

SIMATC

TIA Portal Openness: Projekterstellung automatisieren

Systemhandbuch

Sicherheitshinweis

1

Liesmich zu
TIA Portal Openness

2

Was ist neu in TIA Portal
Openness?

3

Grundlagen

4

Einführung

5

Konfigurationen

6

TIA Portal Openness API

7

Export/Import

8

Wesentliche Änderungen

9

Ausdruck der Online-Hilfe

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

1	Sicherheitshinweis	17
2	Liesmich zu TIA Portal Openness	19
2.1	Liesmich	19
2.2	Größere Änderungen in TIA Portal Openness V16.....	22
2.3	Bekanntgabe größerer Änderungen in künftigen Releases	24
2.4	Hinweise zum Schreiben von langfristig stabilem Code	25
3	Was ist neu in TIA Portal Openness?	27
4	Grundlagen	29
4.1	Voraussetzungen für TIA Portal Openness.....	29
4.2	Installation	30
4.2.1	TIA Portal Openness installieren.....	30
4.2.2	Benutzer der Benutzergruppe "Siemens TIA Openness" hinzufügen.....	31
4.2.3	Auf das TIA Portal zugreifen	37
4.3	Aufgaben von Openness.....	38
4.3.1	Einsatzmöglichkeiten	38
4.3.2	Export/Import.....	39
4.4	Objektliste	39
4.5	Attribute von Bausteinen, DBs und UDTs festlegen	43
4.6	Standard-Bibliotheken.....	45
4.7	Anmerkungen zur Leistung von TIA Portal Openness	45
5	Einführung	47
6	Konfigurationen	49
7	TIA Portal Openness API	53
7.1	Einleitung	53
7.2	Programmierschritte	53
7.3	TIA Portal Openness-Objektmodell.....	54
7.4	Bausteine und Typen des TIA Portal Openness-Objektmodells	61
7.5	Hierarchie von Hardware-Objekten des Objektmodells	69
7.6	Informationen über installierte TIA Portal Openness-Versionen	71
7.7	Anwendungsbeispiel: API-Zugriff in einer Windows-Forms Applikation erstellen	72
7.8	Einsatz der Codebeispiele	77
7.9	Allgemeine Funktionen.....	79
7.9.1	Unterstützung von IntelliSense durch TIA Portal Openness	79

7.9.2	Verbindung zum TIA Portal aufbauen	79
7.9.3	TIA Portal Openness-Firewall	84
7.9.4	Event-Handler	85
7.9.5	Dialoge mit Systemmeldungen programmgesteuert bestätigen	87
7.9.6	Verbindung zum TIA Portal beenden	88
7.9.7	Diagnoseschnittstellen im TIA Portal	89
7.9.8	Ausschließlicher Zugriff	95
7.9.9	Dynamisches Verhalten	97
7.9.10	Arbeiten mit Assoziationen	100
7.9.11	Mit Zusammensetzungen arbeiten	100
7.9.12	Objektgleichheit prüfen	102
7.9.13	Lese-Operationen für Attribute	103
7.9.14	Transaktionsbehandlung	104
7.9.15	Ein Objekt DirectoryInfo/FileInfo erstellen	107
7.9.16	Unterstützung der Selbstbeschreibung für Attribute, Navigatoren, Aktionen und Dienste....	107
7.10	Funktionen der Projekte und Projektdaten	111
7.10.1	Projekt öffnen	111
7.10.2	Projekt anlegen	115
7.10.3	Auf allgemeine Einstellungen des TIA Portals zugreifen	116
7.10.4	Projekt archivieren und abrufen	120
7.10.5	Schreibgeschütztes TIA Portal-Projekt öffnen	124
7.10.6	Auf Sprachen zugreifen	125
7.10.7	Objektstruktur und -Attribute ermitteln	127
7.10.8	Auf Software-Ziel zugreifen	129
7.10.9	Auf mehrsprachige Texte zugreifen und sie enumerieren	130
7.10.10	Aktualisierung der Projekteigenschaft "Simulationsunterstützung"	131
7.10.11	Projektbezogene Attribute lesen	132
7.10.12	Projektgrafik löschen	135
7.10.13	Projekt übersetzen	135
7.10.14	Projekt speichern	137
7.10.15	Projekt schließen	139
7.11	Funktionen für Verbindungen	139
7.11.1	Konfigurierbare Attribute einer Port-zu-Port-Verbindung	139
7.12	Funktionen in Bibliotheken	142
7.12.1	Funktionen auf Objekte und Instanzen	142
7.12.2	Auf globale Bibliotheken zugreifen	143
7.12.3	Zugriff auf Sprachen der globalen Bibliothek	145
7.12.4	Bibliotheken öffnen	147
7.12.5	Offene Bibliotheken enumerieren	149
7.12.6	Bibliotheken speichern und schließen	149
7.12.7	Bibliothek archivieren und abrufen	151
7.12.8	Globale Bibliotheken erstellen	154
7.12.9	Auf Ordner in einer Bibliothek zugreifen	155
7.12.10	Auf Typen zugreifen	158
7.12.11	Auf Typ-Versionen zugreifen	160
7.12.12	Zugriff auf Bausteine in Bibliotheken	165
7.12.13	Auf Instanzen zugreifen	167
7.12.14	Auf Kopiervorlagen zugreifen	169
7.12.15	Masterkopie aus einem Projekt in Bibliothek erzeugen	171
7.12.16	Ein Objekt aus einer Masterkopie erstellen	173
7.12.17	Masterkopien kopieren	175

7.12.18	Ermitteln veralteter Typinstanzen.....	176
7.12.19	Projekt aktualisieren.....	179
7.12.20	Bibliothek aktualisieren	181
7.12.21	Bibliotheksinhalte löschen.....	182
7.13	Funktionen für das Aufrufen von Geräten, Netzwerken und Verbindungen.....	184
7.13.1	Editor "Geräte & Netze" öffnen	184
7.13.2	PLC-Target und HMI-Target abfragen	185
7.13.3	Auf Attribute eines Adressobjekts zugreifen	187
7.13.4	Aufrufen eines Modulkanals	190
7.14	Funktionen in Netzwerken.....	191
7.14.1	Subnetz anlegen	191
7.14.2	Auf Subnetze zugreifen	192
7.14.3	Auf interne Subnetze zugreifen	194
7.14.4	Typkennung von Subnetzen abrufen	194
7.14.5	Attribute eines Subnetzes aufrufen	195
7.14.6	Globales Subnetz löschen	201
7.14.7	Alle Beteiligten eines Subnetzes enumerieren.....	202
7.14.8	IO-Systeme eines Subnetzes enumerieren	202
7.14.9	Auf Teilnehmer zugreifen	203
7.14.10	Attribute eines Teilnehmers aufrufen	204
7.14.11	Teilnehmer mit einem Subnetz verbinden.....	208
7.14.12	Teilnehmer von einem Subnetz trennen	208
7.14.13	IO-System erstellen.....	209
7.14.14	Attribute eines IO-Systems aufrufen	210
7.14.15	IO-Connector mit einem IO-System verbinden	210
7.14.16	Mastersystem oder IO-System einer Schnittstelle abrufen	211
7.14.17	IO-Controller abrufen	212
7.14.18	IO-Connector abrufen	213
7.14.19	IO-Connector von einem IO-System oder einem DP-Mastersystem trennen	214
7.14.20	Attribute eines DP-Mastersystems aufrufen.....	214
7.14.21	Attribute eines PROFINET IO-Systems aufrufen	215
7.14.22	DP-Mastersystem löschen	217
7.14.23	Profinet IO-System löschen	217
7.14.24	DP-Mastersystem erstellen	218
7.14.25	Port-Verschaltungsinformationen des Port-Geräteelements aufrufen	219
7.14.26	Attribute der Port-Verschaltung	220
7.14.27	Attribute eines Ports aufrufen.....	223
7.14.28	DP-Mastersysteme eines Subnetzes enumerieren.....	224
7.14.29	Zugeordnete IO-Connectors enumerieren	225
7.14.30	DP-IO-Connector mit einem DP-Mastersystem verbinden	225
7.14.31	Zugriff auf AS-i-Profil und Parameterattribute für virtuelle Slaves.....	226
7.15	Funktionen auf Geräten	228
7.15.1	Obligatorische Attribute von Geräten	228
7.15.2	Typkennung von Geräten und Geräteelementen abrufen.....	229
7.15.3	App-ID in Gerät und Geräteelementen einstellen	232
7.15.4	App-ID in Gerät und Geräteelementen abrufen	233
7.15.5	App-ID in Gerät und Geräteelementen entfernen	234
7.15.6	Erstellen eines Geräts	235
7.15.7	Geräte enumerieren	236
7.15.8	Auf Geräte zugreifen	239
7.15.9	Löschen eines Geräts	241

7.16	Funktionen auf Geräteelementen.....	242
7.16.1	Obligatorische Attribute von Geräteelementen	242
7.16.2	Ein Geräteelement erstellen und stecken	244
7.16.3	Geräteelemente in einen anderen Steckplatz verschieben.....	248
7.16.4	Geräteelement kopieren.....	249
7.16.5	Geräteelement löschen	250
7.16.6	Geräteelemente enumerieren	250
7.16.7	Geräteelemente aufrufen	252
7.16.8	Auf das Geräteelement als Schnittstelle zugreifen	256
7.16.9	Auf Attribute einer I/O-Geräteschnittstelle zugreifen.....	257
7.16.10	Auf Attribute des IoController zugreifen	259
7.16.11	Auf Attribute des IoConnector zugreifen	260
7.16.12	Auf einen Adresscontroller zugreifen	262
7.16.13	Auf Adressen zugreifen.....	263
7.16.14	Auf "Hardwarekennung" zugreifen	265
7.16.15	Auf Hardwarekennungscontroller zugreifen	266
7.16.16	Auf Kanäle von Geräteelementen zugreifen	267
7.16.17	PSC-Datei erstellen und exportieren.....	268
7.16.18	Verwalten von Verbindungen bei Erweiterungsbaugruppenträgern.....	269
7.17	Funktionen auf Daten eines HMI-Gerätes.....	270
7.17.1	Bilder	270
7.17.1.1	Benutzerdefinierte Bilderordner erzeugen	270
7.17.1.2	Bild aus einem Ordner löschen	271
7.17.1.3	Bildvorlage aus einem Ordner löschen	272
7.17.1.4	Alle Bilder aus einem Ordner löschen.....	272
7.17.2	Zyklen.....	273
7.17.2.1	Zyklus löschen	273
7.17.3	Textlisten.....	274
7.17.3.1	Textliste löschen	274
7.17.4	Grafiklisten	275
7.17.4.1	Grafikliste löschen.....	275
7.17.5	Verbindungen.....	275
7.17.5.1	Verbindung löschen	275
7.17.6	Variablenliste	276
7.17.6.1	Benutzerdefinierte Ordner für HMI-Variablen erzeugen.....	276
7.17.6.2	Variablen einer HMI-Variablenliste enumerieren	276
7.17.6.3	Einzelne Variable aus einer HMI-Variablenliste löschen	277
7.17.6.4	Variablenliste aus einem Ordner löschen	278
7.17.7	VB-Skripte	278
7.17.7.1	Benutzerdefinierte Ordner für Skripte erzeugen	278
7.17.7.2	VB-Skript aus einem Ordner löschen	279
7.17.8	Benutzerdefinierten Ordner eines Bediengeräts löschen	280
7.18	Funktionen für den Zugriff auf ein HMI-Gerät (Unified).....	280
7.18.1	Objektliste	280
7.18.2	HMI-Unified-Software-Objekt	281
7.18.3	Fehler abfragen.....	283
7.18.4	Meldungen	286
7.18.4.1	Mit Analogmeldungen arbeiten	286
7.18.4.2	Mit einzelnen Bitmeldungen arbeiten	290
7.18.4.3	Mit Meldeklassen arbeiten	293
7.18.5	Logs	296

