte Daten-Update gesendet hat. Wenn Sie auf die Eigenschaft *LastUpdateTime* zugreifen, erhalten Sie den Wert, den der Automation-Server vom Custom-Server über das GetStatus-Interface bekommen hat.

#### Syntax

```
LastUpdateTime As Date
```

#### Bemerkungen

Diese Eigenschaft gibt den Zeitpunkt an, an dem der Server zum letzten Mal Daten an einen Anwendungs-Client gesendet hat. Es treten Fehler auf, wenn sich der Client nicht mit der Methode *Connect* mit dem OPCServer verbunden hat.

#### Beispiel

```
Dim AnOPCServerTime As Date
AnOPCServerTime = AnOPCServer.LastUpdateTime
```

### 7.2.1.4 MajorVersion

#### Beschreibung

(Nur lesbar) Diese Eigenschaft gibt die Hauptversionsnummer des Servers an (z. B. 1 bei der Version 1.32). Wenn Sie auf die Eigenschaft *MajorVersion* zugreifen, erhalten Sie den Wert, den der Automation-Server vom Custom-Server über das GetStatus-Interface bekommen hat.

#### Syntax

```
MajorVersion As Integer
```

#### Bemerkungen

Es treten Fehler auf, wenn sich der Client nicht mit der Methode *Connect* mit dem OPC-Server verbunden hat.

#### Beispiel

```
Dim AnOPCServerMajorVersion As String
AnOPCServerMajorVersion = Str(AnOPCServer.MajorVersion)
```

### 7.2.1.5 MinorVersion

#### Beschreibung

(Nur lesbar) Diese Eigenschaft gibt die untergeordnete Versionsnummer des Servers an (z. B. 32 bei der Version 1.32). Wenn Sie auf die Eigenschaft *MinorVersion* zugreifen, erhalten Sie den Wert, den der Automation-Server vom Custom-Server über das GetStatus-Interface bekommen hat.

#### Syntax

```
MinorVersion As Integer
```

#### Bemerkungen

Es treten Fehler auf, wenn sich der Client nicht mit der Methode *Connect* mit dem OPC-Server verbunden hat.

#### Beispiel

```
Dim AnOPCServerMinorVersion As String  
AnOPCServerMinorVersion = Str(AnOPCServer.MinorVersion)
```

### 7.2.1.6 BuildNumber

#### Beschreibung

(Nur lesbar) Diese Eigenschaft gibt die Build-Nummer des Servers an. Wenn Sie auf die Eigenschaft *BuildNumber* zugreifen, erhalten Sie den Wert, den der Automation-Server vom Custom-Server über das GetStatus-Interface bekommen hat.

#### Syntax

```
BuildNumber As Integer
```

#### Bemerkungen

Es treten Fehler auf, wenn sich der Client nicht mit der Methode *Connect* mit dem OPC-Server verbunden hat.

#### Beispiel

```
Dim BuildNumber as Integer  
BuildNumber = AnOPCServer.BuildNumber
```

### 7.2.1.7 VendorInfo

#### Beschreibung

(Nur lesbar) Diese Eigenschaft gibt eine Information über den Hersteller des Servers aus (als String). Wenn Sie auf die Eigenschaft *VendorInfo* zugreifen, erhalten Sie den Wert, den der Automation-Server vom Custom-Server über das GetStatus-Interface bekommen hat.

#### Syntax

```
VendorInfo As String
```

#### Bemerkungen

Es treten Fehler auf, wenn sich der Client nicht mit der Methode *Connect* mit dem OPC-Server verbunden hat.

#### Beispiel

```
Dim info As String
info = AnOPCServer.VendorInfo
```

### 7.2.1.8 ServerState

#### Beschreibung

(Nur lesbar) Diese Eigenschaft gibt den Zustand des Servers als Konstante des Typs OPCServerState an.

#### Syntax

```
ServerState As Long
```

#### Einstellungen

##### **OPC\_STATUS\_RUNNING**

Der Server läuft normal. Dies ist der Normalzustand des Servers.

##### **OPC\_STATUS\_FAILED**

Beim Server ist ein Hersteller spezifischer Fehler aufgetaucht, und er kann nicht korrekt arbeiten. Die Fehlerbeseitigung ist Hersteller spezifisch. Jede andere Methode des Servers gibt den Fehlercode E\_FAIL aus.

##### **OPC\_STATUS\_NOCONFIG**

Der Server läuft zwar, hat aber keine Informationen über die Konfigurierung und kann daher nicht korrekt arbeiten.

Hinweis: Der Server benötigt Informationen über die Konfigurierung, damit er arbeiten kann! Server, die keine Informationen zur Konfigurierung benötigen, zeigen diesen Status nicht an.

### OPC\_STATUS\_SUSPENDED

Der Server hat zeitweise keine Verbindung über eine Hersteller spezifische Methode, und er sendet und empfängt keine Daten. Angaben zur Qualität lauten:  
OPC\_QUALITY\_OUT\_OF\_SERVICE.

### OPC\_STATUS\_TEST

Der Server befindet sich im Test-Modus. Der Daten-Output ist von der Hardware abgekoppelt, der Server verhält sich ansonsten jedoch normal. Abhängig von der Hersteller spezifischen Implementierung sind die Eingangsdaten entweder tatsächlich vorhanden oder werden simuliert. Angaben zur Qualität werden normal ausgegeben.

## Bemerkungen

Die o. g. Statusangaben werden in der Spezifikation zur Data Access-Custom-Schnittstelle beschrieben und von einem OPC-Server über die Custom-Schnittstelle zurückgegeben; für weitere Informationen s. *IOPCServer::GetStatus()* in der Spezifikation *OPC Data Access Custom Interface*. Diese Angaben hat der Automation-Server vom Custom-Server übernommen, die über die Schnittstelle *GetStatus ()* miteinander verbunden sind. Es treten Fehler auf, wenn sich der Client nicht mit der Methode *Connect* mit dem OPC-Server verbunden hat.

## Beispiel

```
Dim ServerState As Long  
ServerState = AnOPCServer.ServerState
```

### 7.2.1.9 LocaleID

## Beschreibung

(Les- und schreibbar) Diese Eigenschaft identifiziert die Lokalisierung; mit *LocaleID* können Strings, die vom Server zurückgegeben werden, sprachspezifisch angepasst werden. Diese *LocaleID* wird von der Methode *GetErrorString* dieser Schnittstelle verwendet.

## Syntax

```
LocaleID As Long
```

## Bemerkungen

Die *LocaleID* sollte in allen Funktionen des Servers, die von der *LocaleID* betroffen sind, als Standard eingestellt sein.

Es treten Fehler auf, wenn sich der Client nicht mit der Methode *Connect* mit dem OPC-Server verbunden hat.

## Beispiel

```
' Ermitteln der Eigenschaft:  
Dim LocaleID As Long  
LocaleID = AnOPCServer.LocaleID  
  
' Festlegen der Eigenschaft:  
AnOPCServer.LocaleID = LocaleID
```

### 7.2.1.10 Bandwidth

#### Beschreibung

(Nur lesbar) *Bandwidth* gibt die Bandbreite des Servers an. Diese Eigenschaft ist serverspezifisch. Es wird empfohlen, die Bandbreite des Servers prozentual zur verfügbaren Bandbreite anzugeben. Der Wert beträgt hFFFFFFF, wenn der Server keine Bandbreite errechnen kann. Wenn Sie auf diese Eigenschaft zugreifen, erhalten Sie den Wert, den der Automation-Server vom Custom-Server über das GetStatus-Interface bekommen hat.

#### Syntax

```
Bandwidth As Long
```

#### Bemerkungen

Es treten Fehler auf, wenn sich der Client nicht mit der Methode *Connect* mit dem OPC-Server verbunden hat.

## Beispiel

```
Dim Bandwidth As Long  
Bandwidth = AnOPCServer.Bandwidth
```

### 7.2.1.11 OPCGroups

#### Beschreibung

(Nur lesbar) *OPCGroups* ist eine Collection von Objekten der Klasse *OPCGroup*. Dies ist eine Standard-Eigenschaft des OPCServer-Objekts.

#### Syntax

```
OPCGroups As OPCGroups
```

## Beispiel

```
' genaue Spezifizierung der Eigenschaft:  
Dim groups As OPCGroups  
Set groups = AnOPCServer.OPCGroups  
  
' Default-Spezifikation:  
Dim groups As OPCGroups  
Set groups = AnOPCServer
```

### 7.2.1.12 PublicGroupNames

#### Beschreibung

(Nur lesbar) Diese Eigenschaft gibt die Namen der PublicGroups an, die der Server anbietet. Diese Namen können in ConnectPublicGroup verwendet werden. Die Namen werden als ein Array von Strings ausgegeben.

#### Syntax

```
PublicGroupNames As Variant
```

#### Bemerkungen

Es treten Fehler auf, wenn sich der Client nicht mit der Methode *Connect* mit dem OPC-Server verbunden hat. Unterstützt der verwendete OPC-Server keine Public Groups oder sind keine Public Groups definiert, wird eine leere Liste ausgegeben.

## Beispiel

```
Dim AllPublicGroupNames As Variant  
AllPublicGroupNames = AnOPCServer.PublicGroupNames
```

### 7.2.1.13 ServerName

#### Beschreibung

(Nur lesbar) Diese Eigenschaft gibt den Namen des Servers aus, mit dem der Client über die Methode *Connect* verbunden ist.

#### Syntax

```
ServerName As String
```

## Bemerkungen

Der Automation-Server liefert den Wert, den er lokal im Cache-Speicher findet. Wird kein Wert ausgegeben, ist der Client nicht mit einem Data-Access-Server verbunden.

## Beispiel

```
Dim info As String
info = AnOPCServer.ServerName
```

### 7.2.1.14 ServerNode

#### Beschreibung

(Nur lesbar) Diese Eigenschaft gibt den Rechnernamen des Knotens im Netzwerk aus, auf dem der OPC-Server ausgeführt wird. Wenn Sie auf diese Eigenschaft zugreifen, erhalten Sie den Wert, den der Automation-Server lokal im Cache-Speicher findet.

#### Syntax

```
ServerNode As String
```

#### Bemerkungen

Wenn der Client nicht mit einem Data-Access-Server verbunden ist, wird kein Wert ausgegeben. Wurde in der Methode *Connect* kein Host-Name angegeben, ist der String ebenfalls leer.

## Beispiel

```
Dim info As String
info = AnOPCServer.ServerName
```

### 7.2.1.15 ClientName

#### Beschreibung

(Les- und schreibbar) Hier kann der Client optional einen ClientName beim Server registrieren lassen. Dieses Vorgehen wird hauptsächlich bei der Fehlersuche angewendet. Es wird empfohlen, daß der Client seinen Teilnehmer- und EXE-Namen hier festlegt.

#### Syntax

```
ClientName As String
```

## Bemerkungen

Es wird empfohlen, sowohl Teilnehmer- als auch Client-Namen durch Strichpunkt (;) getrennt in den String einzutragen.

## Beispiel

```
' Ermitteln der Eigenschaft:  
Dim info As String  
info = AnOPCServer.ClientName  
  
' Festlegen der Eigenschaft:  
AnOPCServer.ClientName = "NodeName;c:\programfiles\vendor\someapplication.exe"
```

## 7.2.2 Methoden des Objekts OPCServer

GetOPCServers  
Connect  
Disconnect  
CreateBrowser  
GetErrorString  
QueryAvailableLocaleIDs  
QueryAvailableProperties  
GetItemProperties  
LookupItemIDs

### 7.2.2.1 GetOPCServers

#### Beschreibung

Diese Eigenschaft gibt die Namen (ProgIDs) der registrierten OPC-Server aus. Verwenden Sie eine dieser ProgIDs in der Methode *Connect*. Die Namen werden als ein Array von Strings ausgegeben.

#### Syntax

```
GetOPCServers (Optional Node As Variant) As Variant Node
```

##### Node

Mit dem Knotenname kann festgelegt werden, für welche Netzwerknoten der Automation-Server alle registrierten OPC-Server ausgeben soll.

## Bemerkungen

Zur Registrierung für Custom-Server siehe Standard der OPC Data Access-Custom-Schnittstelle.

Der Gebrauch eines Knotens ist optional. Wird ein Knotenname verwendet, findet der Zugriff auf einen anderen Rechner mittels DCOM statt. Erlaubte Knotennamen sind UNC-Namen (Server) und DNS-Namen (server.com, www.vendor.com oder 180.151.19.75).