7.18.5.1	Arbeiten mit Datenarchiven.....	296
7.18.5.2	Arbeiten mit Meldearchiven.....	299
7.18.5.3	Arbeiten mit Archivvariablen	302
7.18.6	Variablen und VariablenTabellen	306
7.18.6.1	Mit VariablenTabellen arbeiten.....	306
7.18.6.2	Arbeiten mit HMI-Variablen	308
7.18.6.3	Arbeiten mit Systemvariablen	317
7.18.7	Verbindungen.....	318
7.18.7.1	Mit Zusammensetzungen arbeiten.....	318
7.18.8	Laufzeiteinstellungen	322
7.18.8.1	Arbeiten mit Laufzeiteinstellungen	322
7.18.9	Bilder und Dynamisierungen	323
7.18.9.1	Mit Bildern arbeiten	323
7.18.9.2	Grundlegende Bildobjekte.....	328
7.18.9.3	Element-Bildobjekte	351
7.18.9.4	Steuerbildobjekte	380
7.18.9.5	Bildbausteininstanz-Bildobjekte	402
7.18.9.6	Mit Ereignissen für Bilder/Bildobjekte und Eigenschaften arbeiten.....	409
7.18.9.7	Mit Dynamisierung und Ereignissen für Bilder/Bildobjekte über Skripte arbeiten	412
7.18.9.8	Mit Dynamisierung für Bilder/Bildobjekte arbeiten	415
7.18.9.9	Lizenz für den Zugriff auf ein Unified-Gerät prüfen	418
7.18.10	Zugriff auf Hierarchien im Common Plant Model	421
7.18.10.1	Mit der Anlagensicht arbeiten.....	421
7.18.10.2	Mit Anlagensichtknoten arbeiten	422
7.18.10.3	Anlagensicht enumerieren	423
7.18.10.4	Mit der Anlagensicht und Geräten arbeiten.....	425
7.18.11	Zugriff auf Instanzen im Common Plant Model	427
7.18.11.1	Mit CPM-Objektinstanzen arbeiten	427
7.18.11.2	Mit Anlagensichtknoten von CPM-Objektinstanzen arbeiten	428
7.18.11.3	Schnittstellenvariablen von CPM-Objektinstanzen enumerieren	429
7.18.12	Auf Eigenschaften für Schnittstellen/Archivvariablen von Anlagenobjektinstanzen zugreifen	431
7.18.12.1	Zugriff auf und Aktualisierung von Eigenschaften von Schnittstellenvariablen von CPM-Anlagenobjektinstanzen	431
7.18.12.2	Zugriff auf und Aktualisierung von Eigenschaften von Member-Variablen von Schnittstellenvariablen	433
7.18.12.3	Zugriff auf und Aktualisierung von Eigenschaften von Logging-Variablen von Member-Variablen	435
7.19	Funktionen für den Zugriff auf die Daten eines PLC-Geräts.....	437
7.19.1	Status einer PLC ermitteln	437
7.19.2	Auf Parameter einer Online-Verbindung zugreifen	439
7.19.3	Zugriff auf den Fingerabdruck zum schnellen Vergleich von Stationen	443
7.19.4	Zugriff auf CM DP als DP-Slave und Transferbereich	451
7.19.5	PLC von R/H-System online setzen.....	452
7.19.6	Auf Softwarebehälter über primären PLC eines R/H-Systems zugreifen	454
7.19.7	PLCs eines R/H-Systems laden.....	455
7.19.8	Funktionen zum Laden von Daten ins PLC-Gerät	462
7.19.8.1	PC-System laden	462
7.19.8.2	Hardware- und Softwarekomponenten ins PLC-Gerät laden.....	465
7.19.8.3	PLC starten und stoppen	475
7.19.8.4	Unterstützung von Callbacks	476
7.19.8.5	PLC durch Passwort schützen	478

7.19.8.6	Handhabung bausteingebundener PLC-Passwörter.....	479
7.19.9	Funktionen für den Zugriff auf den PLC-Dienst.....	480
7.19.9.1	Einstellung der Zugriffsstufen.....	480
7.19.9.2	Zugriff auf die OPC UA Server-Schnittstelle	482
7.19.9.3	Auf Software-Prüfsumme zugreifen	484
7.19.9.4	PC-Schnittstelle zuordnen.....	485
7.19.10	Hardware, Software und Dateien in PLC-Gerät laden	487
7.19.11	PLC-Software vergleichen	493
7.19.12	PLC-Hardware vergleichen	496
7.19.13	Online-Verbindung zur PLC aufbauen oder trennen	498
7.19.14	Projektsprache der PLC zuordnen	499
7.19.15	Beobachtungs- und Forcetabellen für Webserver- und PLC-Display zuordnen	500
7.19.16	Zugriff auf Webserver und OPC UA-Benutzerverwaltung.....	503
7.19.17	Zugriff auf Domänen-Einstellungen.....	506
7.19.18	Display-Passwort festlegen.....	508
7.19.19	Zugänglichkeit	509
7.19.20	Modulbeschreibung aktualisieren.....	510
7.19.21	Zertifikat verwalten	511
7.19.22	Bausteine	514
7.19.22.1	Gruppe "Programmbausteine" abfragen	514
7.19.22.2	Systemgruppe für Systembausteine abfragen	515
7.19.22.3	Systemuntergruppen enumerieren.....	515
7.19.22.4	Benutzerdefinierte Bausteingroupen enumerieren.....	517
7.19.22.5	Alle Bausteine enumerieren	518
7.19.22.6	Informationen eines Bausteins/Anwenderdatentyps abfragen	519
7.19.22.7	Einen Baustein schützen und Schutz aufheben.....	520
7.19.22.8	Baustein löschen.....	523
7.19.22.9	Gruppe für Bausteine erzeugen	523
7.19.22.10	Gruppe für Bausteine löschen.....	524
7.19.22.11	Auf Attribute sämtlicher Bausteine zugreifen	525
7.19.22.12	Einen ProDiag-FB anlegen	526
7.19.22.13	Auf Überwachungen und Eigenschaften des ProDiag-FB zugreifen	528
7.19.22.14	ProDiag-FB-Bausteine und -Attribute lesen	529
7.19.22.15	Externe Datei hinzufügen.....	530
7.19.22.16	Quelle aus Baustein generieren	531
7.19.22.17	Bausteine aus Quelle erzeugen	532
7.19.22.18	Generierung aus einer Quelle mit bekanntem Quellformat	533
7.19.22.19	Anwenderdatentyp löschen	536
7.19.22.20	Externe Datei löschen	536
7.19.22.21	Baustineditor starten	537
7.19.22.22	Bausteine mit Hilfe von Fingerabdrücken ändern	538
7.19.22.23	Bausteine für benutzerdefinierte Seiten generieren/löschen	540
7.19.23	Technologieobjekte	542
7.19.23.1	Übersicht über Funktionen für Technologieobjekte	542
7.19.23.2	Übersicht über Technologieobjekte und Versionen	543
7.19.23.3	Überblick der Datentypen.....	545
7.19.23.4	Zusammensetzung von Technologieobjekten abfragen	546
7.19.23.5	Technologieobjekt erstellen	546
7.19.23.6	Technologieobjekt löschen.....	547
7.19.23.7	Technologieobjekt übersetzen	548
7.19.23.8	Technologieobjekte enumerieren	549
7.19.23.9	Technologieobjekt finden	550

7.19.23.10	Parameter eines Technologieobjekts enumerieren.....	551
7.19.23.11	Parameter eines Technologieobjekts finden.....	551
7.19.23.12	Parameter eines Technologieobjekts lesen	552
7.19.23.13	Parameter eines Technologieobjekts schreiben	553
7.19.23.14	S7-1200 Motion Control	554
7.19.23.15	S7-1500 Motion Control	562
7.19.23.16	PID-Regelung.....	581
7.19.23.17	Zählen	582
7.19.23.18	Easy Motion Control.....	582
7.19.24	Variablen und VariablenTabellen	583
7.19.24.1	Starten des PLC-Variableneditors.....	583
7.19.24.2	Systemgruppen für PLC-Variablen abfragen	584
7.19.24.3	PLC-VariablenTabelle anlegen	584
7.19.24.4	Benutzerdefinierte Gruppen für PLC-Variablen enumerieren	585
7.19.24.5	Benutzerdefinierte Gruppen für PLC-Variablen erzeugen	586
7.19.24.6	Benutzerdefinierte Gruppen für PLC-Variablen löschen	587
7.19.24.7	PLC-VariablenTabellen in einem Ordner enumerieren	587
7.19.24.8	Informationen einer PLC-VariablenTabelle abfragen	588
7.19.24.9	Zeitpunkt der letzten Änderung einer PLC-VariablenTabelle lesen	590
7.19.24.10	PLC-VariablenTabelle aus einer Gruppe löschen	590
7.19.24.11	PLC-Variablen enumerieren.....	591
7.19.24.12	Auf PLC-Variablen zugreifen.....	592
7.19.24.13	Auf PLC-Konstanten zugreifen.....	593
7.19.25	Funktionen für Software-Einheiten	595
7.19.25.1	Mit Software-Einheiten arbeiten	595
7.19.25.2	Auf Software-Einheiten zugreifen.....	598
7.19.25.3	Auf Software-Einheiten zugrundeliegende Objekte zugreifen.....	599
7.19.25.4	Auf vorhandene Beziehungen einer Einheit zugreifen	602
7.19.25.5	Eigenschaften von Software-Einheiten aktualisieren	604
7.19.25.6	Objekt einer Software-Einheit veröffentlichen	606
7.19.25.7	Externe Quellen in Einheiten hinzufügen	608
7.19.25.8	Einheiten als Masterkopien	611
7.19.25.9	Vorhandene Beziehungen aktualisieren und Beziehungen erstellen/löschen	613
7.20	Funktionen der Versionskontrollschnittstelle (VCI)	618
7.20.1	Auf die VCI-Systemgruppe im Projekt zugreifen.....	618
7.20.2	Benutzergruppen in VCI-Gruppe enumerieren	619
7.20.3	VCI-Benutzergruppe in VCI-Gruppe erstellen.....	620
7.20.4	Eigenschaften von VCI-Gruppen aktualisieren	621
7.20.5	VCI-Benutzergruppe löschen	622
7.20.6	VCI-Workspaces in VCI-Gruppe enumerieren	622
7.20.7	VCI-Workspace in VCI-Gruppe erstellen	623
7.20.8	VCI-Workspace-Eigenschaften aktualisieren.....	624
7.20.9	VCI-Workspace löschen.....	625
7.20.10	VCI-Workspace-Zuordnungen in VCI enumerieren	626
7.20.11	VCI-Workspace-Zuordnung in VCI-Workspace erstellen.....	627
7.20.12	VCI-Workspace-Zuordnungen aktualisieren	628
7.20.13	VCI-Workspace-Zuordnung löschen	629
7.20.14	Workspace-Zuordnung synchronisieren	629
7.21	Funktionen auf OPC.....	633
7.21.1	Konfiguration des sicheren Kommunikationsprotokolls für den OPC UA-Server	633
7.21.2	Sicherheitsrichtlinie für OPC UA festlegen.....	635

7.22	SiVArc Openness	636
7.22.1	Einleitung	636
7.22.2	SiVArc-Diensteigenschaften	637
7.22.3	Kopieren von Regeln oder Gruppen aus der Bibliothek	638
7.22.4	Suchen von Bildregeln oder Bildregelgruppen	640
7.22.5	Löschen von Regeln und Regelgruppen	641
7.22.6	UMAC-Einrichtung für Openness	642
7.22.7	SiVArc-Generierung	642
7.23	Openness für CP 1604/CP 1616/CP 1626	645
7.24	Openness-Transferbereiche für PN/PN-Koppler	648
7.25	Virtuelle Openness-Module/Submodule für ET 200SP PN HF	652
7.26	Funktionen für SINUMERIK	654
7.26.1	Einführung	654
7.26.2	Typkennung- Kennzeichnung der Komponenten	655
7.26.3	Grundlagen	655
7.26.4	Objektmodell	659
7.26.5	Referenz	661
7.26.5.1	DriveObject	661
7.26.5.2	DriveObjectContainer	661
7.26.5.3	DriveObjectComposition	662
7.26.5.4	AddressComposition	662
7.26.5.5	AddressContext	664
7.26.5.6	AddressIoType	664
7.26.6	Codebeispiel	664
7.26.6.1	Allgemein	664
7.26.6.2	Erste Schritte in SINUMERIK durchführen	665
7.26.6.3	NCU anlegen	665
7.26.6.4	NX-Baugruppe anlegen	665
7.26.6.5	NX-Baugruppe mit NCU verbinden	666
7.26.6.6	Archive erstellen	667
7.26.6.7	Safety Integrated aktivieren	669
7.27	Funktionen für SINUMERIK ONE	670
7.27.1	Einführung	670
7.27.2	Typkennung-Kennzeichnung der Komponenten	671
7.27.3	Grundlagen	671
7.27.4	Objektmodell	675
7.27.5	Referenz	677
7.27.5.1	Namespace Siemens.Engineering.MC.DriveConfiguration	677
7.27.5.2	Namespace Siemens.Engineering.MC.Drives	683
7.27.5.3	ArchiveProvider	693
7.27.6	Codebeispiel	693
7.27.6.1	Allgemein	693
7.27.6.2	Erste Schritte in SINUMERIK durchführen	693
7.27.6.3	NCU anlegen	694
7.27.6.4	NX-Baugruppe anlegen	694
7.27.6.5	NX-Baugruppe mit NCU verbinden	695
7.27.6.6	Auf NCK-Ereignisse zugreifen	695
7.27.6.7	Archive erstellen	697
7.27.6.8	Safety Integrated aktivieren	698

7.27.6.9	Beispiele für Namespace Siemens.Engineering.MC.DriveConfiguration.....	699
7.27.6.10	Beispiele für Namespace Siemens.Engineering.MC.Drives	703
7.28	Funktionen für Startdrive.....	707
7.28.1	Einführung	707
7.28.2	TypeIdentifier - Kennzeichnung der Komponenten.....	708
7.28.3	Referenzen.....	709
7.28.3.1	AddressComposition	709
7.28.3.2	AddressContext.....	710
7.28.3.3	AddressIoType	710
7.28.3.4	ConfigurationEntry	711
7.28.3.5	DriveDomainFunctions	711
7.28.3.6	DriveObject	712
7.28.3.7	DriveObjectActivation	712
7.28.3.8	DriveObjectContainer	713
7.28.3.9	DriveObjectTypeHandler	713
7.28.3.10	DriveParameter	714
7.28.3.11	DriveParameterComposition	715
7.28.3.12	EncoderConfiguration	716
7.28.3.13	HardwareProjection.....	717
7.28.3.14	MotorConfiguration.....	718
7.28.3.15	OnlineDriveObject	719
7.28.3.16	OnlineDriveObjectContainer	719
7.28.3.17	StartDriveDownloadCheckConfiguration.....	720
7.28.3.18	SafetyTelegram	720
7.28.3.19	Telegram	721
7.28.3.20	TelegramComposition	722
7.28.3.21	TelegramType	723
7.28.3.22	TorqueTelegram.....	724
7.28.4	Codebeispiele	724
7.28.4.1	Aktivierungsstatus ermitteln	724
7.28.4.2	Antriebsfunktionen durchführen	725
7.28.4.3	Antriebsgerät anlegen	726
7.28.4.4	Antriebskomponente anlegen	727
7.28.4.5	Antriebsobjekt ermitteln.....	727
7.28.4.6	Antriebsobjekt-Typ ermitteln	728
7.28.4.7	BICO-Parameter lesen und schreiben	728
7.28.4.8	Download	729
7.28.4.9	DRIVE-CLiQ-Verbindungen bearbeiten	731
7.28.4.10	Erste Schritte in Startdrive durchführen	732
7.28.4.11	Geber-Typ festlegen	733
7.28.4.12	Geräte-Projektierung durchführen.....	740
7.28.4.13	Komponente für eine Antriebskomponente anlegen (nur S120)	741
7.28.4.14	Motor-Typ und Motor-Konfiguration festlegen.....	742
7.28.4.15	Parameter lesen und schreiben	745
7.28.4.16	Parameter online lesen und schreiben.....	747
7.28.4.17	Parametrierung speichern	747
7.28.4.18	Safety Integrated-Telegramme verwenden	748
7.28.4.19	Telegramme einfügen und erweitern	749
7.28.4.20	Torque-Telegramme verwenden	750
7.28.4.21	Werkseinstellungen wiederherstellen.....	751
7.29	Funktionen für DCC	751

7.29.1	Einführung	751
7.29.2	DCC Openness	752
7.29.3	Objektmodell DCC Openness	753
7.29.4	Referenzen.....	753
7.29.4.1	DriveControlChartContainer	753
7.29.4.2	DriveControlChartComposition	754
7.29.4.3	DcclImportOptions	755
7.29.4.4	DcclImportResultData	755
7.29.4.5	DriveControlChart	756
7.29.4.6	Import DCB extension library	757
7.29.5	Codebeispiele	757
7.29.5.1	Allgemein	757
7.29.5.2	Zugriff auf DriveControlChartContainer über DriveObject	757
7.29.5.3	Pläne abrufen.....	757
7.29.5.4	Auf Pläne zugreifen.....	758
7.29.5.5	Pläne importieren	758
7.29.5.6	Pläne exportieren	758
7.29.5.7	Alle Pläne exportieren	759
7.29.5.8	Pläne in der Ablaufreihenfolge abrufen	759
7.29.5.9	Pläne anhand des Namens finden	759
7.29.5.10	Importbericht anzeigen.....	760
7.29.5.11	Pläne löschen.....	760
7.29.5.12	Ablaufreihenfolge optimieren	760
7.29.5.13	DCB Extension-Bibliothek importieren	761
7.29.6	Ausnahmen DCC Openness	762
7.29.6.1	Umgang mit Exceptions	762
7.29.6.2	DriveControlChart Exceptions	764
7.29.6.3	DriveControlChartComposition Exceptions	764
7.29.6.4	ImportDcbLibrary Exceptions	765
7.30	Ausnahmen	766
7.30.1	Umgang mit Exceptions	766
7.30.2	Custom Exception	769
8	Export/Import	773
8.1	Überblick	773
8.1.1	Grundlagen zum Import/Export	773
8.1.2	Einsatzgebiet von Import/Export	776
8.1.3	Versionsspezifischer SimaticML-Import	777
8.1.4	Bearbeiten der XML-Datei.....	778
8.1.5	Export von Projektierungsdaten	778
8.1.6	Import von Projektierungsdaten	780
8.2	Import/Export von Projektdaten	782
8.2.1	Grafiksammlung	782
8.2.1.1	Export/Import von Grafiken	782
8.2.1.2	Grafiken eines Projektes exportieren	783
8.2.1.3	Grafiken in ein Projekt importieren	784
8.2.2	Projekttexte	785
8.2.2.1	Export von Projekttexten	785
8.2.2.2	Import von Projekttexten	786
8.3	Import/Export von Daten eines HMI-Geräts	788
8.3.1	Aufbau einer XML-Datei.....	788

8.3.2	Struktur der Daten für Import/Export	790
8.3.3	Zyklen.....	794
8.3.3.1	Zyklen exportieren.....	794
8.3.3.2	Zyklen importieren.....	795
8.3.4	VariablenTabellen	796
8.3.4.1	HMI-VariablenTabellen exportieren.....	796
8.3.4.2	HMI-VariablenTabelle importieren.....	799
8.3.4.3	Einzelne Variable einer HMI-VariablenTabelle exportieren.....	800
8.3.4.4	Einzelne Variable in eine HMI-VariablenTabelle importieren	801
8.3.4.5	Besonderheiten beim Export/Import von HMI-Variablen	801
8.3.5	VB-Skripte	803
8.3.5.1	VB-Skripte exportieren	803
8.3.5.2	VB-Skripte aus einem Ordner exportieren	804
8.3.5.3	VB-Skripte importieren	805
8.3.6	Textlisten.....	806
8.3.6.1	Textlisten aus einem HMI-Gerät exportieren	806
8.3.6.2	Textliste in ein HMI-Gerät importieren	807
8.3.6.3	Erweiterte XML-Formate für Export/Import von Textlisten	808
8.3.7	Grafiklisten	810
8.3.7.1	Grafiklisten exportieren	810
8.3.7.2	Grafikliste importieren	810
8.3.8	Verbindungen.....	811
8.3.8.1	Verbindungen exportieren.....	811
8.3.8.2	Verbindungen importieren.....	812
8.3.9	Bilder	813
8.3.9.1	Übersicht der exportierbaren Bild-Objekte	813
8.3.9.2	Alle Bilder eines HMI-Geräts exportieren.....	817
8.3.9.3	Bild aus einem Bildordner exportieren	818
8.3.9.4	Bilder in ein HMI-Gerät importieren.....	820
8.3.9.5	Permanentfenster exportieren.....	823
8.3.9.6	Permanentfenster importieren.....	823
8.3.9.7	Alle Bildvorlagen eines HMI-Geräts exportieren	824
8.3.9.8	Bildvorlagen aus einem Ordner exportieren.....	825
8.3.9.9	Bildvorlagen importieren	827
8.3.9.10	Exportieren eines Pop-up-Bilds.....	829
8.3.9.11	Importieren eines Pop-up-Bildes	830
8.3.9.12	Exportieren eines Slide-in-Bilds	831
8.3.9.13	Importieren eines Slide-in-Bilds	833
8.3.9.14	Bild mit Eingabemaske exportieren.....	834
8.3.9.15	Bild mit Eingabemaske importieren.....	836
8.4	Beobachtungs- und Forcetabelle exportieren/importieren	839
8.5	Daten eines PLC-Geräts importieren/exportieren	841
8.5.1	Bausteine	841
8.5.1.1	XML-Struktur des Abschnitts Bausteinschnittstelle	841
8.5.1.2	Änderungen des Objektmodells und Dateiformat XML	851
8.5.1.3	DBs mit Schnappschüssen exportieren	853
8.5.1.4	Bausteine mit Know-how-Schutz exportieren	855
8.5.1.5	Export/Import von SCL-Bausteinen.....	856
8.5.1.6	Export/Import strukturierter Typen von SCL-Bausteinen	870
8.5.1.7	Export/Import von SCL-Aufrufbausteinen	876
8.5.1.8	Mehrsprachige Kommentare in SCL exportieren/importieren	893

8.5.1.9	F-Bausteine exportieren.....	894
8.5.1.10	System-Bausteine exportieren	894
8.5.1.11	GRAPH-Bausteine mit mehrsprachigem Text exportieren.....	895
8.5.1.12	Baustein importieren	896
8.5.1.13	Bausteine exportieren	898
8.5.1.14	Bausteine/UDT mit offenem Verweis importieren	905
8.5.1.15	Bausteine/UDT für Strukturänderungsobjekte importieren	906
8.5.1.16	Export/Import des Unit-spezifischen Veröffentlichungsattributs von Bausteinen und Typen	908
8.5.2	Technologieobjekte	912
8.5.2.1	S7-1200 Motion Control	912
8.5.2.2	S7-1500 Motion Control	912
8.5.2.3	PID-Regelung.....	912
8.5.2.4	Zählen	912
8.5.2.5	Easy Motion Control.....	912
8.5.3	Variablenlisten	912
8.5.3.1	PLC-Variablenlisten exportieren	912
8.5.3.2	PLC-Variablenliste importieren	914
8.5.3.3	Einzelne Variable oder Konstante aus einer PLC-Variablenliste exportieren	914
8.5.3.4	Einzelne Variable oder Konstante in eine PLC-Variablenliste importieren	916
8.5.4	Anwenderdatentyp exportieren	917
8.5.5	Anwenderdatentyp importieren	917
8.5.6	Export von Daten im Format OPC UA XML	919
8.6	Hardware-Daten importieren/exportieren	919
8.6.1	AML-Dateiformat	919
8.6.2	Pruned AML	920
8.6.3	Übersicht über Objekte und Parameter von CAx-Import/-Export	921
8.6.4	Struktur der CAx-Daten zum Import/Export	923
8.6.5	AML-Typkennungen	928
8.6.6	Exportieren/Importieren von Informationen zur Basiseinheit über AML	931
8.6.7	Export/Import einer AML-Datei mit Verbindung zum Erweiterungsbaugruppenträger	934
8.6.8	Verwalten von Verbindungen bei Erweiterungsbaugruppenträgern	940
8.6.9	Export von CAx-Daten	941
8.6.10	Import von CAx-Daten.....	944
8.6.11	Export/Import von Submodulen.....	946
8.6.12	CAx-Daten ohne logische Adresse importieren	955
8.6.13	Ausnahmen beim Import und Export von CAx-Daten	962
8.6.14	Round-Trip-Geräte und -Module	963
8.6.15	AML importieren und dabei unbekannte Artefakte ignorieren.....	966
8.6.16	Export/Import-Topologie.....	971
8.6.17	Import eines Geräts mit Bibliotheksreferenzen	973
8.6.18	Export eines Geräteelements.....	974
8.6.19	Import eines Geräteteleobjekts	978
8.6.20	Export/Import eines Geräts mit festgelegter Adresse	981
8.6.21	Export/Import eines Geräts mit Kanälen	984
8.6.22	Export von Geräteelementobjekten.....	986
8.6.23	Import von Geräteelementobjekten.....	991
8.6.24	Export/Import von GSD/GSDML-basierten Geräten und Geräteelementen	994
8.6.25	Exportieren/Importieren der Gerätekonfiguration mit virtueller Schnittstelle	1000
8.6.26	Exportieren/Importieren von Subnetzen.....	1003
8.6.27	Export/Import von IO-Systemen.....	1011
8.6.28	Exportieren/Importieren mehrsprachiger Kommentare	1013

8.6.29	Unterstützung von Attributen bei AML AR APC 1.1	1014
8.6.29.1	Exportieren/Importieren von PLC-Variablen	1014
8.6.29.2	Export/Import von RH/PLC.....	1017
8.6.29.3	Export/Import von AML mit benutzerdefinierten Variablen und Geräteelementen.....	1019
8.6.29.4	Import/Export fehlersicherer PLCs	1022
8.6.29.5	Import/Export fehlersicherer E/A	1028
8.6.29.6	Herstellerspezifisches Attribut exportieren/importieren	1029
8.6.30	AML-Attribute im Vergleich zu TIA Portal Openness-Attributen.....	1030
9	Wesentliche Änderungen	1035
9.1	Größere Änderungen in TIA Portal Openness V15.1.....	1035
9.2	Größere Änderungen in TIA Portal Openness V15.....	1038
9.3	Die wichtigsten Änderungen in V14 SP1	1040
9.3.1	Die wichtigsten Änderungen in V14 SP1	1040
9.3.2	Wesentliche Änderungen im Objektmodell	1043
9.3.3	Änderungen an der Pilotfunktionalität	1047
9.3.4	Änderungen bei Export und Import	1052
9.3.4.1	Änderungen bei Export und Import	1052
9.3.4.2	Änderungen in der API	1052
9.3.4.3	Schemaerweiterung	1053
9.3.4.4	Schemaänderungen.....	1056
9.3.4.5	Verhaltensänderungen	1059
9.3.4.6	Änderungen an Bausteinattributen.....	1070
9.4	Die wichtigsten Änderungen in V14	1072
9.4.1	Wesentliche Änderungen des Objektmodells	1072
9.4.2	Vor dem Hochrüsten einer Anwendung auf TIA Portal Openness V14	1074
9.4.3	Wesentliche Stringänderungen	1075
9.4.4	Importieren von Dateien, die mit TIA Portal Openness V13 SP1 und älter erzeugt wurden..	1078
Index.....	1081	

Sicherheitshinweis

Sicherheitsinformationen

Siemens bietet Produkte und Lösungen mit Industrial Security-Funktionen zur Unterstützung des sicheren Betriebs von Anlagen, Systemen, Maschinen, Geräten und/oder Netzwerken.