## Beispiel

```
' Ermitteln der registrierten OPC-Server (die tatsächlich
' vorhandenen OPC-Server, die zu einer VB-Listbox hinzugefügt
' werden):
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

#### Beschreibung

Soll eine Verbindung zu einem OPC-Data-Access-Server (der das Custom-Interface zur Verfügung stellt) aufgebaut werden, muss diese Methode aufgerufen werden.

#### Syntax

```
Connect (ProgID As String,
         Optional Node As Variant)
```

##### ProgID:

*ProgID* ist ein String, der den registrierten OPC-Data-Access-Server (der das Custom-Interface zur Verfügung stellt) eindeutig identifiziert.

##### Node:

Mit dem Knotennamen kann festgelegt werden, dass der Verbindungsauflauf zu einem anderen Rechner mittels DCOM stattfindet.

## Bemerkungen

Jede Instanz eines OPC-Automation-Server ist mit einem OPC-Data-Access-Server (der das Custom-Interface zur Verfügung stellt) verbunden.

Der Gebrauch eines Knotens ist optional. Wird ein Knotenname verwendet, findet der Zugriff auf einen anderen Rechner mittels DCOM statt. Erlaubte Knotennamen sind UNC-Namen (Server) und DNS-Namen (server.com, www.vendor.com oder 180.151.19.75).

Bei dieser Methode ruft der Automation Wrapper *CoCreateInstanceEx* auf und baut damit im spezifizierten Rechner (StrNodeName) einen Data-Access-Custom-Server (der durch die ProgID spezifiziert ist) auf.

Wird diese Methode ein weiteres Mal aufgerufen, ohne daß vorher die Verbindung durch einen Aufruf der Methode *Disconnect* beendet wurde, hebt der Automation Wrapper automatisch die vorherige Verbindung auf.

Verwenden Sie die Methode *GetOPCServers*, um gültige ProgIDs zu ermitteln.

## Beispiel

```
' Herstellen einer Verbindung zum ersten registrierten
' OPCServer, der über die Methode "GetOPCServers" ermittelt
' wurde:
Dim AllOPCServers As Variant
AllOPCServers = AnOPCServer.GetOPCServers
AnOPCServer.Connect(AllOPCServers(1))

' Verbindung zu einem bestimmten Server oder einem
' Netzwerkrechner:
Dim ARealOPCServer As String
Dim ARealOPCNodeName As String
ARealOPCServer = "Herstellername.DataAccessCustomServer"
ARealOPCNodeName = "EinComputername"
AnOPCServer.Connect (ARealOPCServer, ARealOPCNodeName)
```

### 7.2.2.3 Disconnect

#### Beschreibung

Diese Methode baut die Verknüpfung zu einem OPC-Server ab.

#### Syntax

```
Disconnect()
```

#### Bemerkungen

Mit dieser Methode läßt sich die Verbindung zu einem Server abbauen; anschließend kann dann entweder eine neue Verbindung zu einem anderen Server aufgebaut oder das Server-Objekt gelöscht werden.

Es ist guter Programmierstil, wenn die Client-Anwendung mit den entsprechenden Automation Methoden alle Objekte abbaut, die von ihr erzeugt wurden (einschließlich aller OPCGroup(s) und OPCItem(s)-Objekte). Die Methode *Disconnect* entfernt alle Gruppen-Objekte und Referenzierungen zum jeweiligen OPC-Custom-Server.

## Beispiel

```
AnOPCServer.Disconnect
```

### 7.2.2.4 CreateBrowser

#### Beschreibung

Diese Methode erzeugt ein Objekt der Klasse OPCBrowser.

#### Syntax

```
CreateBrowser() As OPCBrowser
```

#### Bemerkungen

Die OPC-Browser-Schnittstelle ist optional und muss nicht zwangsläufig von einem OPC-Custom-Interface-Server unterstützt werden. Dies bedeutet, dass ein OPC-Custom-Interface-Server, der die Browser-Schnittstelle nicht implementiert, auch kein Objekt der Klasse OPC-Browser zurückgibt.

#### Beispiel

```
Dim ARealOPCServer As String
Dim ARealOPCNodeName As String
ARealOPCServer = "Herstellername.DataAccessCustomServer"
ARealOPCNodeName = "EinComputername"
AnOPCServer.Connect(ARealOPCServer, ARealOPCNodeName)
Dim AnOPCServerBrowserObject As OPCBrowser
Set AnOPCServerBrowserObject = AnOPCServer.CreateBrowser
```

### 7.2.2.5 GetErrorString

#### Beschreibung

Diese Methode konvertiert eine Fehlermeldung in einen lesbaren String. Der Server gibt den String in der Lokalisierung aus, die in der Eigenschaft LocaleID des Servers festgelegt ist.

#### Syntax

```
GetErrorString (ErrorCode As Long) As String
```

#### ErrorCode

*ErrorCode* ist ein serverspezifischer Fehlercode, den die Client-Anwendung von einer Interface-Funktion des Servers zurückerhalten hat. Für diesen Fehlercode fragt die Client-Anwendung die Textdarstellung nach.

## Beispiel

```
Dim AnOPCServerErrorString As String
' dieses Beispiel setzt voraus, dass während des Hinzufügens
' mehrerer Items ein Item als ungültig erkannt wird; der Code
' wird hier zur Vereinfachung nicht vollständig angegeben
AnOPCItemCollection.Add AddItemCount,
AnOPCItemIDs,
AnOPCItemServerHandles,
AnOPCItemErrors
' Ermitteln und Anzeigen der Fehlermeldung, um dem Benutzer
' mitzuteilen, weshalb das Item nicht hinzugefügt werden
' konnte
AnOPCServerErrorString =
    AnOPCServer.GetErrorString(AnOPCItemErrors (index))
' weiterer Code
ErrorBox.Text = AnOPCServerErrorString
' weiterer Code
```

### 7.2.2.6 QueryAvailableLocaleIDs

#### Beschreibung

Diese Methode gibt die verfügbaren LocaleIDs für die bestehende Server-Client-Verbindung als einen Array von Long-Werten aus.

#### Syntax

```
QueryAvailableLocaleIDs () As Variant
```

## Beispiel

```
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

#### Beschreibung

Diese Methode gibt eine Liste der Beschreibungen und ID-Codes der verfügbaren Eigenschaften der jeweiligen ItemID aus. Diese Liste kann je nach ItemID unterschiedlich sein. Sie sollte relativ stabil für eine bestimmte ItemID sein. Dies bedeutet, dass sie u. U. von Änderungen der Systemkonfiguration betroffen ist.

## Syntax

```
QueryAvailableProperties (ItemID As String,
                         ByRef Count As Long,
                         ByRef PropertyIDs () as Long,
                         ByRef Descriptions() As String,
                         ByRef DataTypes() As Integer)
```

### **ItemID**

*ItemID* ist die ItemID, zu der die verfügbaren Eigenschaften gesucht werden sollen.

### **Count**

*Count* gibt die Anzahl der gefundenen Eigenschaften an.

### **PropertyIDs**

*PropertyIDs* sind IDs des Typs DWORD für die gefundenen Eigenschaften. Diese IDs können an *GetItemProperites* oder *LookupItemIDs* weitergegeben werden.

### **Descriptions**

*Descriptions* ist eine Kurzbeschreibung des Herstellers zu jeder Eigenschaft.

Hinweis: Die LocaleID hat keinen Einfluss auf die Sprache dieser Beschreibung.

### **DataTypes**

*DataTypes* ist der Datentyp, den *GetItemProperties* für diese Eigenschaft ausgibt.

## Beispiel

```
' Ermitteln der verfügbaren Eigenschaften:
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

#### Beschreibung

Diese Methode gibt eine Liste der aktuellen Werte der Eigenschaften für die eingegebene ItemID aus.

## Syntax

```
GetItemProperties ( ItemID As String,
                    Count As Long,
                    ByRef PropertyIDs() as Long,
                    ByRef PropertyValues() As Variant,
                    ByRef Errors() As Long)
```

### **ItemID**

*ItemIDs* ist die ItemID, zu der eine Liste der Eigenschaften angefordert wird.

### **Count**

*Count* gibt die Anzahl der zurückgegebenen Eigenschaften an.

### **PropertyIDs**

*PropertyIDs* sind IDs des Typs DWORD für die gewünschten Eigenschaften. Diese IDs wurden von *QueryAvailableProperties* ausgegeben.

### **PropertyValues**

*PropertyValues* ist ein Array der Größe *Count* mit Variablen des Typs VARIANT, der vom Server ausgegeben wird; dieser Array enthält die aktuellen Werte der gewünschten Eigenschaften.

### **Errors**

*Errors* ist ein Array mit Fehlercodes, die anzeigen, ob jede Eigenschaft zurückgegeben wurde.

## Beispiel

```
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

        ' Festlegen der Werte für ItemCount und 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**

#### Beschreibung

Diese Methode liefert eine Liste der den übergebenen PropertyIDs entsprechenden ItemIDs (falls vorhanden). Die ItemIDs können in OPCGroups eingefügt und für einen effektiveren Zugriff auf Daten der jeweiligen Item-Eigenschaften verwendet werden. Ein Fehler innerhalb eines Fehler-Arrays kann ein Hinweis darauf sein, dass die zurückgegebene Eigenschafts-ID für das entsprechende Item nicht definiert ist.

## Syntax

```
LookupItemIDs (ItemID As String,
               Count As Long,
               PropertyIDs() as Long,
               ByRef NewItemIDs() As String,
               ByRef Errors () As Long)
```

### **ItemID**

*ItemID* zeigt die ItemID an, zu der eine Liste der Eigenschaften angefordert wird.

### **Count**

*Count* entspricht der Anzahl der zurückgegebenen Eigenschaften.

### **PropertyIDs**

*PropertyIDs* sind IDs des Typs DWORD für die gewünschten Eigenschaften. Diese IDs wurden von *QueryAvailableProperties* ausgegeben.

### **NewItemIDs**

*NewItemIDs* enthält die zurückgegebene Liste der ItemIDs.

### **Errors**

*Errors* ist ein Array mit Fehlercodes, die angeben, ob ItemIDs zurückgegeben wurden.

## Beispiel

```
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 = "EinOPCDataAccessItem"
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 Ereignisse des Objekts OPCServer

Für den OPC-Server existiert lediglich das Ereignis ServerShutdown.

### 7.2.3.1 ServerShutdown

#### Beschreibung

Dieses Ereignis wird ausgelöst, wenn der Server herunterfahren soll und alle aktiven Clients die Verbindung zum Server abbrechen sollen. Die Client-Anwendung unterstützt diesen Befehl, so dass der Client auf Befehl des Servers die Verbindung zum Server abbaut, wobei er auch alle Groups und Items abbauen sollte.

#### Syntax

```
ServerShutdown (Reason As String)
```

##### ServerReason

Dieser optionale Textstring des Servers gibt an, weshalb dieser herunterfährt.

#### Beispiel

```
Dim WithEvents AnOPCServer As OPCServer
Dim ARealOPCServer As String
Dim ARealOPCNodeName As String
Set AnOPCServer = New OPCServer
    ' dieses Beispiel ist nötig,
    ' um das Erzeugen eines Objekts mit dem Zusatze "WithEvents"
    ' zu erleichtern
ARealOPCServer = "Herstellername.DataAccessCustomServer"
ARealOPCNodeName = "EinComputername"
AnOPCServer.Connect(ARealOPCServer, ARealOPCNodeName)
Private Sub AnOPCServer_ServerShutdown
    (ByRef aServerReason As String)
    ' Programmcode zum Abbau der Verbindung zum Server
End Sub
```

## 7.3 Das Collection-Objekt OPCGroups

#### Beschreibung

Das Objekt OPCGroups ist eine Collection von Objekten der Klasse OPCGroup und der Methoden, mit denen diese erstellt, organisiert und entfernt werden.

Dieses Objekt verfügt über Eigenschaften für OPCGroup-Voreinstellungen. Wird ein Objekt der Klasse OPCGroup hinzugefügt, wird sein Anfangsstatus mit Hilfe der voreingestellten Eigenschaft DefaultGroupXXXX festgelegt. Die Voreinstellungen können verändert werden, um OPCGroups mit unterschiedlichem Anfangsstatus zu erzeugen. Veränderungen der Voreinstellungen haben keine Auswirkungen auf schon bestehende Group-Objekte. Sobald ein Group-Objekt hinzugefügt wurde, können dessen Eigenschaften verändert werden. Dadurch wird die Anzahl der Parameter, die zum Aufruf der Methode Add benötigt werden, reduziert.

## Beispiel

Der folgende Code wird benötigt, damit die anschließenden VB-Beispiele funktionieren.

```
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 AnOPCItemServerErrors() As Long
Set AnOPCServer = New OPCServer
ARealOPCServer = "Herstellername.DataAccessCustomServer"
ARealOPCNodeName = "EinComputername"
AnOPCServer.Connect(ARealOPCServer, ARealOPCNodeName)
Set MyGroups = AnOPCServer.OPCGroups
MyGroups.DefaultGroupIsActive = True
Set OneGroup = MyGroups.Add( "EinOPCGroupName" )
Set AnOPCItemCollection = OneGroup.OPCItems
```

### 7.3.1 Eigenschaften des Objekts OPCGroups

- Parent
- DefaultGroupIsActive
- DefaultGroupUpdateRate
- DefaultGroupDeadband
- DefaultGroupLocaleID
- DefaultGroupTimeBias
- Count

#### 7.3.1.1 Parent

##### Beschreibung

(Nur lesbar) Diese Eigenschaft gibt eine Referenz auf das übergeordnete OPCServer-Objekt zurück.

##### Syntax

```
Parent As OPCServer
```

### 7.3.1.2 DefaultGroupIsActive

#### Beschreibung

(Les- und schreibbar) Diese Eigenschaft gibt den Anfangswert für die Eigenschaft */sActive* einer neu generierten OPCGroup vor.

#### Syntax

```
DefaultGroupIsActive As Boolean
```

#### Bemerkungen

Die Standard-Einstellung dieser Eigenschaft ist *True*.

#### Beispiel

```
' Beispiel VB-Syntax (Ermitteln der Eigenschaft):  
Dim DefaultGroupIsActive As Boolean  
DefaultGroupIsActive = MyGroups.DefaultGroupIsActive  
  
' Beispiel VB-Syntax (Festlegen der Eigenschaft):  
MyGroups.DefaultGroupIsActive = FALSE
```

### 7.3.1.3 DefaultGroupUpdateRate

#### Beschreibung

(Les- und schreibbar) Diese Eigenschaft gibt den Anfangswert für die Eigenschaft *UpdateRate* einer neu generierten OPCGroup in Millisekunden vor; Defaulteinstellung: 1000 ms (= 1 Sekunde).

#### Syntax

```
DefaultGroupUpdateRate As Long
```

#### Beispiel

```
' Beispiel VB-Syntax (Ermitteln der Eigenschaft):  
Dim DefaultGroupUpdateRate As Long  
DefaultGroupUpdateRate = MyGroups.DefaultGroupUpdateRate  
  