Um Anlagen, Systeme, Maschinen und Netzwerke vor Cyberbedrohungen zu schützen, ist es notwendig, ein ganzheitliches, modernes Industrial Security-Konzept einzubinden und kontinuierlich zu pflegen. Die Produkte und Lösungen von Siemens bilden nur ein Element eines solchen Konzepts.

Der Kunde ist dafür verantwortlich, den unberechtigten Zugriff auf seine Anlagen, Systeme, Maschinen und Netzwerke zu verhindern. Systeme, Maschinen und Komponenten dürfen nur in dem erforderlichen Umfang und mit entsprechenden Sicherheitsmaßnahmen (z. B. Firewalls und Netzwerksegmentierung) mit dem Unternehmensnetzwerk oder dem Internet verbunden werden.

Außerdem sollten die Richtlinien von Siemens zu entsprechenden Sicherheitsmaßnahmen berücksichtigt werden. Weitere Informationen zu Industrial Security finden Sie unter

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

Die Produkte und Lösungen von Siemens werden ständig weiterentwickelt, um sie sicherer zu machen. Siemens empfiehlt, Produkt-Updates bei Verfügbarkeit unbedingt anzuwenden und stets die neuesten Produktversionen zu verwenden. Die Verwendung von Produktversionen, die nicht mehr unterstützt werden, und die Missachtung der neuesten Updates kann das Risiko des Kunden gegenüber Cyberbedrohungen erhöhen.

Um stets über Produkt-Updates informiert zu sein, melden Sie sich für den Siemens Industrial Security RSS-Feed an unter

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

Liesmich zu TIA Portal Openness

2.1 Liesmich

Sicherheitsmaßnahmen für TIA Portal Openness-Anwendungen

Es ist empfehlenswert,

- eine TIA Portal Openness-Anwendung mit Administratorrechten im Ordner "Programme" zu installieren.
- Das dynamische Laden von Programmbestandteilen wie Assemblies oder DLLs aus dem Benutzerbereich zu vermeiden.
- Die TIA Portal Openness-Anwendung mit Benutzerrechten auszuführen.

Hardware-Parameter

Eine Beschreibung der Hardware-Parameter steht im Installationsordner von TIA Portal unter Siemens\Automation\Portal V*\PublicAPI\V*\HW Parameter Description \Openness_hwparameter_description.pdf zur Verfügung.

Hinweis

V* bezieht sich auf den entsprechend der installierten Version von TIA Portal angepassten Pfad.

Kopieren einer TIA Portal Openness-Anwendung

Wenn Sie eine ausführbare TIA Portal Openness-Anwendung kopieren, kann es unter bestimmten Umständen vorkommen, dass der Verzeichnispfad, in dem die TIA Portal Openness-Anwendung ursprünglich erzeugt wurde, von der TIA Portal Openness-Anwendung ausgelesen wird.

Abhilfe:

Wenn Sie die TIA Portal Openness-Anwendung in ein neues Verzeichnis kopiert haben, öffnen und schließen Sie den Dialog "Eigenschaften", um den Windows-Cache zu aktualisieren.

Unterstützung bestimmter Funktionen in einem TIA Portal-Projekt

Mehrbenutzerbetrieb

TIA Portal Openness unterstützt keine administrativen Mehrbenutzeroperationen. Der Grund dafür ist, dass der Einsatz von TIA Portal Openness in Mehrbenutzerprojekten nicht empfohlen wird. Beachten Sie, dass bestimmte Aktionen in TIA Portal Openness den von der Benutzeroberfläche des TIA Portals vorgegebenen Mehrbenutzer-Arbeitsfluss sogar stören

2.1 Liesmich

können. Wenn Sie Änderungen dennoch mit TIA Portal Openness vornehmen möchten, exportieren Sie zuvor das Mehrbenutzerprojekt in ein Einzelbenutzerprojekt.

Fehlersicherheit

Wenn Sie mit TIA Portal Openness arbeiten, sind gewisse Einschränkungen in Bezug auf die Fehlersicherheit zu beachten. Weitere Informationen finden Sie in der Dokumentation "SIMATIC Sicherheitssysteme - Konfiguration und Programmierung".

Verbesserung der Leistung von TIA Portal Openness

Um die maximale Leistung von TIA Portal Openness zu erreichen, können Sie die globale Suchfunktion des TIA Portals ausschalten. Zum Ausschalten der globalen Suche nutzen Sie die Benutzeroberfläche oder den TIA Portal Openness API-Aufruf. Wenn die TIA Portal Openness-Anwendung fertig ist, kann die globale Suche wieder eingeschaltet werden. Obwohl sich die Leistung durch das Ausschalten der globalen Suche verbessert, arbeiten alle TIA Portal Openness-Funktionen auch mit eingeschalteter globaler Suche einwandfrei.

Threadsicherer Programmcode

Achten Sie darauf, dass Ihr Code threadsicher ist: Ein Ereignis erscheint in verschiedenen Threads.

Exportverhalten von Bildschirmelementen mit aktiviertem Style

Beim Export eines Bildschirmelements mit aktiviertem Style werden nicht die Attribute des Style-Elements exportiert, sondern nur die Attribute des Bildschirmelements vor Aktivierung des Styles. Wenn ein Style ausgewählt wird und "UseDesignColorSchema" für das Bildschirmelement aktiviert ist, übernimmt das Bildschirmelement auf der Benutzeroberfläche die Attributwerte des Styles. In der Datenbank werden für dieses Bildschirmelement jedoch noch immer die Attributwerte gespeichert, die vor Auswahl des Styles festgelegt waren. TIA Portal Openness exportiert diese tatsächlichen, in der Datenbank gespeicherten Werte.

Wird der Style deaktiviert und wieder aktiviert und das Bildschirmelement dann erneut exportiert, werden für dieses Bildschirmelement die im Style-Element geltenden Attributwerte exportiert. Die Attributwerte des für dieses Bildschirmelement ausgewählten Style-Elements werden in der Datenbank gespeichert, wenn "UseDesignColorSchema" nicht aktiviert ist.

Dieses Problem kann wie folgt behoben werden:

1. Weisen Sie das Bildschirmelement dem Style-Element zu:
 - Die Datenbank enthält die Attributwerte, die vor Aktivierung des Styles galten.
 - Die Benutzeroberfläche übernimmt Attribute direkt vom Style-Element.
2. Exportieren Sie das dem Style-Element zugewiesene Bildschirmelement:
 - Die XML-Datei enthält die Attributwerte aus der Datenbank, die den Werten vor Aktivierung des Styles entsprechen.
3. Deaktivieren Sie "UseDesignColorSchema":
 - Die Attributwerte des Style-Elements werden in der Datenbank den Attributen des Bildschirmelements hinzugefügt.

4. Aktivieren Sie "UseDesignColorSchema":
 - Die Attributwerte des Bildschirmelements in der Datenbank werden nicht verändert und entsprechen noch denen aus Schritt 3.
 - Die Benutzeroberfläche übernimmt Attribute direkt vom Style-Element.
5. Exportieren Sie das dem Style-Element zugewiesene Bildschirmelement:
 - Die XML-Datei enthält die Attributwerte aus der Datenbank, die in Schritt 3 festgelegt wurden und den Werten des Style-Elements entsprechen.

Kopieren von S7-1500 Technologieobjekten für die Bewegungssteuerung

Das Kopieren von TO_CamTrack, TO_OutputCam oder TO_MeasuringInput aus der Projektbibliothek oder globalen Bibliothek in das Projekt ist nicht möglich.

Importieren von ASi-Slaves über AML

Wird einer der folgenden ASi-Slaves über eine AML-Datei importiert, wird die Firmware-Version des Geräteelements in jedem Fall auf V13.0 gesetzt:

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

Exportieren und Importieren von Funktionstasten

Funktionstasten werden während des Imports synchronisiert. Wenn eine Funktionstaste im globalen Bildschirm erzeugt wird und die Taste im Bildschirm leer ist, nutzt die entsprechende Funktionstaste die globale Definition in allen Bildschirmen.

Wenn Sie die globale Nutzung von Funktionstasten nach dem Import deaktivieren möchten, definieren Sie leere Tasten in den Bildschirmen und importieren die Bildschirmtypen in der folgenden Reihenfolge: Globaler Bildschirm, Vorlagen, Bildschirme.

Wenn Sie beim Exportieren von Bildschirmen sicherstellen möchten, dass die globale Definition einer Funktionstaste nicht von der Vorlage oder vom globalen Bildschirm genutzt wird, erzeugen Sie eine leere Funktionstaste in dem Bildschirm. Wählen Sie die erforderliche Funktionstaste im Bildschirm. Aktivieren Sie anschließend die Eigenschaft "Use global assignment" und deaktivieren Sie diese wieder.

Zugriff auf ein Gerät, das online ist

Das Schreiben von Attributen in ein Gerät, das online ist, wird nicht unterstützt. Das Lesen von Attributen wird unterstützt.

Das Trennen von einem Subnetz, während das Gerät online ist, wird nicht unterstützt.

Instanzspezifische Attribute beim Importieren von Bausteinen über TIA Portal Openness

In bestimmten Situationen können die Importregeln den Verlust von instanzenspezifischen Attributen wie zum Beispiel Anlaufwerten bedeuten.

Informationen zu bestimmten Funktionen

Beachten Sie die FAQ-Einträge beim Siemens Industry Online Support für weitere Informationen zu den folgenden Openness-Funktionen:

- Projekt archivieren/wiederherstellen
- Beobachtungstabellen exportieren/importieren

2.2 Größere Änderungen in TIA Portal Openness V16

Änderungen

Wenn Sie die Hinweise zur versionsübergreifenden Programmierung beachtet haben und Ihr Projekt nicht auf V16 aktualisieren, laufen Ihre Anwendungen ohne jede Einschränkung auf jedem Rechner, auch wenn nur ein TIA Portal V16 installiert ist.

Wenn Sie Ihr Projekt auf V16 aktualisieren, ist es notwendig, Ihre Anwendung mit der SiemensEngineering.dll von V16 neu zu übersetzen. In manchen Fällen kann es erforderlich sein, den Code Ihrer Anwendung anzupassen.

Exportieren von Datenbausteinen mit Export-Option "Keine"

Ab TIA Portal Openness V16 werden schreibgeschützte Teile eines Datenbausteins als informative Elemente exportiert. Diese Attribute können in der exportierten XML nicht mehr geändert werden.

Adressattribute von I&M und Profisafe für Drucktaster- und Tastentafeln

Ab TIA Portal Openness V16 stehen Adressattribute von I&M und Profisafe für die Drucktaster- und Tastentafeln auf Modulebene zur Verfügung.

Exportieren von Datenbausteinattributen der Eigenschaft "Speicheraufbau"

Ab TIA Portal Openness V16 werden Instanz-DBs von FBs, Attribute von ARRAY-DBs und GRAPH-Bausteinen mit ReadOnly="True" exportiert.

Nicht unterstützte Attribute in Einheiten

Ab TIA Portal Openness V16 wird der inkonsistente Import von Einheitsbausteinen und anwenderdefinierten Datentypen (UDT) mit dem Attribut "Access" unterstützt, Ausnahme sind VariablenTabellen.