' Beispiel VB-Syntax (Festlegen der Eigenschaft):  
MyGroups.DefaultGroupUpdateRate = 250
```

### 7.3.1.4 DefaultGroupDeadband

#### Beschreibung

(Les- und schreibbar) Diese Eigenschaft gibt den Anfangswert für die Eigenschaft *Deadband* für OPCGroups in Prozent vor, die mit *Add* erzeugt wurden. Die Eigenschaft *Deadband* wird prozentual zur gesamten Breite angegeben (gültige Werte von 0 bis 100%).

#### Syntax

```
DefaultGroupDeadband As Single
```

#### Bemerkungen

Die Standard-Einstellung bei dieser Eigenschaft ist "0". Ist der Wert >100 oder <0, wird eine Fehlermeldung ausgegeben.

#### Beispiel

```
' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim DefaultGroupDeadband As Single
DefaultGroupDeadband = MyGroups.DefaultGroupDeadband

' Beispiel VB-Syntax (Festlegen der Eigenschaft:
MyGroups.DefaultGroupDeadband = 10
```

### 7.3.1.5 DefaultGroupLocaleID

#### Beschreibung

(Les- und schreibbar) Diese Eigenschaft gibt den Anfangswert für die Eigenschaft *LocaleID* für OPCGroup vor, die mit *Add* erzeugt wurde.

#### Syntax

```
DefaultGroupLocaleID As Long
```

#### Bemerkungen

Die Standard-Einstellung dieser Eigenschaft stimmt mit den Einstellungen der *LocaleID* des Servers überein.

## Beispiel

```
' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim DefaultGroupLocaleID As Long
DefaultGroupLocaleID = MyGroups.DefaultGroupLocaleID

' Beispiel VB-Syntax (Festlegen der Eigenschaft):
MyGroups.DefaultGroupLocaleID =
    ConvertLocaleIdStringToLocaleIdLong("English")
```

### 7.3.1.6 DefaultGroupTimeBias

#### Beschreibung

(Les- und schreibbar). Diese Eigenschaft gibt den Anfangswert in Minuten für die Eigenschaft *TimeBias* (= Zeit-Offset) für Objekte der Klasse OPCGroup vor, die mit *Add* erzeugt wurden.

#### Syntax

```
DefaultGroupTimeBias As Long
```

#### Bemerkungen

Die Standard-Einstellung für diese Eigenschaft ist *0 Minuten*.

## Beispiel

```
' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim DefaultGroupTimeBias As Long
DefaultGroupTimeBias = MyGroups.DefaultGroupTimeBias

' Beispiel VB-Syntax (Festlegen der Eigenschaft):
MyGroups.DefaultGroupTimeBias = 60
```

### 7.3.1.7 Count

#### Beschreibung

(Nur lesbar) Dies ist die Anzahl der Gruppen. Diese Eigenschaft wird für Collection-Objekte benötigt.

#### Syntax

```
Count As Long
```

## Beispiel

```
' Beispiel VB-Syntax:  
For index = 1 to MyGroups.Count  
' weiterer Code  
Next index
```

### 7.3.2 Methoden des Objekts OPCGroups

Item  
Add  
GetOPCGroup  
Remove  
RemoveAll  
ConnectPublicGroup  
RemovePublicGroup

#### 7.3.2.1 Item

##### Beschreibung

Diese Methode liefert eine Referenz auf das angegebene Item. Das Item kann durch seinen Namen oder einen Index (beginnend bei 1) spezifiziert werden. GetOPCGroup wird zur Refenzierung nach ServerHandle benutzt. Bei OPCGroups ist *Item* die Standard-Methode.

##### Syntax

Item (ItemSpecifier As Variant) As OPCGroup

##### ItemSpecifier

*Item Specifier* ist entweder das ServerHandle der OPCGroup oder der Name einer OPCGroup. Um mit einem Index zu referenzieren, wird die Methode Item benutzt.

## Beispiel

```
' Beispiel VB-Syntax:  
Dim AnOPCGroup As OPCGroup  
Set AnOPCGroup = MyGroups.Item(3)  
  
' oder  
Set AnOPCGroup = MyGroups("Group3")
```

### 7.3.2.2 Add

#### Beschreibung

Diese Methode erzeugt ein neues OPCGroup-Objekt und fügt es der Collection hinzu. Die Eigenschaften dieser neuen Gruppe werden durch die aktuellen Standard-Einstellungen des Objekts OPCServer festgelegt. Nachdem eine Gruppe hinzugefügt wurde, können ihre Eigenschaften nach Bedarf verändert werden.

#### Syntax

```
Add (Optional Name As Variant) As OPCGroup
```

##### Name

*Name* gibt den Namen der Gruppe an. Der Name muss innerhalb aller anderen vom entsprechenden Client erzeugten Gruppen eindeutig sein. Wird kein Name angegeben, ist der vom Server generierte Name unter allen bestehenden Gruppen eindeutig.

#### Bemerkungen

Wird kein Name angegeben, erzeugt der Server selbst einen eindeutigen Namen. Wird ein Name angegeben, der nicht eindeutig ist, wird kein OPCGroup-Objekt erzeugt, und VB gibt eine Fehlermeldung aus, wenn dieses nicht erzeugte Objekt aufgerufen werden soll.

Weitere Informationen zu Fehlern und Ausnahmen finden Sie im Anhang zur Referenz Automation-Schnittstelle.

#### Beispiel

```
MyGroups.DefaultGroupIsActive = True  
Set OneGroup = MyGroups.Add("EinOPCGroupName")
```

### 7.3.2.3 GetOPCGroup

#### Beschreibung

Diese Methode liefert die Referenz auf eine durch den Namen oder das Server-Handle bezeichnete OPCGroup.

#### Syntax

```
GetOPCGroup (ItemSpecifier As Variant) As OPCGroup
```

##### ItemSpecifier

*ItemSpecifier* ist entweder der ServerHandle der OPCGroup oder der Name einer OPCGroup. Um mit einem Index zu referenzieren, wird die Methode Item benutzt.

## Beispiel

```
' Das Beispiel setzt voraus,
' dass "EinOPCGroupName" schon hinzugefügt wurde
Set OneGroup = MyGroups.GetOPCGroup("EinOPCGroupName")
```

### 7.3.2.4 Remove

#### Beschreibung

Diese Methode entfernt eine OPCGroup im Server.

#### Syntax

```
Remove (ItemSpecifier As Variant)
```

##### ItemSpecifier

*ItemSpecifier* ist entweder der ServerHandle der OPCGroup oder der Name einer OPCGroup. Um mit einem Index zu referenzieren, wird die Methode *Item* benutzt.

#### Bemerkungen

Ist die OPCGroup eine *PublicGroup*, kann diese Methode nicht angewendet werden. Weitere Informationen zu Fehlern und Ausnahmen finden Sie im Anhang zur Referenz Automation-Schnittstelle.

## Beispiel

```
Set OneGroup = MyGroups.Add("EinOPCGroupName")
' weiterer Code
MyGroups.Remove("EinOPCGroupName")

' oder
Set OneGroup = MyGroups.Add("EinOPCGroupName")
' weiterer Code
MyGroups.Remove(OneGroup.ServerHandle)
```

### 7.3.2.5 RemoveAll

#### Beschreibung

Entfernt alle existierenden Objekte der Klassen OPCGroup und OPCItem, um das Herunterfahren des Servers vorzubereiten.

#### Syntax

```
RemoveAll()
```

## Bemerkungen

Diese Methode erleichtert die komplette Freigabe von Unterobjekten durch den Client, wenn das Server-Objekt gelöscht wird. *RemoveAll* entspricht der Methode *Remove*, wenn diese für alle noch vorhandenen Objekte der Klassen OPCGroup und OPCItem aufgerufen wird. Da Objekte der Klasse OPCBrowser keine Unterobjekte des Servers sind, können sie über die Methode *RemoveAll* nicht entfernt werden.

## Beispiel

```
Set OneGroup = MyGroups.Add("EinOPCGroupName")
Set OneGroup = MyGroups.Add("EinOPCGroupName1")
Set OneGroup = MyGroups.Add("EinOPCGroupName2")
' weiterer Code
MyGroups.RemoveAll
```

### 7.3.2.6 ConnectPublicGroup

## Beschreibung

PublicGroups sind voreingestellte Gruppen eines Servers. Zu diesen Gruppen können Verbindungen aufgebaut werden, ohne dass sie zuvor dem Server hinzugefügt wurden.

## Syntax

```
ConnectPublicGroup (Name As String) As OPCGroup
```

### Name

*Name* ist der Name der Gruppe, die verbunden werden soll.

## Bemerkungen

Ist der Name ungültig oder wird PublicGroups nicht unterstützt, kann diese Methode nicht angewendet werden.

Weitere Informationen zu Fehlern und Ausnahmen finden Sie im Anhang zur Referenz Automation Schnittstelle.

## Beispiel

```
Set OneGroup = MyGroups.ConnectPublicGroup("AnOPCServerDefinedPublicGroup")
```

### 7.3.2.7 RemovePublicGroup

## Beschreibung

Diese Methode entfernt die OPCGroup, die als Parameter angegeben wurde.

## Syntax

```
RemovePublicGroup (ItemSpecifier As Variant)
```

### ItemSpecifier

*ItemSpecifier* ist entweder der ServerHandle der OPCGroup, der von *ConnectPublicGroup* zurückgegeben wurde, oder der Name einer OPCGroup.

## Bemerkungen

Wird *PublicGroups* nicht unterstützt oder wurde die Gruppe nicht über die Methode *ConnectPublicGroup* verbunden, kann *RemovePublicGroup* nicht angewendet werden. Weitere Informationen zu Fehlern und Ausnahmen finden Sie im Anhang zur Referenz Automation Schnittstelle.

## Beispiel

```
Set OneGroup = MyGroups.ConnectPublicGroup("EinOPCGroupName")
' weiterer Code
MyGroups.RemovePublicGroup("EinOPCGroupName")

'oder
Set OneGroup = MyGroups.ConnectPublicGroup ("EinOPCGroupName")
' weiterer Code
MyGroups.RemovePublicGroup(OneGroup.ServerHandle)
```

## 7.3.3 Ereignisse des Objekts OPCGroups

Für das Collection-Objekt OPCGroups existiert lediglich das Ereignis GlobalDataChange.

### 7.3.3.1 GlobalDataChange

#### Beschreibung

*GlobalDataChange* vereinfacht die Bearbeitung von Ereignissen über alle Gruppen der Collection hinweg, indem es über Werte- und Zustandsänderungen aller Items in allen Gruppen benachrichtigt.

## Syntax

```
GlobalDataChange (TransactionID As Long,
                  GroupHandle As Long,
                  NumItems As Long,
                  ClientHandles() As Long,
                  ItemValues() As Variant,
                  Qualities() As Long,
                  TimeStamps() As Date)
```

**TransactionID**

*TransactionID* ist die vom Client spezifizierte Transaktions-ID. Ist der Wert ungleich 0, ist dieses Ereignis auf Grund eines *AsyncRefreshes* erfolgt. Ist der Wert gleich 0, so ist dieses Ereignis aufgrund einer normalen Ereignisbehandlung erfolgt.

**GroupHandle**

*GroupHandle* ist das ClientHandle der OPCGroup, zu der die geänderten Daten gehören.

**NumItems**

*NumItems* gibt die Anzahl der ausgegebenen Items an.

**ClientHandles**

*ClientHandles* ist ein Array von Client-Handles für die Items.

**Item Values**

*ItemValues* ist ein Array mit Werten.

**Qualities**

*Qualities* ist ein Array der Qualität für den Wert jedes Items.

**TimeStamps**

*TimeStamps* ist ein Array mit UTC-Zeitstempeln für den Wert jedes Items.

## Bemerkungen

Bitte beachten Sie, dass das Ereignis *OnDataChange* bei einer OPCGroup normal genutzt werden sollte. Dieses Ereignis ermöglicht es, ein EventHandle so einzurichten, dass es Datenänderungen von mehreren Objekten der Klasse OPCGroups verarbeiten kann.

Normalerweise besitzt eine Anwendung für jede Gruppe ein EventHandle, um Datenänderungen zu empfangen und zu bearbeiten. So braucht man nur ein EventHandle, und mit dem Gebrauch des *GroupHandle* steht fest, für welche Gruppe das Ereignis ausgelöst wurde.

Das Ereignis GlobalDataChange wird für jedes Objekt der Klasse OPCGroup ausgelöst, das Items enthält, deren Werte oder Zustand sich seit dem Zeitpunkt, zu dem das Ereignis zuletzt ausgelöst wurde, verändert hat. Das jeweilige Ereignis des OPCGroup-Objekts wird ebenfalls ausgelöst. Wenn die Anwendung sowohl globale als auch gruppenspezifische Ereignisse verwendet, erhält sie zweimal eine Benachrichtigung: einmal für das globale und einmal für das gruppenspezifische Ereignis.

## Beispiel

```

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)

    ' Geben Sie hier den Clientcode zum Bearbeiten von geänderten
    ' Werten ein
    .
    .
End Sub

```

## 7.4 Das Objekt OPCGroup

### Beschreibung

Mit Hilfe von OPCGroup kann ein Client Daten verwalten. Die Gruppe kann beispielsweise Items an einer bestimmten Benutzerschnittstelle oder in einem Bericht repräsentieren. Daten können gelesen und geschrieben werden. Rückmeldungen der Items in der Gruppe an den Client können nach Bedarf aktiviert und deaktiviert werden. Der Client kann die Aktualisierungszeit der Datenänderung im OPC-Server konfigurieren.

### Syntax

OPCGroup

## Beispiel

Der folgende Code wird benötigt, damit die anschließenden VB-Beispiele funktionieren.