Schemadefinition der XML-Datei von SimaticML

Ab TIA Portal Openness V16 ändert sich der Standardwert des Attributs SystemDefined für den Teilnehmer des booleschen Attributs in der Schemadefinition der XML-Datei von SimaticML zu "False". Diese Änderung hat jedoch keine Auswirkungen auf Funktionen für den Export/Import von XML-Dateien Sie ist nur relevant für Benutzer, die versuchen, XML-Dateien mit Hilfe der Schemadatei zu generieren.

XML-Importmethode für Bausteine und Typen verbessert

Ab TIA Portal Openness V16 verfügt die neue Importmethode über zusätzliche Parameter (path, importOptions und swlImportOptions) sowie den neuen Enumerationswert IgnoreUnitAttribute. Mit swlImportOptions.IgnoreUnitAttributes können Sie einen Baustein aus einer Einheit in eine Zusammenstellung von Nichteinheiten importieren.

AML-GUIDs werden in App-IDs gespeichert

Ab TIA Portal Openness V16 werden AML-GUIDs im Attribut CustomIdentity (App-IDs) und nicht mehr in Comment gespeichert, um den Round-Trip-Austausch von Gerät und Modul zu unterstützen.

Name von HMI-Objektklassen

Ab TIA Portal Openness V16 enthält die folgende Tabelle eine Liste der erforderlichen Namensänderungen für HMI-Objektklassen:

Klassenname	Neuer Klassenname
AnalogAlarmComposition	HmiAnalogAlarmComposition
DiscreteAlarmComposition	HmiDiscreteAlarmComposition
AlarmClassComposition	HmiAlarmClassComposition
DataLogComposition	HmiDataLogComposition
AlarmLogComposition	HmiAlarmLogComposition
LoggingTagComposition	HmiLoggingTagComposition
LogConfiguration	DataLog

Eigenschaftenname für Datenarchiv/Meldearchiv hinzugefügt/entfernt

Ab TIA Portal Openness V16 werden die folgenden neuen Eigenschaften "StorageDevice" und "StorageFolder" dem DataLog/AlarmLog (Datenarchiv/Meldearchiv) hinzugefügt und Eigenschaften wie "StoragePath" und "RequireExplicitRelease" werden aus DataLog/AlarmLog und ScreenItems entfernt.

Das aktualisierte Codebeispiel für die Eigenschaften StorageDevice und StorageFolder finden Sie hier:

```
HmiDataLog dataLog = hmiSoftware.DataLogs.Find("DataLog1");
dataLog.Settings.StorageDevice = DeviceNode.Local;
dataLog.Settings.StorageFolder = @"D:\workdir\DataLogs";
```

Eigenschaftsname aus HMI Tag entfernt

Ab TIA Portal Openness V16 ist die Eigenschaft 'DisplayName' aus dem Objekt HMI Tag entfernt.

2.3 Bekanntgabe größerer Änderungen in künftigen Releases

Bekanntgabe von Änderungen

Die TIA Portal Openness API wird in einer späteren Version geändert. Es besteht keine Notwendigkeit, den Code Ihrer Anwendung sofort zu ändern, weil Anwendungen auf der Grundlage früherer Releases ohne jede Einschränkung laufen. Aber für neue Anwendungen empfiehlt es sich, die neue Funktionalität zu nutzen und die Neucodierung Ihrer Anwendung zu planen, da ab V17 die folgenden Methoden nicht mehr unterstützt werden.

- Typ von Zusammensetzungen und Zuweisungen

Typ von Zusammensetzungen und Zuweisungen

Die folgenden Typen ändern sich, um das Momentabbildverhalten anzuzeigen:

- AddressAssociation
- AddressComposition
- AddressControllerAssociation
- ChannelComposition
- DeviceItemAssociation
- DeviceItemComposition
- HwIdentifierAssociation
- HwIdentifierComposition
- HwIdentifierControllerAssociation
- IoConnectorComposition
- IoControllerComposition

Simocode Openness

Bitte beachten Sie, dass zwar alle Parameter der Wahrheitstabelle in Openness V15.1 genutzt werden können, sich ihre Implementierung in V16 jedoch ändert.

Anstatt jedes Bit einzeln festzulegen, wird es möglich sein, alle Output-Bits einer Wahrheitstabelle mit einer schnellen Array-Operation einzustellen.

Wenn Sie Ihre bestehende Openness-Anwendung in V16 weiterverwenden möchten, müssen Sie Ihren Code ggf. an die neuen Gegebenheiten anpassen.

Außenleiter eines Energy Meter

Die Außenleiter eines Energy Meter waren als Kanal ausgeführt. Ab V16 werden sie als Strukturen basierend auf komplexen Datentypen eingerichtet. Für neue Anwendungen wird empfohlen, auf Außenleiter als Strukturen zuzugreifen.

2.4 Hinweise zum Schreiben von langfristig stabilem Code

Versionswechsel

Wenn Sie einige Hinweise zum Schreiben von langfristig stabilem Code beachten, werden Sie Ihre Anwendung mit anderen Versionen des TIA Portals nutzen können, ohne den Code Ihrer Anwendung zu ändern.

Hinweis

V* und *.ap* im Dokument beziehen sich auf den entsprechend der installierten Version des TIA Portals angepassten Pfad und die Erweiterung.

Registry-Pfad und Datei appconfig

Modifikationen sind notwendig, um den Registry-Pfad und die Konfigurationsdatei der Anwendung zu ändern. Beispiel:

"C:\Programme\Siemens\Automation\Portal V14\PublicAPI\V14
SP1\Siemens.Engineering.dll"

muss geändert werden in

"C:\Programme\Siemens\Automation\Portal V*\PublicAPI\V*\Siemens.Engineering.dll".

Um langfristig stabilen Code zu schreiben, muss der Registry-Pfad konfigurierbar sein und die Konfigurationsdatei der Anwendung muss aktualisiert werden.

Installationspfad

Modifikationen sind notwendig, um den Installationspfad des TIA Portals zu ändern. Beispiel:

"C:\Programme\Siemens\Automation\Portal V14\PublicAPI\V14

SP1\Siemens.Engineering.dll"

muss geändert werden in

"C:\Programme\Siemens\Automation\Portal V*\PublicAPI\V*\Siemens.Engineering.dll".

Um langfristig stabilen Code zu schreiben, muss der Installationspfad konfigurierbar sein.

Pfad von AmiHost

Modifikationen sind notwendig, um den Pfad von AmiHost zu ändern. Beispiel:

"C:\Programme\Siemens\Automation\Portal V14\bin\Siemens.Automation.Cax.AmiHost.exe"

muss geändert werden in "C:\Programme\Siemens\Automation\Portal V*\bin
\Siemens.Automation.Cax.AmiHost.exe".

Um langfristig stabilen Code zu schreiben, muss der Pfad von AmiHost konfigurierbar sein.

Erweiterungen von Projektdateien und Bibliotheken des TIA Portals

Modifikationen sind notwendig, um die Erweiterungen von Projektdatei und Bibliotheken des TIA Portals zu ändern. Beispiel:

*.ap14
muss geändert werden in
.ap.

Um langfristig stabilen Code zu schreiben, müssen Erweiterungen von Projektdateien und Bibliotheken des TIA Portals konfigurierbar sein.

Öffnen eines Projekts

Um langfristig stabilen Code zu schreiben, muss die Methode `Projects.OpenWithUpgrade` statt der Methode `Projects.Open` verwendet werden.

Hierarchie beim Vergleichen, Übersetzen oder Herunterladen von Ergebnissen

Die Hierarchie und/oder die Reihenfolge beim Vergleichen, Übersetzen oder Herunterladen von Ergebnissen kann sich versionsübergreifend ändern.

Um langfristig stabilen Code zu schreiben, müssen Sie es vermeiden, Vermutungen über die Tiefe und Reihenfolge bestimmter Ergebnisse anzustellen.

Das Kategorienlayout ist darauf ausgelegt, hinsichtlich der eindeutigen Typnamen `CompilerResult`, `CompareResult`, `DownloadResult`, `UploadResult` langfristig stabil zu sein. Es gibt auch eine neue Ergebniskategorie: `UploadResult`. Inhalt, Hierarchie und Reihenfolge folgen der Anzeige auf der TIA Portal-Benutzeroberfläche des aktuell ausgeführten oder installierten TIA Portals.

Was ist neu in TIA Portal Openness?

Kompatibilität und Langzeitstabilität

- "Siemens.Engineering.dll"-Assemblies
Da die "Siemens.Engineering.dll"-Assemblies von V14 SP1, V15, V15.1 und V16 im Lieferumfang enthalten sind, laufen Anwendungen, die auf V14 SP1, V15 und V15.1 basieren, auch in V16 ohne Änderung. Um die Funktionen von V16 nutzen zu können, müssen Sie die DLL von V16 integrieren und die Anwendung neu übersetzen.
Die "Siemens.Engineering.dll"-Assemblies befinden sich im Installationsverzeichnis unter "PublicAPI[version]". Beispielsweise ist die dll-Datei für V14 SP1 unter "C:\Programme\Siemens\Automation\Portal V*\PublicAPI\V14 SP1\Siemens.Engineering.dll" zu finden.
- Exportieren von SIMATIC-ML-Dateien
Durch die "Siemens.Engineering.dll"-Assemblies von V14 SP1, V15, V15.1 und V16 werden SIMATIC-ML-Dateien von TIA-Portal-Version V16 erzeugt.
- Importieren von SIMATIC ML-Dateien
Jede Version der "Siemens.Engineering.dll"-Assemblies kann SIMATIC-ML-Dateien aus der gleichen Version und aus jeder unterstützten Version des vorherigen Release importieren. Beispiel: SIMATIC-ML-Dateien von V15 können mit den "Siemens.Engineering.dll"-Assemblies von V16 importiert werden. SIMATIC-ML-Dateien von V16 können mit den "Siemens.Engineering.dll"-Assemblies von V15 importiert werden.

Hinweis

Die Versionsnummer V* bezieht sich auf die installierte Version der TIA Portal Openness API.

Weitere Informationen zu Änderungen am Objektmodell finden Sie unter "Größere Änderungen in TIA Portal Openness V16".

Neue Funktionen

Die folgenden neuen Funktionen und Innovationen sind in TIA Portal Openness V16 verfügbar. Weitere Einzelheiten zu den verschiedenen Themen finden Sie in den einzelnen Abschnitten der Produktdokumentation.

- Anwenderspezifische Merker zur Identifikation von Geräten und Modulen
- Abfragen von Online-Prüfsummen von PLCs für Hardware und Software
- Unterstützung von Software Units
- Unterstützung des Exports/Imports von technologischen Objekten
- Unterstützung der Versionskontrollschnittstelle (VCI)
- Erweiterungen für CAx-Exporte und -Importe bezüglich
Unterstützung der APC AR V1.1
Verwendung von Objekten aus Bibliotheken
Unterstützung von Grundgeräten ET200SP

- Erweiterung für die Unterstützung der Hardware-Konfiguration bezüglich
OPC UA Server-Konfiguration und Benutzerverwaltung
Verwaltung von Zertifikaten
Webserver-Konfiguration und Benutzerverwaltung
Beobachtungstabellen für Webserver und Display
- Unterstützung der Konfiguration von WinCC Unified
- Unterstützung von SiVArc in Bezug auf
Erstellung und Änderung von SiVArc-Regeln
- Unterstützung der Sicherheitstechnik, auch wenn für das Sicherheitsprogramm ein
Passwort eingerichtet ist

Grundlagen

4.1 Voraussetzungen für TIA Portal Openness

Voraussetzungen für den Einsatz von TIA Openness-Anwendungen

- Auf dem PC ist ein auf dem TIA Portal basierendes Produkt installiert, zum Beispiel "STEP 7 Professional" oder "WinCC Professional".
- "TIA Portal Openness" ist auf dem PC installiert.
Siehe TIA Portal Openness installieren (Seite 30)

Unterstützte Windows-Betriebssysteme

Die folgende Tabelle zeigt, welche Kombinationen aus Betriebssystem Windows, TIA Portal und Benutzeranwendung miteinander kompatibel sind:

Betriebssystem Windows	TIA Portal	Benutzeranwendung
64-Bit	64-Bit	32-Bit, 64-Bit und beliebige CPU

Voraussetzungen für die Programmierung von TIA Portal Openness-Anwendungen

- Microsoft Visual Studio 2015 Update 1 oder höher mit .Net Framework SDK 4.6.2 und dem Windows Classic Desktop package

Notwendige Kenntnisse des Benutzers

- Kenntnisse als Systemtechniker
- Fortgeschrittene Kenntnisse in Microsoft Visual Studio 2015 Update 1 oder höher mit .Net Framework SDK 4.6.2
- Fortgeschrittene Kenntnisse in C# / VB.net und .Net Framework
- Benutzerkenntnisse des TIA Portals

TIA Portal Openness-Remoting-Kanäle

Die TIA Portal Openness-Remoting-Kanäle sind als Typ IpcChannel registriert, wobei der Parameter "ensureSecurity" auf "false" gesetzt ist.