```

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 AnOPCItemServerErrors() As Long
Set AnOPCServer = New OPCServer
ARealOPCServer = "Herstellername.DataAccessCustomServer"
ARealOPCNodeName = "EinComputername"
AnOPCServer.Connect(ARealOPCServer, ARealOPCNodeName)
Set MyGroups = AnOPCServer.OPCGroups
MyGroups.DefaultGroupIsActive = True
Set OneGroup = MyGroups.Add("EinnOPCGroupName")
Set AnOPCItemCollection = OneGroup.OPCItems
For x = 1 To AddItemCount
    ClientHandles(x) = x + 1
    AnOPCItemID(x) = "Register_" & x
Next x
AnOPCItemCollection.AddItemCount, AnOPCItemIDs,
AnOPCItemServerHandles, AnOPCItemServerErrors
```

## 7.4.1 Eigenschaften des Objekts OPCGroup

Parent  
Name  
IsPublic  
IsActive  
IsSubscribed  
ClientHandle  
ServerHandle  
LocaleID  
TimeBias  
DeadBand  
UpdateRate  
OPCItems

### 7.4.1.1 Parent

#### Beschreibung

(Nur Lesbar) Diese Eigenschaft gibt die Referenz an das übergeordnete OPCServer-Objekt zurück.

#### Syntax

```
Parent As OPCServer
```

### 7.4.1.2 Name

#### Beschreibung

(Les- und schreibbar) *Name* enthält den Namen der Gruppe.

## Syntax

Name As String

### Name

*Name* enthält den Namen der Gruppe. Der Name muss innerhalb aller anderen vom Client erzeugten Gruppen eindeutig sein.

## Bemerkungen

Gruppen können optional einen Namen erhalten. Mit dieser Eigenschaft kann der User einen Namen vergeben, der eindeutig sein muss. Wird kein Name angegeben, erzeugt der Server durch die Methode *Add* des Objekts OPCGroups einen eindeutigen Namen für die Gruppe.

## Beispiel

```
' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As String
Set OneGroup = MyGroups.Add("EinOPCGroupName")
CurrentValue = OneGroup.Name

' Beispiel VB-Syntax (Festlegen der Eigenschaft):
Set OneGroup = MyGroups.Add("EinOPCGroupName")
OneGroup.Name = "EinName"
```

### 7.4.1.3 IsPublic

## Beschreibung

(Nur lesbar) Diese Eigenschaft gibt *True* zurück, wenn es sich um eine PublicGroup handelt, ansonsten *False*.

## Syntax

IsPublic As Boolean

## Beispiel

```
Dim CurrentValue As Boolean
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("EinOPCGroupName")
' weiterer Code
Set CurrentValue = OneGroup.IsPublic
' so erhält man den Wert
```

#### 7.4.1.4 IsActive

##### Beschreibung

(Les- und schreibbar) Diese Eigenschaft legt fest, ob eine Gruppe aktiv ist. Eine Gruppe, die aktiv ist, erhält Daten. Eine inaktive Gruppe erhält normalerweise keine Daten, es sei denn, dies wird für les- und schreibbare Operationen notwendig.

##### Syntax

```
IsActive As Boolean
```

##### Bemerkungen

Die Standard-Einstellung für diese Eigenschaft ist der dem Wert des OPCGroups-Objekts entsprechende Standardwert zum Zeitpunkt *Add()*.

##### Beispiel

```
' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Boolean
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("EinOPCGroupName")
' weiterer Code
Set CurrentValue = OneGroup.IsActive
' so erhält man den Wert

' Beispiel VB-Syntax (Festlegen der Eigenschaft):
Dim CurrentValue As Boolean
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("EinOPCGroupName")
' weiterer Code
OneGroup.IsActive = True
```

#### 7.4.1.5 IsSubscribed

##### Beschreibung

(Les- und schreibbar) Diese Eigenschaft kontrolliert asynchrone Nachrichten an die Gruppe. Eine Gruppe, die im Server registriert ist, erhält vom Server Mitteilungen über geänderte Daten.

##### Syntax

```
IsSubscribed As Boolean
```

## Bemerkungen

Die Standard-Einstellung für diese Eigenschaft ist der dem Wert des OPCGroups-Objekts entsprechende Standardwert zum Zeitpunkt *Add()*.

## Beispiel

```
' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Boolean
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("EinOPCGroupName")
' weiterer Code
Set CurrentValue = OneGroup.IsSubscribed
' so erhält man den Wert

' Beispiel VB-Syntax (Festlegen der Eigenschaft):
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("EinOPCGroupName")
' weiterer Code
OneGroup.IsSubscribed = True
' so wird der Wert festgelegt
```

### 7.4.1.6 ClientHandle

#### Beschreibung

(Les- und schreibbar) *ClientHandle* ist ein Long-Wert, der für die Gruppe steht. Mit dieser Eigenschaft kann der Client schnell den Empfänger der Daten feststellen. Dieser Wert ist normalerweise ein Index o. ä. und wird zusammen mit Daten oder Angaben über den Status an den Client zurückgegeben.

#### Syntax

```
ClientHandle As Long
```

## Beispiel

```
' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Long
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup =
    MyGroups.ConnectPublicGroup ("EinOPCGroupName")
' weiterer Code
Set CurrentValue = OneGroup.ClientHandle
' so erhält man den Wert

' Beispiel VB-Syntax (Festlegen der Eigenschaft):
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup =
    MyGroups.ConnectPublicGroup ("EinOPCGroupName")
' weiterer Code
OneGroup.ClientHandle = 1975
' so wird der Wert festgelegt
```

### 7.4.1.7 ServerHandle

#### Beschreibung

(Nur lesbar) Dies ist das Handle, das der Server der Gruppe zugewiesen hat. Der Wert dieser Eigenschaft wird als Long-Wert angegeben; die Gruppe kann damit eindeutig identifiziert werden. Der Client muss dieses Handle einigen Methoden als Parameter übergeben, die mit Objekten der Klasse OPCGroup arbeiten (z. B. *OPCGroups.Remove*).

#### Syntax

```
ServerHandle As Long
```

## Beispiel

```
' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Long
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("EinOPCGroupName")
' weiterer Code
Set CurrentValue = OneGroup.ServerHandle
' so erhält man den Wert
```

#### 7.4.1.8 LocaleID

##### Beschreibung

(Les- und schreibbar) Diese Eigenschaft identifiziert die SprachID; mit dieser Eigenschaft können Strings lokalisiert werden, die vom Server zurückgegeben werden. Die Standardeinstellung dieser Eigenschaft ist abhängig vom Wert, der in der Collection OPCGroups (*DefaultGroupLocaleID*) definiert ist.

##### Syntax

```
LocaleID As Long
```

##### Beispiel

```
' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Long
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("EinOPCGroupName")
' weiterer Code
Set CurrentValue = OneGroup.LocaleID

' Beispiel VB-Syntax (Festlegen der Eigenschaft):
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("EinOPCGroupName")
' weiterer Code
OneGroup.LocaleID = StringToLocaleID("English")
```

#### 7.4.1.9 TimeBias

##### Beschreibung

(Les- und schreibbar) Diese Eigenschaft wird benötigt, um den Zeitstempel der Daten in die lokale Zeit des Geräts umzuwandeln. Einheit: Millisekunden.

##### Syntax

```
TimeBias As Long
```

## Beispiel

```
' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Long
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("EinOPCGroupName")
' weiterer Code
Set CurrentValue = OneGroup.TimeBias

' Beispiel VB-Syntax (Festlegen der Eigenschaft):
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("EinOPCGroupName")
' weiterer Code
OneGroup.TimeBias = 100
```

### 7.4.1.10 DeadBand

#### Beschreibung

(Les- und schreibbar) Der Wert dieser Eigenschaft wird prozentual zur gesamten Bandbreite angegeben (gültige Werte von 0 bis 100). Die Standardeinstellung dieser Eigenschaft ist abhängig vom Wert, der in der Collection OPCGroups (*DefaultGroupDeadband*) definiert ist.

#### Syntax

```
DeadBand As Single
```

## Beispiel

```
' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Single
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("EinOPCGroupName")
' weiterer Code
Set CurrentValue = OneGroup.DeadBand

' Beispiel VB-Syntax (Festlegen der Eigenschaft):
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("EinOPCGroupName")
' weiterer Code
OneGroup.DeadBand = 5
```

#### 7.4.1.11 UpdateRate

##### Beschreibung

(Les- und schreibbar) Die maximale Frequenz (in ms), mit der DataChange-Ereignisse ausgelöst werden. Ein langsamer Prozess kann dazu führen, dass Datenänderungen weniger häufig vorkommen als mit dieser Frequenz, generell jedoch nie häufiger. Die Standardeinstellung dieser Eigenschaft ist abhängig vom Wert, der in der Collection OPCGroups definiert ist (*DefaultGroupUpdateRate*). Neue Werte für diese Eigenschaft werden mit einer Anfrage nach einem neuen Wert festgelegt. Möglicherweise unterstützt der Server eine bestimmte Frequenz nicht, so dass das Ergebnis eine andere Frequenz ist, wenn die Eigenschaft eingelesen wird (der Server verwendet die nächst größere Frequenz als die geforderte).

##### Syntax

```
UpdateRate As Long
```

##### Beispiel

```
' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Long
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("EinOPCGroupName")
' weiterer Code
DefaultGroupUpdateRate = OneGroup.UpdateRate

' Beispiel VB-Syntax (Festlegen der Eigenschaft):
Set MyGroups = AnOPCServer.OPCGroups
Set OneGroup = MyGroups.ConnectPublicGroup("EinOPCGroupName")
' weiterer Code
OneGroup.UpdateRate = 50
```

#### 7.4.1.12 OPCItems

##### Beschreibung

OPCItems ist eine Sammlung von OPCItem-Objekten; dies ist die Standard-Eigenschaft des Objekts OPCGroup.

##### Syntax

```
OPCItems As OPCItems
```

## Beispiel

```
' Beispiel VB-Syntax (Ermitteln der Eigenschaft):  
Dim AnOPCItemCollection As OPCItems  
Set MyGroups = AnOPCServer.OPCGroups  
Set OneGroup = MyGroups.ConnectPublicGroup("EinOPCGroupName")  
' weiterer Code  
Set AnOPCItemCollection = OneGroup.OPCItems
```

## 7.4.2 Methoden des Objekts OPCGroup

SyncRead  
SyncWrite  
AsyncRead  
AsyncWrite  
AsyncRefresh  
AsyncCancel

### 7.4.2.1 SyncRead

#### Beschreibung

Diese Funktion liest Werte, Qualität und Zeitstempel für ein Item oder auch mehrere Items einer Gruppe.

#### Syntax

```
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* ist die Datenquelle; OPC\_DS\_CACHE oder OPC\_DS\_DEVICE

##### NumItems

*NumItems* enthält die Anzahl der zu lesenden Items.

##### ServerHandles

*ServerHandles* ist ein Array von ServerHandles für die zu lesenden Items.

##### Values

*Values* ist ein Array von Werten.

##### Errors

*Errors* ist ein Array von Long-Werten, die angeben, ob die jeweiligen Items erfolgreich gelesen wurden, d. h., ob durch die Methode *Read* ein definierter Wert, eine Qualität und ein Zeitstempel erhalten wurden.

Hinweis: Wird im Fehlercode *FAILED* ausgegeben, sind Werte, Qualität und Zeitstempel nicht definiert (UNDEFINED)!

#### **Qualities**

*Qualities* enthält einen Wert vom Typ VARIANT, der ein Integer-Array mit Werten zur Qualität enthält.

#### **TimeStamps**

*TimeStamps* enthält einen Wert vom Typ VARIANT, der einen Daten-Array mit UTC-Zeitstempeln enthält. Kann das Gerät keinen Zeitstempel liefern, wird er vom Server erzeugt.

### **Bemerkungen**

Die Funktion wird abgeschlossen, bevor das Ergebnis ausgegeben wird. Die Daten können aus dem Cache gelesen werden (CACHE); in diesem Fall sollten die Daten innerhalb der UpdateRate und der Eigenschaft *Deadband* der Gruppe übereinstimmen. Die Daten können auch aus dem Gerät gelesen werden (DEVICE); dann soll ein tatsächliches Lesen aus dem eigentlichen Gerät durchgeführt werden. Auf die genaue Implementierung der Lesemethoden CACHE und DEVICE wird hier nicht weiter eingegangen.

Die Daten, die aus dem Cache gelesen werden, sind nur dann gültig, wenn sowohl die Gruppe als auch das Item aktiv sind. Ist eines dieser beiden Elemente inaktiv, wird anstatt der Qualität die Konstante OPC\_QUALITY\_OUT\_OF\_SERVICE ausgegeben.

Der Zustand von Group oder Item (aktiv/inaktiv) hat keinen Einfluss auf die Lesemethode DEVICE.

## Beispiel

```

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
    ' legt fest, welches Item gelesen werden soll
    ServerHandles(ServerIndex) = AnOPCItemServerHandles(ServerIndex)
Next ServerIndex
OneGroup.SyncRead Source,
    NumItems,
    ServerHandles,
    Values,
    Errors,
    Qualities,
    TimeStamps
For ServerIndex = 1 to NumItems
    ' verarbeitet die Werte
    TextBox(ServerIndex).Text = Values(ServerIndex)
Next ServerIndex
End Sub

```

### 7.4.2.2 SyncWrite

#### Beschreibung

Diese Methode schreibt Werte für ein oder mehrere Items einer Gruppe. Die Funktion wird vollständig ausgeführt. Die Werte werden an DEVICE geschrieben; dies bedeutet, daß die Funktion nicht zurückgegeben werden sollte, bis sichergestellt ist, daß das Gerät die Daten akzeptiert (bzw. nicht angenommen) hat.

#### Syntax

```

SyncWrite (NumItems As Long,
          ServerHandles() As Long,
          Values() As Variant,
          ByRef Errors() As Long)

```

##### **NumItems**

*NumItems* gibt die Anzahl der zu schreibenden Items an.

##### **ServerHandles**

*ServerHandles* ist ein Array von ServerHandles für die zu schreibenden Items.

**Values**

*Values* ist ein Array von Werten.

**Errors**

*Errors* ist ein Array von Long-Werten, die angeben, ob die jeweiligen Items erfolgreich geschrieben wurden.

**Bemerkungen**

Der Zustand von Group oder Item (aktiv/inaktiv) hat keinen Einfluss auf die Schreibmethode.

**Beispiel**

```
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
    ' legt fest, welches Item geschrieben werden soll
    ServerHandles(ServerIndex) = AnOPCItemServerHandles(ServerIndex)
    Values(ServerIndex) = ServerIndex * 2
    ' der Wert des Items ist hier beliebig
Next ServerIndex
OneGroup.SyncWrite NumItems, ServerHandles, Values, Errors
For ServerIndex = 1 to NumItems
    ' verarbeitet die Fehler
    TextBox(ServerIndex).Text = Errors(ServerIndex)
Next ServerIndex
End Sub
```

**7.4.2.3 AsyncRead****Beschreibung**

Diese Methode liest ein Item oder mehrere Items einer Gruppe. Die Ergebnisse werden mittels des Ereignisses *AsyncReadComplete*, das dem Objekt OPCGroup zugeordnet ist, zurückgegeben.

**Syntax**

```
AsyncRead (NumItems As Long,
           ServerHandles() As Long,
           ByRef Errors() As Long,
           TransactionID As Long,
           ByRef CancelID As Long)
```

**NumItems**

*NumItems* enthält die Anzahl der zu lesenden Items.

**ServerHandles**

*ServerHandles* ist ein Array von ServerHandles für die zu lesenden Items.

**Errors**

*Errors* ist ein Array von Long-Werten, der den Status des jeweiligen Items angibt, das gelesen werden soll.

**TransactionID**

*TransactionID* ist die vom Client spezifizierte Transaktions-ID. Diese ID ist auch im entsprechenden Ereignis enthalten.

**CancelID**

*CancelID* ist eine vom Server generierte Transaktions-ID, mit der der Client die Transaktion abbrechen kann.

**Bemerkungen**

Die Methode *AsyncRead* setzt voraus, daß das Objekt der Klasse OPCGroup mit Ereignissen (Dim WithEvents xxx As OPCGroup) deklariert wird, so dass das Ergebnis von *AsyncRead* zur Client-Anwendung zurückgegeben wird. Das Ereignis *AsyncReadComplete*, das dem Objekt OPCGroup zugeordnet ist, wird vom Automation-Server mit dem Ergebnis aus *AsyncRead* ausgelöst.

Der Zustand von Group oder Item (aktiv/inaktiv) hat keinen Einfluss auf die Lesemethode DEVICE.

Für weitere Informationen s. *IOPCAsyncIO2::Read* in der Spezifikation *OPC Data Access Custom Interface*.

**Beispiel**

```
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
    ' legt fest, welches Item gelesen werden soll
    ServerHandles(ServerIndex) = AnOPCItemServerHandles(ServerIndex)
Next ServerIndex
OneGroup.AsyncRead NumItems, ServerHandles, Errors,
                  ClientTransactionID, ServerTransactionID
End Sub
```

#### 7.4.2.4 AsyncWrite

##### Beschreibung

Diese Methode schreibt Werte für ein oder mehrere Items in einer Gruppe. Die Ergebnisse werden über das Ereignis *AsyncWriteComplete* zurückgegeben, das dem Objekt OPCGroup zugeordnet ist.

##### Syntax

```
AsyncWrite (NumItems As Long,
           ServerHandles() As Long,
           Values() As Variant,
           ByRef Errors() As Long,
           TransactionID As Long,
           ByRef CancelID As Long)
```

##### NumItems

*NumItems* gibt die Anzahl der zu schreibenden Items an.

##### ServerHandles

*ServerHandles* ist ein Array von ServerHandles für die zu schreibenden Items.

##### Values

*Values* ist ein Array von Werten.

##### Errors

*Errors* ist ein Array von Long-Werten, der den Status des jeweiligen Items angibt, das geschrieben werden soll.

##### TransactionID

*TransactionID* ist die vom Client spezifizierte Transaktions-ID. Diese ID ist auch im entsprechenden Ereignis enthalten.

##### CancelID

*CancelID* ist eine vom Server generierte Transaktions-ID, mit der der Client die Transaktion abbrechen kann.

##### Bemerkungen

Die Methode *AsyncWrite* setzt voraus, daß das Objekt der Klasse OPCGroup mit Ereignissen (Dim WithEvents xxx As OPCGroup) dimensioniert wurde, so dass das Ergebnis der Operation *AsyncWrite* zur Automation-Client-Anwendung zurückgegeben werden kann. Das Ereignis *AsyncWriteComplete*, das dem Objekt OPCGroup zugeordnet ist, wird vom Automation-Server mit den Ergebnissen aus der Operation *AsyncWrite* ausgelöst.

Für weitere Informationen s. *IOPCAsyncIO2::Write* in der Spezifikation *OPC Data Access Custom Interface*.

## Beispiel

```

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
        ' legt fest, welches Item geschrieben werden soll
        ServerHandles(ServerIndex) = AnOPCItemServerHandles(ServerIndex)
        Values(ServerIndex) = ServerIndex * 2
        ' der Wert des Items ist hier beliebig
    Next ServerIndex
    OneGroup.AsyncWrite NumItems, ServerHandles, Values, Errors,
                        ClientTransactionID, ServerTransactionID
End Sub

```

### 7.4.2.5 AsyncRefresh

#### Beschreibung

Diese Methode erzeugt ein Ereignis für alle aktiven Items einer Gruppe (zeigt an, ob sie sich verändert haben oder nicht). Inaktive Items werden nicht aufgeführt. Das Ergebnis wird sowohl mittels des Ereignisses *DataChange* ausgegeben, das dem Objekt OPCGroup zugeordnet ist, als auch mittels des Ereignisses *GlobalDataChange*, das dem Objekt OPCGroups zugeordnet ist.

#### Syntax

```

AsyncRefresh (Source As Integer,
              TransactionID As Long,
              ByRef CancelID As Long)

```

##### Source

*Source* ist die Datenquelle; OPC\_DS\_CACHE oder OPC\_DS\_DEVICE

##### TransactionID

*TransacionID* ist die vom Client spezifizierte Transaktions-ID. Diese ID ist auch im entsprechenden Ereignis enthalten.

##### CancelID

*CancelID* ist eine vom Server generierte Transaktions-ID, mit der der Client die Transaktion abbrechen kann.

## Bemerkungen

Die Methode *AsyncRefresh* setzt voraus, daß das Objekt der Klasse OPCGroup mit Ereignissen (Dim WithEvents xxx As OPCGroup) dimensioniert wurde, so dass das Ergebnis der Operation *AsyncRefresh* zur Automation-Client-Anwendung zurückgegeben wird. Das Ereignis *DataChange*, das dem Objekt OPCGroup zugeordnet ist, wird vom Automation-Server mit den Ergebnissen aus der Operation *AsyncRefresh* ausgelöst.

Wenn die Client-Anwendung das Objekt der Klasse OPCGroups mit dem Zusatz *WithEvents* deklariert hat (Dim WithEvents xyz As OPCGroups), wird das Ereignis *GlobalDataChange* für diese Gruppe mit den Ergebnissen der Datenaktualisierung ausgelöst.

Für weitere Informationen s. *IOPCAsyncIO2::Refresh* in der Spezifikation *OPC Data Access Custom Interface*.

## Hinweis

Der Aufruf der Funktion *Async2RefreshDevice* der OPC-Automation-Schnittstelle liefert alle Items, auch "nur schreibbare".

## Beispiel

```
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

#### Beschreibung

Durch diese Methode wird der Server aufgefordert, ausstehende Transaktionen zu annullieren. Das Ereignis *AsyncCancelComplete* gibt an, ob die Annulierung durchgeführt wurde.

#### Syntax

`AsyncCancel (CancelID As Long)`

#### CancelID

*CancelID* ist eine vom Server erzeugte CancelID, die von einer Methode *AsyncRead*, *AsyncWrite* oder *AsyncRefresh* zurückgegeben wurde.

## Bemerkungen

Die Methode *AsyncCancel* setzt voraus, daß das Objekt der Klasse OPCGroup mit Ereignissen (Dim WithEvents xxx As OPCGroup) deklariert wurde, so dass das Ergebnis der Operation *AsyncCancel* zur Automation-Client-Anwendung zurückgegeben wird. Das Ereignis *AsyncCancelComplete*, das dem Objekt OPCGroup zugeordnet ist, wird vom Automation-Server mit den Ergebnissen aus der Operation *AsyncCancel* ausgelöst. Die vom Client spezifizierte Transaktions-ID (TransactionID) wird mit dem Ereignis *AsyncCancelComplete* an die Automation-Client-Anwendung zurückgegeben.

Für weitere Informationen siehe *IOPCAsyncIO2::Cancel* in der Spezifikation *OPC Data Access Custom Interface*.

## Beispiel

```
Private Sub AsyncCancelButton_Click()
    Dim ServerIndex As Long
    Dim CancelID As Long
    CancelID = 1 ' eine Transaktions-ID
    ' aus einem der "Async"-Aufrufe
    ' wie AsyncRead, AsyncWrite oder AsyncRefresh
    OneGroup.AsyncCancel CancelID
End Sub
```

## 7.4.3 Ereignisse des Objekts OPCGroup

- DataChange
- AsyncReadComplete
- AsyncWriteComplete
- AsyncCancelComplete

### 7.4.3.1 DataChange

#### Beschreibung

Dieses Ereignis wird dann ausgelöst, wenn sich der Wert oder die Qualität eines Wertes eines Items innerhalb der Gruppe geändert hat. Das Ereignis wird nicht schneller ausgelöst als es die Aktualisierungsrate der Gruppe vorgibt. Dies bedeutet, dass die Werte der Items vom Server so lange gepuffert werden, bis aktuelle Zeit + Aktualisierungsrate größer sind als der Zeitpunkt des vorhergegangenen Updates. Dieses Verhalten hängt auch davon ab, ob die Gruppe und die Items aktiv sind. Damit Werte in einem Ereignis an den Client geschickt werden, müssen sowohl die Items als auch die zugehörige Gruppe aktiv sein.

## Syntax

```
DataChange (TransactionID As Long,
           NumItems As Long,
           ClientHandles() As Long,
           ItemValues() As Variant,
           Qualities() As Long,
           TimeStamps() As Date)
```

### TransactionID

*TransactionID* ist die vom Client spezifizierte Transaktions-ID. Wird ein Wert ungleich 0 ausgegeben, wurde der Aufruf als Ergebnis der Methode *AsyncRefresh* generiert. Ist der Wert gleich 0, so wurde der Aufruf als Ergebnis einer normalen Ereignisbehandlung generiert.

### NumItems

*NumItems* gibt die Anzahl der ausgegebenen Items an.

### ClientHandles

*ClientHandles* ist ein Array von Client-Handles für die Items.

### Item Values

*ItemValues* ist ein Array mit Werten.

### Qualities

*Qualities* ist ein Array der Qualität für den Wert jedes Items.

### TimeStamps

*TimeStamps* ist ein Array mit UTC-Zeitstempeln für den Wert jedes Items. Kann das Gerät keinen Zeitstempel liefern, übernimmt dies der Server.

## Bemerkungen

Ändert sich der Wert des Items schneller als die Aktualisierungsrate, wird nur der letzte aktuelle Wert für jedes Item gepuffert und im Ereignis an den Client zurückgegeben.

## Beispiel

```
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)
    ' fügen Sie hier Ihren Programmcode ein,
    ' um die geänderten Daten zu bearbeiten
End Sub
```

### 7.4.3.2 AsyncReadComplete

## Beschreibung

Dieses Ereignis wird ausgelöst, wenn die Methode *AsyncRead* abgeschlossen ist.

## Syntax

```
AsyncReadComplete (TransactionID As Long,
                  NumItems As Long,
                  ClientHandles() As Long,
                  ItemValues() As Variant,
                  Qualities() As Long,
                  TimeStamps() As Date,
                  Errors() As Long)
```

### **TransactionID**

*TransactionID* ist die vom Client spezifizierte Transaktions-ID.

### **NumItems**

*NumItems* gibt die Anzahl der ausgegebenen Items an.

### **ClientHandles**

*ClientHandles* ist ein Array von Client-Handles für die Items.

### **ItemValues**

*ItemValues* ist ein Array mit Werten.

### **Qualities**

*Qualities* ist ein Array der Qualität für den Wert jedes Items.

### **TimeStamps**

*TimeStamps* ist ein Array mit UTC-Zeitstempeln für den Wert jedes Items. Liefert das Gerät keinen Zeitstempel, übernimmt dies der Server.

### **Errors**

*Errors* ist ein Array von Long-Werten, die angeben, ob die jeweiligen Items erfolgreich gelesen wurden, d. h., ob durch die Methode *Read* ein definierter Wert, eine Qualität und ein Zeitstempel erhalten wurde. Wird im Fehlercode *FAILED* ausgegeben, sind Werte, Qualität und Zeitstempel nicht definiert (*UNDEFINED*)!

## Beispiel

```
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)
  ' fügen Sie hier Ihren Programmcode ein,
  ' um die geänderten Daten zu bearbeiten
End Sub
```

### 7.4.3.3 AsyncWriteComplete

## Beschreibung

Dieses Ereignis wird ausgelöst, wenn die Methode *AsyncWrite* abgeschlossen ist.

## Syntax

```
AsyncWriteComplete (TransactionID As Long,
                   NumItems As Long,
                   ClientHandles() As Long,
                   Errors() As Long)
```

### TransactionID

*TransactionID* ist die vom Client spezifizierte Transaktions-ID.

### NumItems

*NumItems* gibt die Anzahl der ausgegebenen Items an.

### ClientHandles

*ClientHandles* ist ein Array von Client-Handles für die Items.

### Errors

*Errors* ist ein Array von Long-Werten, die angeben, ob die jeweiligen Items erfolgreich geschrieben wurden.

## Beispiel