Hinweis

Vermeiden Sie, einen weiteren IpcChannel mit dem Parameterwert "ensureSecurity" ungleich "false" und einer Priorität höher als oder gleich "1" zu registrieren.

Für den IpcChannel sind die folgenden Attribute festgelegt:

Attribut	Einstellungen
"name" und "portName"	Auf "\${Process.Name}_{Process.Id}" oder "\${Process.Name}_{Process.Id}_{AppDomain.Id}" gesetzt, wenn in einer anderen als der Standard-AppDomain der Anwendung registriert.
"priority"	Auf den Standardwert "1" gesetzt.
"typeFilterLevel"	Auf "Voll" gesetzt.
"authorizedGroup"	Auf die Zeichenkette des NT-Kontowerts für das integrierte Benutzerkonto (d.h. jeder) gesetzt.

Siehe auch

Benutzer der Benutzergruppe "Siemens TIA Openness" hinzufügen (Seite 31)

4.2 Installation

4.2.1 TIA Portal Openness installieren

Einführung

Installiert wird "TIA Portal Openness" über das Kontrollkästchen für TIA Portal Openness (unter Optionen) bei der Installation des TIA Portals.

Voraussetzungen

- Hardware und Software des Programmiergeräts oder PCs erfüllen die Systemvoraussetzungen.
- Sie haben Administratorrechte.
- Laufende Programme wurden geschlossen.
- Die Autorun-Funktion ist deaktiviert.
- WinCC und/oder STEP 7 ist/sind installiert.
- Die Versionsnummer von "TIA Portal Openness" entspricht der Versionsnummer von WinCC und STEP 7.

Hinweis

Wenn bereits eine Vorgängerversion von TIA Portal Openness installiert ist, wird die aktuelle Version parallel installiert.

Vorgehensweise

Zum Installieren von TIA Portal Openness stellen Sie sicher, dass das Kontrollkästchen für TIA Portal Openness während der Installation des TIA Portals ausgewählt ist. Befolgen Sie die nachstehenden Schritte, um die TIA Openness-Installation zu prüfen.

1. Wählen Sie im Menü "Konfiguration" den Ordner "Optionen" aus.
2. Aktivieren Sie das Kontrollkästchen für TIA Portal Openness.
3. Klicken Sie auf "Next" und wählen Sie die erforderliche Option.

Befolgen Sie den Installationsvorgang von TIA Portal, um die Installation von TIA Portal Openness abzuschließen.

Ergebnis

TIA Portal Openness ist auf dem PC installiert. Darüber hinaus wird die lokale Benutzergruppe "Siemens TIA Openness" erzeugt.

Hinweis

Sie haben mit dem Add-on-Paket "TIA Portal Openness" weiterhin keinen Zugriff auf das TIA Portal. Sie müssen Mitglied der Benutzergruppe "Siemens TIA Openness" sein (siehe Benutzer der Benutzergruppe "Siemens TIA Openness" hinzufügen (Seite 31)).

4.2.2 Benutzer der Benutzergruppe "Siemens TIA Openness" hinzufügen

Einleitung

Wenn Sie TIA Portal Openness auf dem PC installieren, wird die Benutzergruppe "Siemens TIA Openness" automatisch erzeugt.

Bei jedem Zugriff auf das TIA Portal mit Ihrer TIA Portal Openness-Anwendung prüft das TIA Portal, ob Sie Mitglied der Benutzergruppe "Siemens TIA Openness" sind, entweder direkt oder indirekt über eine andere Benutzergruppe. Wenn Sie Mitglied der Benutzergruppe "Siemens TIA Openness" sind, startet die TIA Portal Openness-Anwendung und stellt eine Verbindung zum TIA Portal her.

Vorgehensweise

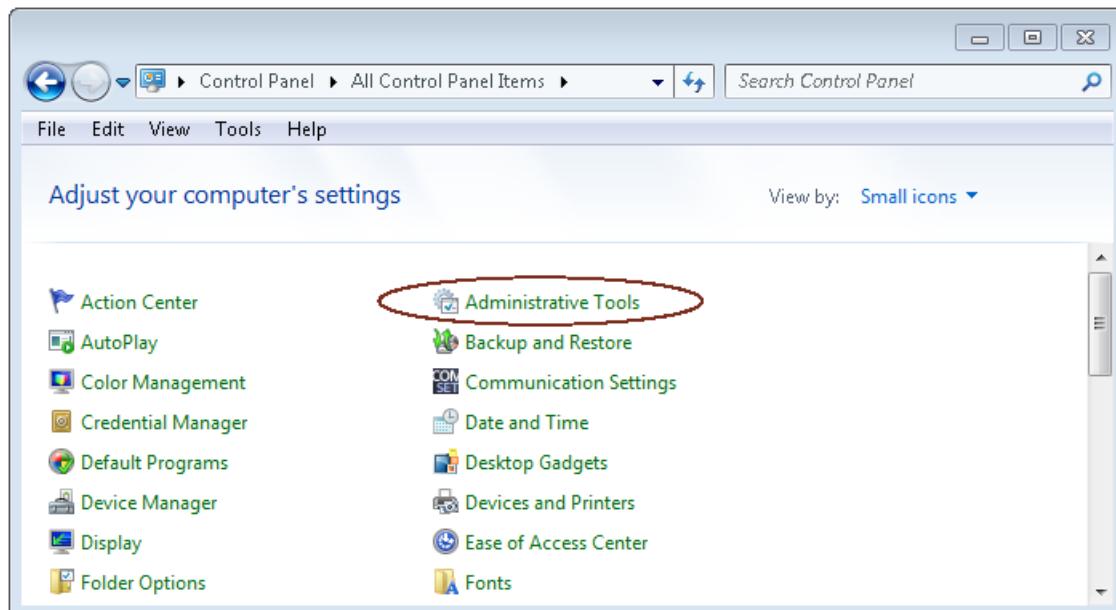
Wenn Sie der Benutzergruppe "Siemens TIA Openness" einen Benutzer hinzufügen möchten, müssen Sie auf Anwendungen aus Ihrem Betriebssystem zurückgreifen. Das TIA Portal unterstützt diesen Vorgang nicht.

Hinweis

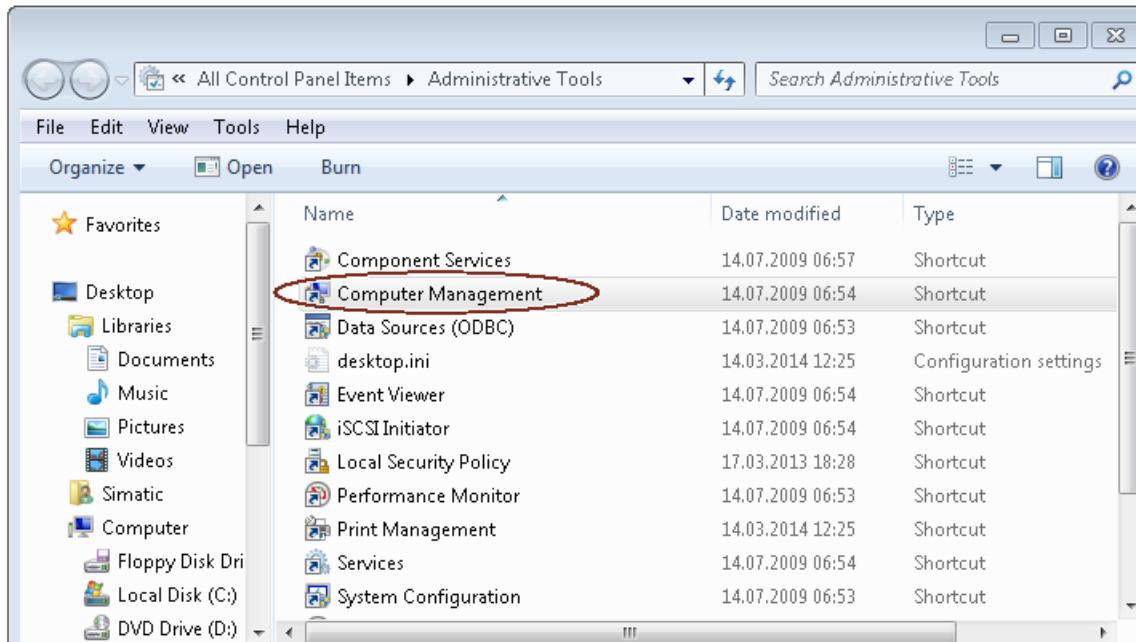
Je nach Konfiguration Ihrer Domain oder Ihres Rechners müssen Sie sich möglicherweise mit Administratorrechten anmelden, um die Benutzergruppe zu erweitern.

Im Betriebssystem Windows 7 (Spracheinstellung Englisch) gehen Sie zum Beispiel wie folgt vor, um einer Benutzergruppe einen Benutzer hinzuzufügen:

1. Wählen Sie "Start" > "Control Panel" aus.
2. Doppelklicken Sie im Control Panel auf "Administrative Tools".



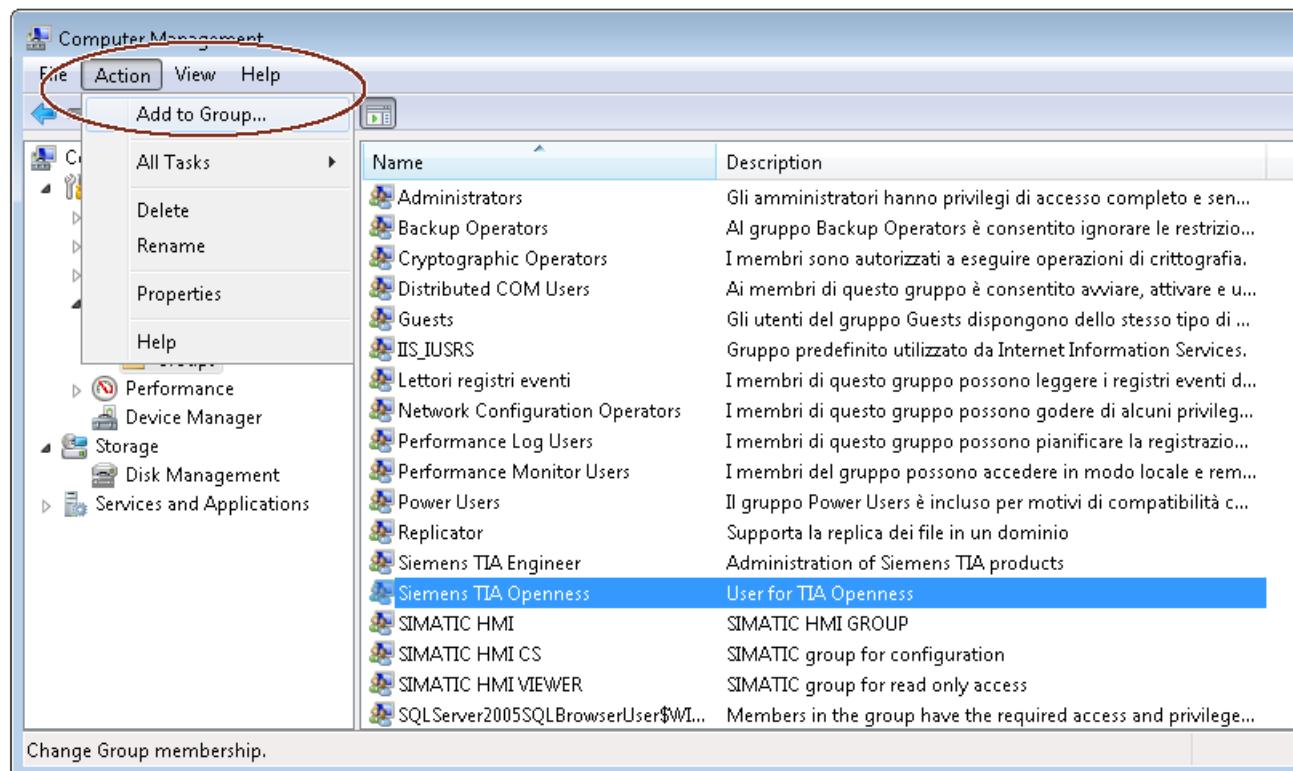
3. Klicken Sie auf "Computer Management", um den gleichnamigen Konfigurationsdialog zu öffnen.



4. Wählen Sie "Local Users and Groups > Groups" aus, um alle erzeugten Benutzergruppen anzuzeigen.
 5. Markieren Sie den Eintrag "Siemens TIA Openness" in der Liste der Benutzergruppen im rechten Fenster.