```
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)
  ' fügen Sie hier Ihren Programmcode ein,
  ' um die Fehler zu bearbeiten
End Sub
```

### 7.4.3.4 AsyncCancelComplete

## Beschreibung

Dieses Ereignis wird ausgelöst, wenn die Methode AsyncCancel abgeschlossen ist.

## Syntax

```
AsyncCancelComplete (TransactionID As Long)
```

### TransactionID

*TransactionID* ist die vom Client spezifizierte Transaktions-ID.

## Beispiel

```
Dim WithEvents AnOPCGroup As OPCGroup
Private Sub AnOPCGroup_AsyncCancelComplete(TransactionID As Long)
  ' fügen Sie hier Ihren Programmcode ein,
  ' der nach dem Abbruch der Operation ausgeführt werden soll
End Sub
```

## 7.5 Das Collection-Objekt OPCItems

### Beschreibung

Dieses Collection-Objekt verfügt über Eigenschaften für die Default-Werte für Objekte der Klasse OPCItem. Wird ein solches Item hinzugefügt, erhalten die Eigenschaften des Items diese Default-Werte (DefaultXXXX) als Anfangsstatus. Diese Voreinstellungen können verändert werden, wenn Items mit anderem Anfangsstatus hinzugefügt werden sollen. Natürlich können die Eigenschaften eines Items auch noch nach dem Hinzufügen geändert werden. Dadurch wird die Anzahl der Parameter, die zum Aufruf der Methode *AddItems* benötigt werden, reduziert.

### Syntax

OPCItems

### Beispiel

Der folgende Code wird benötigt, damit die anschließenden VB-Beispiele funktionieren:

```
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 = "Herstellername.DataAccessCustomServer"
ARealOPCNodeName = "EinComputername"
AnOPCServer.Connect(ARealOPCServer, ARealOPCNodeName)
Set MyGroups = AnOPCServer.OPCGroups
MyGroups.DefaultGroupIsActive = True
Set OneGroup = MyGroups.Add("EinOPCGruppenname")
Set AnOPCItemCollection = OneGroup.OPCItems
```

### 7.5.1 Eigenschaften des Collection-Objekts OPCItems

- Parent
- DefaultRequestedDataType
- DefaultAccessPath
- DefaultIsActive
- Count

### 7.5.1.1 Parent

#### Beschreibung

(Nur lesbar) Diese Eigenschaft gibt die Referenz an das übergeordnete OPCGroup-Objekt zurück.

#### Syntax

```
Parent As OPCGroup
```

### 7.5.1.2 DefaultRequestedDataType

#### Beschreibung

(Les- und schreibbar) *DefaultRequestedDataType* ist der angeforderte Datentyp, der in *Add*-Aufrufen (*AddItem*, *AddItems*) benutzt wird, d. h., der Anfangswert für die Eigenschaft *RequestedDataType* neu hinzugefügter Items. Die Standard-Einstellung für diese Eigenschaft ist VT\_EMPTY (d. h., der Server sendet Daten in dem ihm eigenen Datenformat).

#### Syntax

```
DefaultRequestedDataType As Integer
```

#### Bemerkungen

Jeder gültige VARIANT-Typ kann als angeforderter Datentyp angegeben werden. Informationen zu Fehlern und Ausnahmen finden Sie im Anhang zur Referenz Automation-Schnittstelle.

#### Beispiel

```
' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Integer
Dim SomeValue As Integer
CurrentValue = AnOPCItemCollection.DefaultRequestedDataType

' Beispiel VB-Syntax (Festlegen der Eigenschaft):
AnOPCItemCollection.DefaultRequestedDataType = SomeValue
```

### 7.5.1.3 DefaultAccessPath

#### Beschreibung

(Les- und schreibbar) *DefaultAccessPath* ist der voreingestellte Access-Path, der in *Add*-Aufrufen (*AddItem*, *AddItems*) benutzt wird. Die Standard-Einstellung für diese Eigenschaft ist "" (ein Leerstring).

#### Syntax

```
DefaultAccessPath As String
```

#### Beispiel

```
' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As String
Dim SomeValue As String
CurrentValue = AnOPCItemCollection.DefaultAccessPath

' Beispiel VB-Syntax (Festlegen der Eigenschaft):
AnOPCItemCollection.DefaultAccessPath = SomeValue
```

### 7.5.1.4 DefaultIsActive

#### Beschreibung

(Les- und schreibbar) Diese Eigenschaft gibt den Anfangswert für die Eigenschaft *ActiveState* neu hinzugefügter Items vor. Die Voreinstellung für diese Eigenschaft ist *True*.

#### Syntax

```
DefaultIsActive As Boolean
```

#### Beispiel

```
' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Boolean
Dim SomeValue As Boolean
CurrentValue = AnOPCItemCollection.DefaultIsActive

' Beispiel VB-Syntax (Festlegen der Eigenschaft):
AnOPCItemCollection.DefaultIsActive = SomeValue
```

### 7.5.1.5 Count

#### Beschreibung

(Nur lesbar) Diese Eigenschaft wird für Collection-Objekte benötigt.

#### Syntax

```
Count As Long
```

#### Beispiel

```
' Beispiel VB-Syntax (Ermitteln der Eigenschaft):  
Dim CurrentValue As Long  
Dim SomeValue As Long  
CurrentValue = AnOPCItemCollection.Count
```

## 7.5.2 Methoden des Collection-Objekts OPCItems

```
Item  
GetOPCItem  
AddItem  
AddItems  
Remove  
Validate  
SetActive  
SetClientHandles  
SetDataTypes
```

### 7.5.2.1 Item

#### Beschreibung

Diese Methode wird für Collection-Objekte benötigt.

#### Syntax

```
Item (ItemSpecifier As Variant) As OPCItem
```

##### ItemSpecifier

*ItemSpecifier* ist der mit 1 beginnende Index innerhalb der Collection und gibt ein Objekt der Klasse OPCItem über einen ItemSpecifier zurück.

## Bemerkungen

*Item* liefert eine Referenz auf das durch den Index *ItemSpecifier* beschriebene Item der Collection. Der ItemSpecifier ist der Index (beginnend mit 1) für die Collection. Mit *GetOPCItem* wird ein Item über ein ServerHandle referenziert.

Die Automation-Eigenschaft *Item* wird oft mit dem Objekt *OPCItem* verwechselt. Die Automation-Eigenschaft *Item* ist eine spezielle, reservierte Eigenschaft, die von Automation-Collections benutzt wird, um auf die enthaltenen Items zu verweisen. Das Objekt *OPCItem* ist ein OPC-Automation-spezifischer Objekttyp, der in einer Sammlung von OPC-Items enthalten sein kann.

### 7.5.2.2 GetOPCItem

#### Beschreibung

Diese Methode gibt das zu einem Server-Handle gehörende OPCItem zurück. Das Server-Handle wird von einer Add-Methode (*AddItem*, *AddItems*) geliefert. Mit der Eigenschaft *Item* wird nach Index referenziert.

#### Syntax

```
GetOPCItem (ServerHandle As Long) As OPCItem
```

##### ServerHandle

Dies ist das ServerHandle für Objekte der Klasse OPCItem. Mit *Item* wird über Index referenziert.

#### Beispiel

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(SomeItemServerHandle)
```

### 7.5.2.3 AddItem

#### Beschreibung

Diese Methode erzeugt ein neues Objekt der Klasse OPCItem und fügt es der Collection hinzu. Die Eigenschaften dieses neuen Objekts werden bestimmt von den aktuellen Voreinstellungen im Collection-Objekt OPCItems. Wurde ein Objekt der Klasse OPCItem hinzugefügt, können seine Eigenschaften verändert werden.

#### Syntax

```
AddItem (ItemID As String,
          ClientHandle As Long)
```

##### ItemID

*ItemID* ist die vollständige ItemID.

#### **ClientHandle**

Dies ist das ClientHandle der zugehörigen Gruppe.

#### **Bemerkungen**

Mit dieser Eigenschaft wird der Collection jeweils nur ein Item auf einmal hinzugefügt. Sollen gleichzeitig mehrere Items hinzugefügt werden, sollte dafür die Methode AddItems benutzt werden.

Informationen zu Fehlern und Ausnahmen finden Sie im Anhang zur Referenz Automation-Schnittstelle.

#### **Beispiel**

```
Dim AnOPCItemID as String
Dim AnClientHandle as Long
AnOPCItemID = "N7:0"
AnClientHandle = 1975
AnOPCItemCollection.AddItem AnOPCItemID AnClientHandle
```

#### **7.5.2.4 AddItems**

##### **Beschreibung**

Diese Methode erzeugt neue Objekte der Klasse OPCItem und fügt sie der Collection hinzu. Die Eigenschaften dieses neuen Objekts werden bestimmt von den aktuellen Voreinstellungen im Collection-Objekt OPCItems. Wurde ein Objekt der Klasse OPCItem hinzugefügt, können seine Eigenschaften verändert werden.

##### **Syntax**

```
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* gibt die Anzahl der betroffenen Items an.

##### **ItemIDs**

*ItemIDs* enthält ein Array mit vollständigen ItemIDs.

##### **ClientHandles**

*ClientHandles* ist ein Array der Client-Handles für die bearbeiteten Items.

##### **ServerHandles**

*ServerHandles* ist ein Array von ServerHandles der bearbeiteten Items.

**Errors**

*Errors* ist ein Array von Long-Werten, die den Erfolg jeder einzelnen Items-Operation anzeigen.

**RequestedDataTypes**

*RequestedDataTypes* ist eine optionale Variable vom Typ VARIANT, die einen Integer-Array für die nachgefragten Datentypen enthält.

**AccessPaths**

*AccessPaths* ist ein optionaler Parameter vom Typ VARIANT, der ein Array von Strings enthält.

**Bemerkungen**

Informationen zu Fehlern und Ausnahmen finden Sie im Anhang zur Referenz Automation-Schnittstelle.

**Beispiel**

```

Dim AddItemCount as long
Dim AnOPCItemIDs() as String
Dim AnOPCItemServerHandles as long
Dim AnOPCItemServerError as long
Dim AnOPCRequestedDataTypes as variant
Dim AnOPCAccessPaths as variant
For x = 1 To AddItemCount
    ClientHandles(x) = x + 1
    AnOPCItemID(x) = "Register_" & x
Next x
AnOPCItemCollection.AddItems AddItemCount,
                             AnOPCItemIDs,
                             ClientHandles,
                             AnOPCItemServerHandles,
                             AnOPCItemServerError,
                             AnOPCRequestedDataTypes,
                             AnOPCAccessPaths
' Code für die Fehlerbehandlung
' individuelle Fehler werden im "Errors"-Array aufgeführt

```

**7.5.2.5 Remove****Beschreibung**

Diese Methode entfernt ein Objekt der Klasse OPCItem.

**Syntax**

```

Remove (Count As Long,
        ServerHandles() As Long,
        ByRef Errors() As Long)

```

### Count

*Count* gibt die Anzahl der Items an, die entfernt werden sollen.

### ServerHandles

Dies ist ein Array von ServerHandles der betroffenen Items.

### Errors

*Errors* ist ein Array von Long-Werten, die den Erfolg jeder einzelnen Items-Operation anzeigen.

## Beispiel

```
AnOPCItemCollection.Remove AnOPCItemServerHandles,  
                           AnOPCItemServerErrors  
                           ' Code für die Fehlerbehandlung  
                           ' individuelle Fehler werden im "Errors"-Array aufgeführt
```

### 7.5.2.6 Validate

## Beschreibung

Diese Methode stellt fest, ob ein Objekt oder mehrere Objekte der Klasse OPCServer erfolgreich mittels einer *Add*-Methode (*AddItem*, *AddItems*) erzeugt werden konnte(n), fügt jedoch keine Objekte hinzu.

## Syntax

```
Validate (Count As Long,  
          ItemIDs() As String,  
          ByRef Errors() As Long,  
          Optional RequestedDataTypes As Variant,  
          Optional AccessPaths As Variant)
```

### Count

*Count* gibt die Anzahl der betroffenen Items an.

### ItemIDs

*ItemIDs* ist ein Array mit vollständigen ItemIDs.

### Errors

*Errors* ist ein Array von Long-Werten, die den Erfolg jeder einzelnen Items-Operation anzeigen.

### RequestedDataTypes

*RequestedDataTypes* ist ein Variant, der einen Integer-Array der nachgefragten Datentypen enthält.

### AccessPaths

Dies ist ein Variant, der ein Array von Strings des AccessPaths enthält.

## Bemerkungen

Informationen zu Fehlern und Ausnahmen finden Sie im Anhang zur Referenz Automation-Schnittstelle.

## Beispiel

```

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 für die Fehlerbehandlung
' individuelle Fehler werden im "Errors"-Array aufgeführt

```

### 7.5.2.7 SetActive

#### Beschreibung

Mit dieser Methode können einzelne Objekte der Klasse OPCItem in einer OPCItems-Collection aktiviert oder deaktiviert werden.

#### Syntax

```

SetActive (Count As Long,
           ServerHandles() As Long,
           ActiveState As Boolean,
           ByRef Errors() As Long)

```

##### Count

*Count* gibt die Anzahl der betroffenen Items an.

##### ServerHandles

*ServerHandles* ist ein Array von ServerHandles der betroffenen Items.

##### ActiveState

*ActiveState* gibt den Wert *True* aus, wenn das Item aktiv sein soll, *False*, wenn das Item inaktiv sein soll.

##### Errors

Array von Long-Werten, die den Erfolg jeder einzelnen Items-Operation anzeigen.

## Bemerkungen

Informationen zu Fehlern und Ausnahmen finden Sie im Anhang zur Referenz Automation-Schnittstelle.

## Beispiel

```
' Items aktivieren (entsprechender
' Eigenschaft den Wert TRUE zuweisen)
AnOPCItemCollection.SetActiveItemCount,
                    AnOPCItemServerHandles,
                    TRUE,
                    AnOPCItemServerErrorErrors

' Code für die Fehlerbehandlung
' individuelle Fehler werden im "Errors"-Array aufgeführt
```

### 7.5.2.8 SetClientHandles

#### Beschreibung

Ändert die ClientHandles eines bzw. mehrerer Items einer Gruppe.

#### Syntax

```
SetClientHandles (Count As Long,
                  ServerHandles() As Long,
                  ClientHandles() (As Long,
                  ByRef Errors() As Long)
```

##### Count

*Count* gibt die Anzahl der betroffenen Items an.

##### ServerHandles

*ServerHandles* ist ein Array von ServerHandles der bearbeiteten Items.

##### ClientHandles

*ClientHandles* ist ein Array neuer Client-Handles, die gespeichert werden sollen. Die Client-Handles müssen nicht eindeutig sein.

##### Errors

*Errors* ist ein Array von Long-Werten, die den Erfolg jeder einzelnen Items-Operation anzeigen.

## Beispiel

```
For x = 1 To ItemCount
    ClientHandles(x) = x + 1975
Next x
AnOPCItemCollection.SetClientHandles ItemCount,
                                    AnOPCItemServerHandles,
                                    ClientHandles,
                                    AnOPCItemServerErrorErrors
```

### 7.5.2.9 SetDataTypes

#### Beschreibung

Mit dieser Methode wird der gewünschte Datentyp für ein oder mehrere Items geändert.

#### Syntax

```
SetDataTypes (Count As Long,  
             ServerHandles() As Long,  
             RequestedDataTypes() As Long,  
             ByRef Errors() As Long)
```

##### Count

*Count* gibt die Anzahl der betroffenen Items an.

##### ServerHandles

*ServerHandles* ist ein Array von ServerHandles der bearbeiteten Items.

##### RequestedDataTypes

*RequestedDataTypes* ist ein Array, der die neu festgelegten Datentypen enthält, die gespeichert werden sollen.

##### Errors

*Errors* ist ein Array von Long-Werten, die den Erfolg jeder einzelnen Items-Operation anzeigen.

#### Bemerkungen

Informationen zu Fehlern und Ausnahmen finden Sie im Anhang zur Referenz Automation-Schnittstelle.

#### Beispiel

```
Dim RequestedDataTypes(100) As Long  
For x = 1 To ItemCount  
    RequestedDataTypes (x) = "ein VBInteger"  
Next x  
AnOPCItemCollection.SetDataTypes ItemCount,  
                                  AnOPCItemServerHandles,  
                                  RequestedDataTypes,  
                                  AnOPCItemServerErrors
```

## 7.6

## Das Objekt OPCItem

Ein Objekt der Klasse OPCItem stellt eine Verbindung zu Datenquellen innerhalb des Servers dar. Zu jeden Item gehören ein Wert, eine Qualität und ein Zeitstempel. Der Wert wird als Variable vom Typ VARIANT angegeben und besitzt eine Qualität.

## 7.6.1 Eigenschaften des Objekts OPCItem

Parent  
ClientHandle  
ServerHandle  
AccessPath  
AccessRights  
ItemID  
IsActive  
RequestedDataType  
Value  
Quality  
TimeStamp  
CanonicalDataType  
EUType  
EUInfo

### 7.6.1.1 Parent

#### Beschreibung

(Nur lesbar) Diese Eigenschaft gibt die Referenz an das übergeordnete OPCGroup-Objekt zurück.

#### Syntax

```
Parent As OPCGroup
```

### 7.6.1.2 ClientHandle

#### Beschreibung

(Les- und schreibbar) Diese Eigenschaft liefert einen Wert des Typs Long, der zum OPCItem gehört. Mit Hilfe dieser Eigenschaft erkennt der Client das Ziel der Daten. Ein ClientHandle ist typischerweise ein Index o. ä. Es wird bei Daten- oder Zustandsänderungen zusammen mit Ereignissen des Objekts OPCGroup an den Client zurückgegeben.

#### Syntax

```
ClientHandle As Long
```

## Beispiel

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(EinItemServerHandle)

    ' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Long
Dim SomeValue As Long
CurrentValue = AnOPCItem.ClientHandle

    ' Beispiel VB-Syntax (Festlegen der Eigenschaft):
AnOPCItem.ClientHandle = SomeValue
```

### 7.6.1.3 ServerHandle

#### Beschreibung

(Nur lesbar) Dies ist das Handle, das der Server dem OPCItem zugewiesen hat; ein OPCItem kann damit eindeutig identifiziert werden. Der Client muss dieses Handle einigen Methoden zugänglich machen, die mit Objekten der Klasse OPCItem arbeiten (z. B. *OPCItems.Remove*).

#### Syntax

```
ServerHandle As Long
```

## Beispiel

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(EinItemServerHandle)

    ' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Long
Dim SomeValue As Long
CurrentValue = AnOPCItem.ServerHandle
```

### 7.6.1.4 AccessPath

#### Beschreibung

(Nur lesbar) *AccessPath* liefert den Zugangspfad, den der Client mittels eines *Add*-Aufrufs (*AddItem*, *AddItems*) spezifiziert hat.

#### Syntax

```
AccessPath As String
```

## Beispiel

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(EinItemServerHandle)

' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As String
Dim SomeValue As String
CurrentValue = AnOPCItem.AccessPath
```

### 7.6.1.5 AccessRights

#### Beschreibung

(Nur lesbar) *AccessRights* gibt die Zugangsrechte des Items aus.

#### Syntax

```
AccessRights As Long
```

#### Bemerkungen

Diese Eigenschaft zeigt an, ob für das entsprechende Item nur Lese-, nur Schreib- oder sowohl Lese- als auch Schreibrechte bestehen.

## Beispiel

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(EinItemServerHandle)

' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Long
Dim SomeValue As Long
CurrentValue = AnOPCItem.AccessRights
```

### 7.6.1.6 ItemID

#### Beschreibung

(Nur lesbar) *ItemID* ist die eindeutige Identifikation des Items.

#### Syntax

```
ItemID As String
```

## Beispiel

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(EinItemServerHandle)

' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As String
Dim SomeValue As String
CurrentValue = AnOPCItem.ItemID
```

### 7.6.1.7 IsActive

#### Beschreibung

(Les- und schreibbar) Diese Eigenschaft legt fest, ob für dieses Item Benachrichtigungssereignisse generiert werden sollen.

#### Syntax

```
IsActive As Boolean
```

#### Bemerkungen

Ist das Item aktiv, wird der Wert *True* ausgegeben; ist das Item inaktiv, wird der Wert *False* ausgegeben.

## Beispiel

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(EinItemServerHandle)

' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Boolean
Dim SomeValue As Boolean
CurrentValue = AnOPCItem.IsActive

' Beispiel VB-Syntax (Festlegen der Eigenschaft):
AnOPCItem.IsActive = SomeValue
```

### 7.6.1.8 RequestedDataType

#### Beschreibung

(Les- und schreibbar) *RequestedDataType* ist der Datentyp, in dem der Wert des Items ausgegeben wird.

Hinweis: Wurde der angeforderte Datentyp abgewiesen, wird das Item ungültig, bis der angeforderte Datentyp einen gültigen Wert annimmt.

## Syntax

```
RequestedDataType As Integer
```

## Bemerkungen

Informationen zu Fehlern und Ausnahmen finden Sie im Anhang zur Referenz Automation-Schnittstelle.

## Beispiel

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(EinItemServerHandle)

' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Integer
Dim SomeValue As Integer
CurrentValue = AnOPCItem.RequestedDataType

' Beispiel VB-Syntax (Festlegen der Eigenschaft):
AnOPCItem.RequestedDataType = SomeValue
```

### 7.6.1.9 Value

## Beschreibung

(Nur lesbar) Diese Eigenschaft gibt den zuletzt vom Server gelesenen Wert aus. Dies ist die voreingestellte Eigenschaft von *OPCItem*.

## Syntax

```
Value As Variant
```

## Beispiel

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(EinItemServerHandle)

' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Variant
Dim SomeValue As Variant
CurrentValue = AnOPCItem.Value
```

### 7.6.1.10 Quality

#### Beschreibung

(Nur lesbar) Diese Eigenschaft gibt den zuletzt vom Server gelesenen Wert für die Qualität aus.

#### Syntax

```
Quality As Long
```

#### Beispiel

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(EinItemServerHandle)

' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Long
Dim SomeValue As Long
CurrentValue = AnOPCItem.Quality
```

### 7.6.1.11 TimeStamp

#### Beschreibung

(Nur lesbar) Diese Eigenschaft gibt den zuletzt vom Server gelesenen Wert für den Zeitstempel aus.

#### Syntax

```
TimeStamp As Date
```

#### Beispiel

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(EinItemServerHandle)

' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Date
Dim SomeValue As Date
CurrentValue = AnOPCItem.TimeStamp
```

### 7.6.1.12 CanonicalDataType

#### Beschreibung

(Nur lesbar) Diese Eigenschaft gibt den ursprünglichen Datentyp des Items aus. Dieser Wert kann sich von dem angeforderten Datentyp (RequestedDataType) unterscheiden.

#### Syntax

```
CanonicalDataType As Integer
```

#### Bemerkungen

Informationen zu Fehlern und Ausnahmen finden Sie im Anhang zur Referenz Automation-Schnittstelle.

#### Beispiel

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(EinItemServerHandle)

' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Integer
Dim SomeValue As Integer
CurrentValue = AnOPCItem.CanonicalDataType
```

### 7.6.1.13 EUType

#### Beschreibung

(Nur lesbar) Die Eigenschaft *EUType* gibt an, welche Art von EU-Information in der Eigenschaft EUInfo enthalten ist (EU=Engineering Unit).

#### Syntax

```
EUType As Integer
```

## Bemerkungen

0 bedeutet, dass keine EU-Information erhältlich ist (die Eigenschaft EUInfo hat den Wert VT\_EMPTY).

1 bedeutet *Analog*; die EU-Information enthält ein SAFEARRAY von genau zwei Varianten vom Typ double (VT\_ARRAY | VT\_R8), die dem Minimal- und Maximalwert entsprechen.

2 bedeutet *Enumeration*; die EU-Information enthält ein SAFEARRAY von Strings (VT\_ARRAY | VT\_BSTR), der eine Liste von Strings enthält (z. B. *OPEN*, *CLOSE*, *IN TRANSIT* usw.), die aufeinanderfolgenden Zahlenwerten entsprechen (0, 1, 2 usw.).

Weitere Informationen finden Sie in der Spezifikation *OPC Data Access Custom Interface* unter *OPCItem Attributes*.

## Beispiel

```
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(EinItemServerHandle)

' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Integer
Dim SomeValue As Integer
CurrentValue = AnOPCItem.EUType
```

### 7.6.1.14 EUInfo

#### Beschreibung

(Nur lesbar) *EUInfo* enthält eine Variable mit Informationen über die Einheit (EU=Engineering Unit).

#### Syntax

```
EUInfo As Variant
```

#### Bemerkungen

Weitere Informationen finden Sie in der Spezifikation *OPC Data Access Custom Interface* unter *OPCItem Attributes*.

## Beispiel

```

Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(EinItemServerHandle)

' Beispiel VB-Syntax (Ermitteln der Eigenschaft):
Dim CurrentValue As Variant
Dim SomeValue As Variant
CurrentValue = AnOPCItem.EUInfo

```

## 7.6.2 Methoden des Objekts OPCItem

Read  
Write

### 7.6.2.1 Read

#### Beschreibung

Mit dieser Methode kann ein Item vom Server gelesen werden. *Read* kann mit nur einer Quelle aufgerufen werden (entweder OPCCache oder OPCDevice), um Werte, Qualität und Zeitstempel des Items zu aktualisieren. Müssen Wert, Qualität und Zeitstempel synchron sein, liefern die optionalen Parameter dieser Methode Werte, die gleichzeitig aktualisiert worden sind.

#### Syntax

```
Read (Source As Integer,
      Optional ByRef Value As Variant,
      Optional ByRef Quality As Variant,
      Optional ByRef TimeStamp As Variant)
```

##### Source

*Source* entspricht der Datenquelle, also OPC\_DS\_CACHE oder OPC\_DS\_DEVICE

##### Value

*Value* gibt den zuletzt vom Server gelesenen Wert aus.

##### Quality

*Quality* gibt den zuletzt vom Server gelesenen Wert für die Qualität aus.

##### TimeStamp

*TimeStamp* gibt den zuletzt vom Server gelesenen Zeitstempel aus.

## Beispiel

```
Private Sub ReadButton_Click()
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(EinItemServerHandle)
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
' Verarbeiten der Werte
TextBox.Text = Value
End Sub
```

### 7.6.2.2 Write

#### Beschreibung

Mit dieser Methode kann im OPCServer ein Wert geschrieben werden.

#### Syntax

```
Write (Value As Variant)
```

##### Value

*Value* enthält den Wert, der geschrieben werden soll.

## Beispiel

```
Private Sub WriteButton_Click()
Dim AnOPCItem as OPCItem
Set OPCItem = GetOPCItem(EinItemServerHandle)
Dim Value As Variant
Value = 1975
AnOPCItem.Write Value
End Sub
```

## 7.7

## Definitionen

Hier finden Sie Konstantendefinitionen zum Zustand des Servers und Erklärungen zu Fehlermeldungen.

## 7.7.1 Zustand des Servers

### OPCRunning

Der Server arbeitet ordnungsgemäß; dies ist der normale Serverzustand.

### OPCFailed

Beim Server ist ein Hersteller spezifischer Fehler aufgetreten; der Server arbeitet nicht mehr ordnungsgemäß. Die Fehlerbehebung ist in diesem Fall Hersteller spezifisch. Normalerweise wird der Fehlercode *E\_FAILURE* von einer der anderen Server-Methoden ausgegeben.