The screenshot shows the 'Computer Management' window with the title bar 'Computer Management'. The left navigation pane shows 'Computer Management (Local)' with nodes for System Tools, Task Scheduler, Event Viewer, Shared Folders, Local Users and Groups (with sub-nodes for Users and Groups), Performance, Device Manager, Storage, Disk Management, and Services and Applications. The 'Groups' node under Local Users and Groups is expanded. The right pane displays a table with columns 'Name' and 'Description'. The table lists several groups: Administrators, Backup Operators, Cryptographic Operators, Distributed COM Users, Guests, IIS_IUSRS, Lettori registri eventi, Network Configuration Operators, Performance Log Users, Performance Monitor Users, Power Users, Replicator, Siemens TIA Engineer, Siemens TIA Openness, SIMATIC HMI, SIMATIC HMI CS, and SIMATIC HMI VIEWER. The 'Siemens TIA Openness' group is circled with a red oval.

6. Wählen Sie den Menübefehl "Action > Add to Group..." aus.

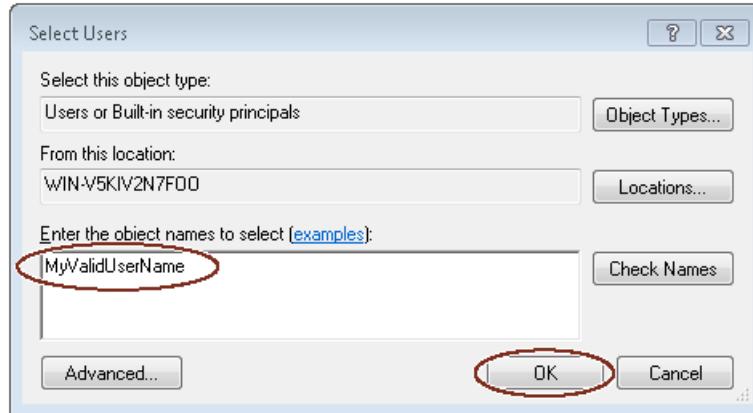


Der Attributedialog der Benutzergruppe öffnet sich:



7. Klicken Sie auf "Add".

Der sich daraufhin öffnende Auswahldialog zeigt die auswählbaren Benutzer an:



8. Geben Sie einen gültigen Benutzernamen in das Eingabefeld ein.

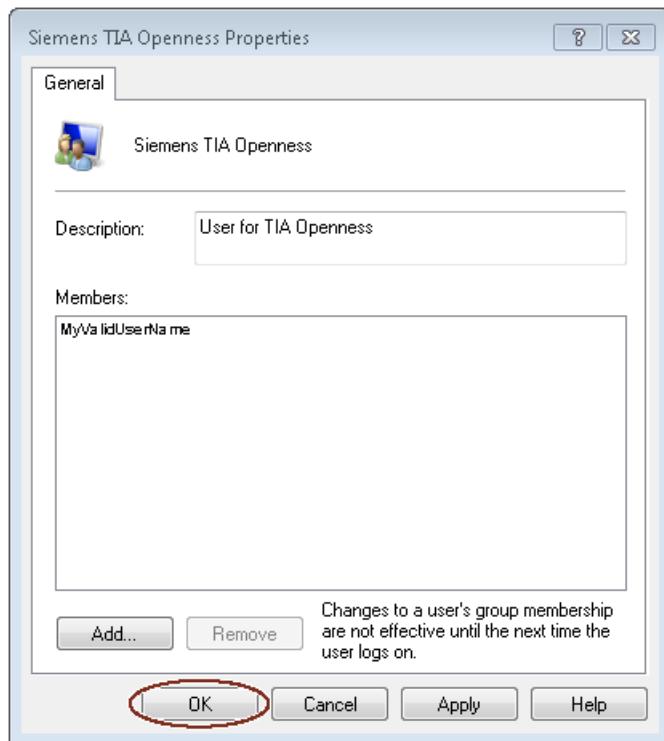
Hinweis

Klicken Sie auf "Check Names", um zu prüfen, ob der Benutzer ein gültiges Benutzerkonto für diese Domain oder diesen Rechner hat.

Im Feld "From this location" wird der Name der Domain oder des Rechners für den eingegebenen Benutzernamen angezeigt. Weitere Informationen hierzu erhalten Sie von Ihrem Systemadministrator.

9. Bestätigen Sie Ihre Auswahl mit "OK".

Der neue Benutzer wird jetzt im Attributedialog der Benutzergruppe angezeigt.



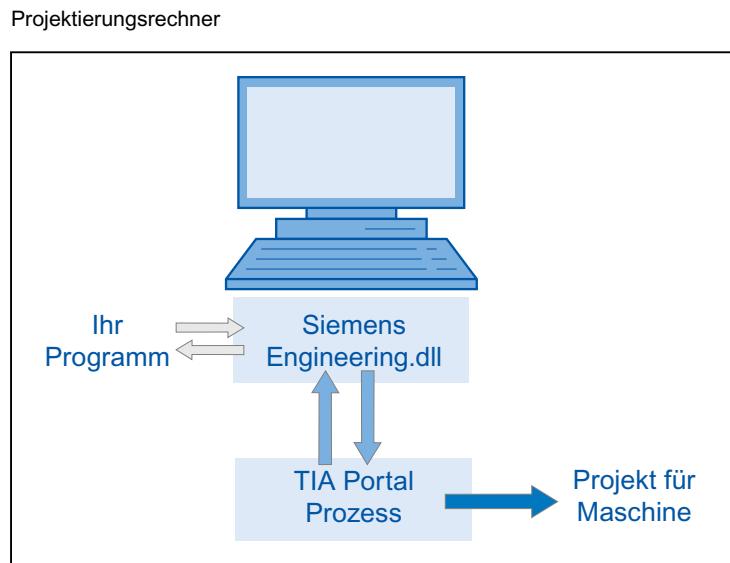
Zusätzliche Benutzer registrieren Sie durch Klicken auf die Schaltfläche "Add".

10. Klicken Sie auf "OK", um diesen Vorgang abzuschließen.

11. Melden Sie sich neu beim PC an, damit die Änderungen wirksam werden.

4.2.3 Auf das TIA Portal zugreifen

Übersicht



Vorgehensweise

1. Um auf das TIA Portal zuzugreifen und zu starten, richten Sie die Entwicklungsumgebung ein.
2. Um das Portal zu starten, instanziieren Sie in Ihrem Programm das Objekt der Portalanwendung.
3. Suchen Sie das gewünschte Projekt und öffnen Sie es
4. Greifen Sie auf die Projektdaten zu.
5. Schließen Sie das Projekt und beenden Sie das TIA Portal.

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Verbindung zum TIA Portal beenden \(Seite 88\)](#)

4.3 Aufgaben von Openness

4.3.1 Einsatzmöglichkeiten

Einleitung

TIA Portal Openness bietet Ihnen verschiedene Möglichkeiten für den Zugriff auf das TIA Portal und eine Auswahl von Funktionen für definierte Aufgaben.

Über die Schnittstelle TIA Portal Openness API greifen Sie auf folgende Bereiche des TIA Portals zu:

- Portaldaten
- Projektdaten
- PLC-Daten
- HMI-Daten
- Antriebsdaten

Hinweis

Sie dürfen die Schnittstelle TIA Portal Openness API nicht dafür nutzen, Prüfungen durchzuführen oder Daten für die Abnahme/Freigabe eines fehlersicheren Systems zu erzeugen. Die Abnahme/Freigabe darf nur mit einem Sicherheitsvordruck unter Verwendung des Add-on-Pakets STEP 7 Safety oder mit dem Funktionstest durchgeführt werden. Die TIA Portal Openness API ist kein Ersatz.

Auf das TIA Portal zugreifen

TIA Portal Openness bietet verschiedene Möglichkeiten für den Zugriff auf das TIA Portal. Sie erzeugen eine externe Instanz des TIA Portals im Prozess mit oder ohne Benutzeroberfläche. Sie können auch gleichzeitig auf laufende Prozesse des TIA Portals zugreifen.

Auf Projekte und Projektdaten zugreifen

Beim Zugriff auf Projekte und Projektdaten verwenden Sie TIA Portal Openness hauptsächlich für die folgenden Aufgaben:

- Projekt öffnen, schließen und speichern
- Objekte enumerieren und abfragen
- Objekte erzeugen
- Objekte löschen

4.3.2 Export/Import

Einleitung

TIA Portal Openess unterstützt den Import und Export von Projektdaten mittels XML-Dateien. Die Funktion Importieren/Exportieren unterstützt die externe Konfiguration vorhandener Engineering-Daten. Sie verwenden diese Funktion, um den Engineering-Prozess effektiv und fehlerfrei zu machen.

Verwendung

Sie nutzen die Funktion Importieren/Exportieren für die folgenden Zwecke:

- Datenaustausch
- Kopieren von Teilen eines Projekts
- Externe Verarbeitung von Konfigurationsdaten, zum Beispiel für Massendatenoperationen mit Suchen und Ersetzen
- Externe Verarbeitung von Konfigurationsdaten für neue Projekte auf der Grundlage bestehender Konfigurationen
- Importieren extern erzeugter Konfigurationsdaten, zum Beispiel Textlisten und Variablen
- Bereitstellen von Projektdaten für externe Anwendungen

4.4 Objektliste

Einleitung

In den folgenden Tabellen werden die verfügbaren Objekte bis einschließlich Runtime Advanced aufgeführt und es wird angegeben, ob diese Objekte von TIA Portal Openess unterstützt werden.

TIA Portal Openess in WinCC unterstützt weder die Visualisierungssoftware Runtime Professional noch Geräteproxydateien.

Objekte

Je nach dem von Ihnen genutzten Bediengerät können Sie die folgenden Projektdaten verwenden:

Tabelle 4-1 Bilder

Objekt	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Bild	Ja	Ja	Ja	Ja
Globales Bild	Ja	Ja	Ja	Ja
Vorlagen	Ja	Ja	Ja	Ja
Permanentfenster	Nein	Ja	Ja	Ja

4.4 Objektliste

Objekt	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Pop-up-Bild	Nein	Ja	Ja	Ja
Slide-in-Bild	Nein	Ja	Ja	Ja

Tabelle 4-2 Bildobjekte

Objekt	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Linie	Ja	Ja	Ja	Ja
Polygonzug	Nein	Ja	Ja	Ja
Polygon	Nein	Ja	Ja	Ja
Ellipse	Ja	Ja	Ja	Ja
Ellipsensegment	Nein	Nein	Nein	Nein
Kreissegment	Nein	Nein	Nein	Nein
Ellipsenbogen	Nein	Nein	Nein	Nein
Kamera-Anzeige	Nein	Nein	Nein	Nein
Kreisbogen	Nein	Nein	Nein	Nein
Kreis	Ja	Ja	Ja	Ja
PDF-Anzeige	Nein	Nein	Nein	Nein
Rechteck	Ja	Ja	Ja	Ja
Verbinder	Nein	Nein	Nein	Nein
Textfeld	Ja	Ja	Ja	Ja
Grafikanzeige	Ja	Ja	Ja	Ja
Rohr	Nein	Nein	Nein	Nein
Doppel-T-Stück	Nein	Nein	Nein	Nein
T-Stück	Nein	Nein	Nein	Nein
Rohrkrümmer	Nein	Nein	Nein	Nein
E/A-Feld	Ja	Ja	Ja	Ja
Datum/Uhrzeit-Feld	Ja	Ja	Ja	Ja
Grafisches E/A-Feld	Ja	Ja	Ja	Ja
Editierbares Textfeld	Nein	Nein	Nein	Nein
Listenfeld	Nein	Nein	Nein	Nein
Kombinationsfeld	Nein	Nein	Nein	Nein
Schaltfläche	Ja	Ja	Ja	Ja
Rundschaltfläche	Nein	Nein	Nein	Nein
Leuchtdrucktaster	Nein	Nein	Ja	Nein
Schalter	Ja	Ja	Ja	Ja
Symbolisches E/A-Feld	Ja	Ja	Ja	Ja
Schlüsselschalter	Nein	Nein	Ja	Nein
Balken	Ja	Ja	Ja	Ja
Symbolbibliothek	Nein	Ja	Ja	Ja
Schieberegler	Nein	Ja	Ja	Ja
Bildlaufleiste	Nein	Nein	Nein	Nein