### OPCNoconfig

Der Server arbeitet zwar, hat jedoch keine Informationen über die Konfiguration geladen, d. h., der Server arbeitet nicht ordnungsgemäß.

Server, die keine derartigen Informationen benötigen, geben diese Zustandsmeldung nicht aus.

### OPCSuspended

Der Betrieb des Servers wurde vorübergehend von einer Hersteller spezifischen Methode unterbrochen, d. h., der Server sendet und empfängt keine Daten. Angaben zur Qualität lauten: OPC\_QUALITY\_OUT\_OF\_SERVICE.

### OPCTest

Der Server befindet sich im Testmodus. Die Datenausgabe ist von der Hardware getrennt, der Server verhält sich ansonsten jedoch normal. Je nach Hersteller-Implementierung beruht der Daten-Input auf Werten, die entweder tatsächlich vorhanden sind oder simuliert werden. Angaben zur Qualität werden normal ausgegeben.

### OPCDisconnected

Das Automation-Server-Objekt ist nicht mit einem OPC-Custom-Interface-Server verbunden.

## 7.7.2 Fehlermeldungen

### OPCInvalidHandle

#### 0xC0040001L

Der Wert dieses Handles ist ungültig. Ein Client darf kein ungültiges Handle an einen Server weitergeben. Tritt dieser Fehler auf, liegt ein Programmierfehler beim Client oder ggf. beim Server vor.

### OPCBadType

**0xC0040004L**

Der Server kann die Daten nicht zwischen dem gewünschten Datentyp und dem ursprünglichen Datentyp konvertieren.

### OPCPublic

**0xC0040005L**

Die gewünschte Operation kann nicht für eine als *public* deklarierte Gruppe durchgeführt werden.

### OPCBadRights

**0xC0040006L**

Die Zugriffsrechte des Items lassen die gewünschte Operation nicht zu.

### OPCUknownItemID

**0xC0040007L**

Die ItemID ist im Adressraum des Servers nicht definiert oder existiert dort nicht mehr (für Lese- und Schreiboperationen).

### OPCIinvalidItemID

**0xC0040008L**

Die ItemID stimmt nicht mit der Syntax des Servers überein.

### OPCIinvalidFilter

**0xC0040009L**

Der String des Filters ist ungültig.

### OPCUknownPath

**0xC004000AL**

Dem Server ist der Zugriffspfad des Items unbekannt.

### OPCRage

**0xC004000BL**

Der Wert liegt außerhalb des gültigen Bereichs.

### OPCDuplicateName

**0xC004000CL**

*DuplicateName* ist ungültig.

**OPCUncorrectedRate****0x0004000DL**

Der Server unterstützt die gewünschte Datenrate nicht, verwendet jedoch eine möglichst ähnliche.

**OPCClamp****0x0004000EL**

Ein Write-Wert wurde zwar akzeptiert, jedoch nicht ausgegeben.

**OPCInuse****0x0004000FL**

Die Operation kann nicht ausgeführt werden, weil noch eine Referenz auf das Objekt besteht.

**OPCInvalidConfig****0xC0040010L**

Das Dateiformat der Serverkonfiguration ist ungültig.

**OPCNotFound****0xC0040011L**

Das gewünschte Objekt (z. B. eine *PublicGroup*) konnte nicht gefunden werden.

**OPCInvalidPID****0xC0040203L**

Die weitergegebene PropertyID des Items ist ungültig.

## 7.8 Anhang zur Referenz Automation-Schnittstelle

**Fehler**

Tritt ein Laufzeitfehler auf, wird der Fehler mittels des Objektes *Err* von Visual Basic eindeutig identifiziert.

Behandelt VB Fehler nicht mit dem Mechanismus *On Err*, wird eine Ausnahme erzeugt; je nach Kontext (Visual Basic im Debug-Modus oder einem ausführbaren Programm) erscheint eine Messagebox mit folgenden Informationen:

Runtime Error: Fehlercode in Dezimalzahlen (in Hexadezimalzeichen)

*Method X of Object Y Failed* (Wenn Sie mit einer ausführbaren Anwendung arbeiten, werden für *X* und *Y* keine Werte angegeben).

Daher empfiehlt die OPC Foundation dringend, dass die Anwendung geeignete Maßnahmen trifft, um OPC-Automation-Fehler abzufangen, die beim Festlegen von Eigenschaften oder

beim Ausführen von Methoden auftreten können.

in *Error-Handler* ist eine Routine, die Fehler in der Anwendung findet und darauf reagiert. Ein OPC-Automation-Client sollte für jede Anwendungsfunktionalität ErrorHandler bereitstellen, wenn eine Eigenschaften festgelegt oder eine Methode aufgerufen wird.

Ein Error-Handler besteht aus folgenden drei Teilen:

1. Erstellen oder Aktivieren einer Verzweigung für den Fehlerfall, die festlegt, wohin die Anwendung springen soll (welche Routine zur Fehlerbehandlung ausgeführt werden soll), wenn ein Fehler auftritt. Das Statement in *On Error* aktiviert die Verzweigung und führt die Anwendung an das Label, das den Anfang der Routine kennzeichnet.
2. Schreiben einer Error-Handling-Routine, die Fehler behandelt, die aus dem Festlegen von Eigenschaften oder Anwenden von Methoden resultieren.
3. Verlassen der Routine.

Es muss Klarheit darüber herrschen, welche Maßnahmen die Anwendung ergreifen soll, wenn ein Fehler auftritt. Soll beispielsweise eine Gruppe hinzugefügt werden, deren Namen (vom Anwender vorgegeben) nicht eindeutig ist, kann der Anwender darüber in Kenntnis gesetzt werden, dass die Gruppe nicht hinzugefügt wurde; außerdem sollte der Anwender aufgefordert werden, einen anderen Namen einzugeben. Die Anwendung kann aber auch erneut versuchen, die Gruppe (mit " ") hinzuzufügen, wobei der Name übernommen wird, den der Server generiert.

## Fehler finden

Eine Verweisung für den Fehlerfall wird aktiviert, wenn VB ein On-Error-Statement ausführt, das einen Error-Handler spezifiziert. Die Verzweigung bleibt aktiv, solange die Prozedur, die diese Verzweigung enthält, selbst aktiv ist, d. h., bis *Exit Sub*, *Exit Function*, *Exit Property*, *End Sub*, *End Function* bzw. *End Property* durchlaufen wird.

Um eine Verzweigung zu erstellen, die zu einer Error-Handling-Routine springt, wird das Statement *On Error Go To Line* verwendet, wobei *Line* die Markierung ist, die den Code der Fehlerbehandlung identifiziert.

## Fehler behandeln

In der Routine zur Behandlung von Fehlern wird zuerst eine Markierung an den Anfang gesetzt. Die Markierung sollte einen beschreibenden Namen haben und muss mit einem Doppelpunkt enden.

Der Hauptteil enthält den Code zur Fehlerbehandlung, entweder in Form von *If(...Then...)Else...-*-Schleifen oder als *Case*-Statement. Dort werden die Maßnahmen gegen die am häufigsten auftretenden Fehler aufgeführt.

Die Eigenschaft *Number* des *Err*-Objekts enthält einen nummerischen Code, der die Laufzeitfehler repräsentiert, die in der letzten Zeit am häufigsten aufgetreten sind. Die Fehlernummer dieser Eigenschaft enthält den Wert, mit dem *GetErrorString* aufgerufen wird, um die Fehlernummer in einen lesbaren String umzuwandeln.

## Beispiel

```
Dim AnOpcServer As OPCServer
Private Sub Command1_Click()
On Error GoTo testerror
Set AnOpcServer = New OPCServer
' falls "fuzz" nicht existiert und deshalb die Verbindung
' nicht aufgebaut werden kann, wird zum Label "testerror"
' verzweigt
AnOpcServer.Connect ("fuzz")
Time = AnOpcServer.CurrentTime
Debug.Print Time
testerror:
Debug.Print Err.Number
End Sub
```



# Literaturhinweise

Sie möchten die Spezifikationen der OPC Foundation lesen oder Sie brauchen noch zusätzliche Informationen zu einem bestimmten Thema.

Thema	wird behandelt in:
Sie wollen sich über ein bestimmtes Thema noch genauer informieren?	Weiterführende Literatur
Sie wollen detaillierte Informationen über OPC Data Access und OPC Alarms & Events?	OPC-Spezifikationen

## Auffinden der SIMATIC NET-Dokumentation

- **Kataloge**

Die Bestellnummern für die hier relevanten Siemens-Produkte finden Sie in den folgenden Katalogen:

- SIMATIC NET Industrielle Kommunikation / Industrielle Identifikation, Katalog IK PI
- SIMATIC Produkte für Totally Integrated Automation und Micro Automation, Katalog ST 70

Die Kataloge sowie zusätzliche Informationen können Sie bei Ihrer Siemens-Vertretung anfordern.

Die Industry Mall finden Sie unter folgender Adresse im Internet:

<https://mall.industry.siemens.com>

- **Dokumentation im Internet**

Die SIMATIC NET-Handbücher finden Sie auf den Internet-Seiten des Siemens Automation Customer Support:

<http://support.automation.siemens.com>

Navigieren Sie zur gewünschten Produktgruppe und nehmen Sie folgende Einstellungen vor:

Register "Beitragsliste", Beitragstyp "Handbücher / Betriebsanleitungen"

- **Dokumentation in der STEP 7-Installation**

Handbücher, die in der Online-Dokumentation der STEP 7-Installation auf Ihrem PG/PC vorhanden sind, finden Sie über das Startmenü ("Start" > "Alle Programme" > "Siemens Automation" > "Dokumentation").

## Siehe auch

SIMATIC NET-Handbücher:

(<http://support.automation.siemens.com/WW/view/de/10805878>)

SIMATIC-Dokumentation: (<http://www.siemens.de/simatic-doku>)

SIMATIC NET

PROFIBUS Netzhandbuch

Siemens AG

(SIMATIC NET Manual Collection)

Im Internet unter folgender Beitrags-ID: 35222591

(<http://support.automation.siemens.com/WW/view/de/35222591>)

SIMATIC NET

Industrial Ethernet Netzhandbuch

Siemens AG

(SIMATIC NET Manual Collection)

Im Internet unter folgender Beitrags-ID: 27069465

(<http://support.automation.siemens.com/WW/view/de/27069465>)

SIMATIC NET

Programmierschnittstelle DP-Base für CP 5613/CP 5614

Siemens AG

(SIMATIC NET Manual Collection)

Im Internet unter folgender Beitrags-ID: 1653531

(<http://support.automation.siemens.com/WW/view/de/1653531>)

SIMATIC NET

IO-Base-Anwenderschnittstelle

Siemens AG

(SIMATIC NET Manual Collection)

Im Internet unter folgender Beitrags-ID: 19779901

(<http://support.automation.siemens.com/WW/view/de/19779901>)

## 8.1 OPC-Spezifikationen

### Welche OPC-Spezifikationen gibt es?

Die erste und grundlegendste Spezifikation der OPC Foundation war die OPC Data Access-Spezifikation. Sie bietet die Verwaltung von Prozessvariablen an und verschiedene Möglichkeiten, auf diese Variablen zuzugreifen.

Die OPC Alarms & Events-Spezifikation legt die Übertragung von Prozessalarmen und Ereignissen fest. Sie ist unabhängig von der Data Access-Spezifikation.

Die OPC Foundation hat bisher folgende Spezifikationen veröffentlicht, die auch auf der SIMATIC NET Manual Collection enthalten sind:

- **OPC Overview**

**Version 1.0, October 27, 1998**

Eine kurze Einführung in die grundlegenden Konzepte von OPC

- **OPC Common Definition and Interfaces**

**Version 1.0, October 27, 1998**

Dieses Dokument beschreibt wichtige Schnittstellen, die bei allen OPC-Servern vorhanden sind.

- **Data Access Custom Interface**

**Version 1.1, September 11, 1997**

Die Spezifikation der Custom-Schnittstelle für Data Access, Version 1.1

- **Data Access Custom Interface**

**Version 2.04, September 5, 2000**

Die Spezifikation der Custom-Schnittstelle für Data Access, Version 2.04

- **Data Access Custom Interface**

**Version 2.05, December 17, 2001**

Die Spezifikation der Custom-Schnittstelle für Data Access, Version 2.05

- **Data Access Custom Interface**

**Version 3.00, March 4, 2003**

Die Spezifikation der Custom-Schnittstelle für Data Access, Version 3.00

- **Data Access Automation Interface**

**Version 2.02, February 4, 1999**

Die Spezifikation der Automation-Schnittstelle für Data Access, Version 2.02

- **OPC Alarms and Events Custom Interface**

**Version 1.10, October 2, 2002**

Eine Beschreibung des OPC Alarms & Events-Servers sowie die Spezifikation der Custom-Schnittstelle dieses Servers

- **Alarm & Events Automation Interface**

**Version 1.01, December 15, 1999**

Die Spezifikation der Automation-Schnittstelle des OPC Alarms & Events-Servers

- **OPC XML-DA Specification**

**Version 1.01, December 21, 2004**

Die Spezifikation der OPC-XML-Schnittstelle für Data Access

- **OPC Unified Architecture**

**Version 1.0 or later**

<http://www.opcfoundation.org/UA/>

Part 1 - Concepts

Part 2 - Security Model

Part 3 - Address Space Model

Part 4 - Services

Part 5 - Information Model

Part 6 - Service Mappings

Part 7 - Profiles

Part 8 - Data Access

Part 9 - Alarms and Conditions

Part 10 - Programs

Part 11 - Historical Access

Part 12 - Discovery

Die Spezifikation von OPC UA.

Weitere Spezifikationen für Aufgaben der Automatisierungstechnik werden folgen.