Objekt	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Kontrollkästchen	Nein	Nein	Nein	Nein
Optionsschaltfläche	Nein	Nein	Nein	Nein
Zeigerinstrument	Nein	Ja	Ja	Ja
Uhr	Nein	Ja	Ja	Ja
Speicherplatzanzeige	Nein	Nein	Nein	Nein
Funktionstasten	Ja	Ja	Ja	Ja
Bildbausteininstanzen	Nein	Ja	Ja	Ja
Bildfenster	Nein	Nein	Nein	Nein
Benutzeranzeige	Ja	Ja	Ja	Ja
HTML-Browser	Nein	Nein	Nein	Nein
Druckauftrag/Skriptdiagnose	Nein	Nein	Nein	Nein
Rezepturanzige	Nein	Nein	Nein	Nein
Meldeanzeige	Nein	Nein	Nein	Nein
Meldeindikator	Nein	Nein	Nein	Nein
Meldefenster	Nein	Nein	Nein	Nein
Anzeige Kriterienanalyse	Nein	Ja ¹⁾	Ja	Ja
ProDiag-Übersicht	Nein	Ja ¹⁾	Ja	Ja
GRAPH-Übersicht	Nein	Ja ¹⁾	Ja	Ja
PLC-Codeanzeige	Nein	Ja ¹⁾	Ja	Ja
f(x)-Kurvenanzeige	Nein	Nein	Nein	Nein
f(t)-Kurvenanzeige	Nein	Nein	Nein	Nein
Tabellenanzeige	Nein	Nein	Nein	Nein
Wertetabelle	Nein	Nein	Nein	Nein
Media Player	Nein	Nein	Nein	Nein
Kanaldiagnose	Nein	Nein	Nein	Nein
WLAN-Empfang	Nein	Nein	Nein	Nein
Zonen-Name	Nein	Nein	Nein	Nein
Zonen-Signal	Nein	Nein	Nein	Nein
Wirkbereichs-Name	Nein	Nein	Nein	Nein
Wirkbereichs-Name (RFID)	Nein	Nein	Nein	Nein
Wirkbereichs-Signal	Nein	Nein	Nein	Nein
Ladezustand	Nein	Nein	Nein	Nein
Handrad	Nein	Nein	Ja	Nein
Hilfe-Indikator	Nein	Nein	Nein	Nein
Sm@rtClient-Anzeige	Nein	Nein	Nein	Nein
Status/Steuern	Nein	Nein	Nein	Nein

Grundlagen

4.4 Objektliste

Objekt	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
System-Diagnoseanzeige	Nein	Nein	Nein	Nein
System-Diagnosefensster	Nein	Nein	Nein	Nein

1) Nur Mobile Panels mit einer höheren Geräteversion als 12.0.0.0 unterstützen dieses Bildschirmobjekt

Tabelle 4-3 Dynamisch

Objekt	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Anzeige	Ja	Ja	Ja	Ja
Bedienbarkeit	Nein	Ja	Ja	Ja
Sichtbarkeit	Ja	Ja	Ja	Ja
Bewegungen	Ja	Ja	Ja	Ja

Tabelle 4-4 Zusätzliche Objekte

Objekt	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Gruppen	Ja	Ja	Ja	Ja
Softkeys	Ja	Ja	Ja	Ja
Zyklen	Ja	Ja	Ja	Ja
VB-Skripte	Nein	Ja	Ja	Ja
Funktionslisten	Ja	Ja	Ja	Ja
Grafiksammlung	Ja	Ja	Ja	Ja

Tabelle 4-5 Variablen

Objekt	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Multiplex-Variablen	Ja	Ja	Ja	Ja
Arrays	Ja	Ja	Ja	Ja
Anwenderdatentypen	Ja	Ja	Ja	Ja
Intern	Nein	Ja	Ja	Ja
Verwendungsstellen von elementaren Datentypen	Ja	Ja	Ja	Ja
Verwendungsstellen von Anwenderdatentypen	Ja	Ja	Ja	Ja
Verwendungsstellen von Arrays	Ja	Ja	Ja	Ja

Zusätzlich unterstützt TIA Portal Openess alle Wertebereiche, die von den Kommunikationstreibern unterstützt werden.

Verbindungen

4.5 Attribute von Bausteinen, DBs und UDTs festlegen

TIA Portal Openness unterstützt nicht integrierte Verbindungen, die auch von den jeweiligen Bediengeräten unterstützt werden. Weitere Informationen finden Sie in der Online-Hilfe für das TIA Portal unter „Prozesse visualisieren > Mit Steuerungen kommunizieren > Geräteabhängigkeit“.

Tabelle 4-6 Listen

Objekt	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Textlisten	Ja	Ja	Ja	Ja
Grafiklisten	Ja	Ja	Ja	Ja

Tabelle 4-7 Texte

Objekt	Basic Panels	Comfort Panels	Mobile Panels	RT Advanced
Mehrsprachige Texte	Ja	Ja	Ja	Ja
Formatierte Texte und deren Verwendungsstellen	Nein	Ja	Ja	Ja

4.5 Attribute von Bausteinen, DBs und UDTs festlegen

Einleitung

Es ist möglich, (allgemeine) Attribute von (in einer beliebigen Programmiersprache geschriebenen) Bausteinen, von DBs und von UDTs über die Openness API zu schreiben und zu lesen.

Technologieobjekte sind intern DBs, deshalb gilt diese Funktion auch für TOs.

Die gleichen Prüfregeln sind auch auf den Namen anzuwenden, und zwar mittels SetAttribute der Openness API für Bausteine, DBs (TOs) und UDTs, die auf die GUI angewendet werden.

Bei einer Regelverletzung wird eine entsprechende Anwenderausnahme zurückgegeben und der Attributwert wird nicht geändert.

Diese Erweiterung der API betrifft nicht den XML-Export/-Import.

Allgemeine Attribute

Legende

- X: Das Attribut ist relevant, Schreiben in XML, Zugriff über API möglich
- -: Das Attribut ist nicht zugänglich, das Attribut existiert nicht

4.5 Attribute von Bausteinen, DBs und UDTs festlegen

Attribute	Datentyp	Verfügbarkeit über die API		
		CodeBlock	DB	UDT
AutoNumber	Bool	X (XML); RW (API)	X (XML); RW (API)	-
CodeModifiedDate	DateTime	X (XML); R (API)	X (XML); R (API)	-
CompileDate	DateTime	X (XML); R (API)	X (XML); R (API)	-
CreationDate	DateTime	X (XML); R (API)	X (XML); R (API)	X (XML); R (API)
HeaderAuthor	String	X (XML); R (API)	X (XML); R (API)	-
HeaderFamily	String	X (XML); R (API)	X (XML); R (API)	-
HeaderName	String	X (XML); R (API)	X (XML); R (API)	-
HeaderVersion	System.Version	X (XML); R (API)	X (XML); R (API)	-
InstanceOfName	String	-	X (XML); R (API)	-
Schnittstelle	String	X (XML); - (API)	X (XML); - (API)	X (XML); - (API)
InterfaceModifiedDate	DateTime	X (XML); R (API)	X (XML); R (API)	X (XML); R (API)
IsConsistent	Bool	X (XML); R (API)	X (XML); R (API)	X (XML); R (API)
IsKnowHowProtected	Bool	X (XML); R (API)	X (XML); R (API)	X (XML); R (API)
MemoryLayout (bisher: Optimization)	MemoryLayout	X (XML); R (API)	X (XML); R (API)	-
ModifiedDate	DateTime	X (XML); R (API)	X (XML); R (API)	X (XML); R (API)
Name	String	X (XML); RW (API)	X (XML); RW (API)	X (XML); RW (API)
Number	Int32	X (XML); RW (API)	X (XML); RW (API)	-
ParameterModified	DateTime	X (XML); R (API)	X (XML); R (API)	-
ProgrammingLanguage	ProgrammingLanguage	X (XML); R (API)	X (XML); R (API)	-
SecondaryType	String	X (XML); R (API)	-	-
StructureModified	DateTime	X (XML); R (API)	X (XML); R (API)	-

4.6 Standard-Bibliotheken

Fügen Sie die folgenden Namensraumanweisungen am Beginn des jeweiligen Codebeispiels ein, um sicherzustellen, dass die Codebeispiele funktionieren:

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

4.7 Anmerkungen zur Leistung von TIA Portal Openness

Stammobjekt

In Importdateien können Sie mehrere Stammobjekte angeben.

Beispiel: Statt einer Textliste für jede XML-Datei können Sie mehrere Textlisten in einer XML-Datei erzeugen.

TIA Portal Openness-Funktionen

Der erste Aufruf einer TIA Portal Openness-Funktion kann länger dauern als nachfolgende Aufrufe der TIA Portal Openness-Funktion.

4.7 Anmerkungen zur Leistung von TIA Portal Openness

Beispiel: Wenn Sie nacheinander mehrere Exporte von Konfigurationsdaten durchführen, kann der erste Export länger dauern als die nachfolgenden Exporte.

Einführung

Einleitung

In TIA Portal Openness werden offene Schnittstellen für das Engineering mit dem TIA Portal beschrieben. Weitere Informationen über "TIA Portal Openness - Efficient generation of program code using code generators" finden Sie im YouTube-Kanal von SIEMENS.

Sie automatisieren das Engineering mit TIA Portal Openness, indem Sie das TIA Portal extern über ein von Ihnen erstelltes Programm steuern.

Mit TIA Portal Openness können Sie folgende Aktionen durchführen:

- Projektdaten erstellen
- Projekte und Projektdaten ändern
- Projektdaten löschen
- Projektdaten einlesen
- Projekte und Projektdaten anderen Anwendungen zur Verfügung stellen

Hinweis

Siemens ist nicht verantwortlich für die Kompatibilität von Daten und Informationen, die über diese Schnittstellen mit Fremdsoftware übertragen werden, und übernimmt dafür keine Gewähr.

Wir weisen ausdrücklich darauf hin, dass eine unsachgemäße Nutzung der Schnittstellen zu Datenverlust und zum Produktionsstillstand führen kann.

Hinweis

Die in dieser Dokumentation enthaltenen Codeausschnitte sind in C#-Syntax geschrieben.

Aufgrund der Kürze der verwendeten Codeausschnitte hat sich die Beschreibung der Fehlerbehandlung ebenfalls verkürzt.

Verwendung

Die TIA Portal Openness-Schnittstelle dient folgenden Zwecken:

- Projektdaten bereitstellen
- Auf den TIA Portal-Prozess zugreifen
- Projektdaten nutzen

Verwendung von Standardwerten aus dem Bereich der Automatisierungstechnik

- Durch Importieren von extern generierten Daten
- Durch Fernsteuerung des TIA Portals zum Generieren von Projekten

Bereitstellen von Projektdaten des TIA Portals für externe Anwendungen

- Durch Exportieren von Projektdaten

Sichern von Wettbewerbsvorteilen durch effizientes Engineering

- Sie brauchen vorhandene Engineering-Daten im TIA Portal nicht zu konfigurieren.
- Automatisierte Engineering-Prozesse ersetzen das manuelle Engineering.
- Niedrige Engineering-Kosten stärken die Anbieterposition gegenüber Mitbewerbern.

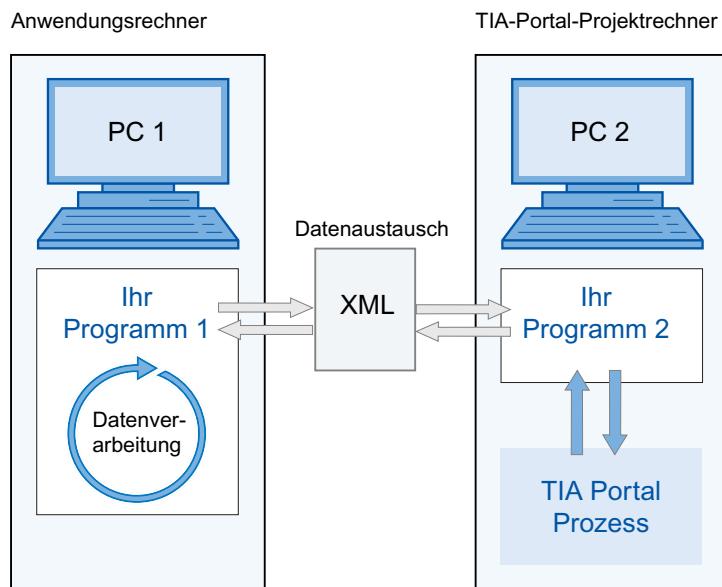
Gemeinsam an Projektdaten arbeiten

- Testroutinen und Massendatenverarbeitung können parallel zur laufenden Konfiguration ablaufen bzw. stattfinden.

Konfigurationen

Sie können mit zwei Varianten von TIA Portal Openness arbeiten:

Die Anwendung und das TIA Portal befinden sich auf unterschiedlichen Computern



- Der Datenaustausch erfolgt über XML-Dateien. Die XML-Dateien können von Ihren Programmen exportiert oder importiert werden.
- Die aus dem TIA Portal-Projekt auf PC2 exportierten Daten können auf PC1 geändert und wieder importiert werden.

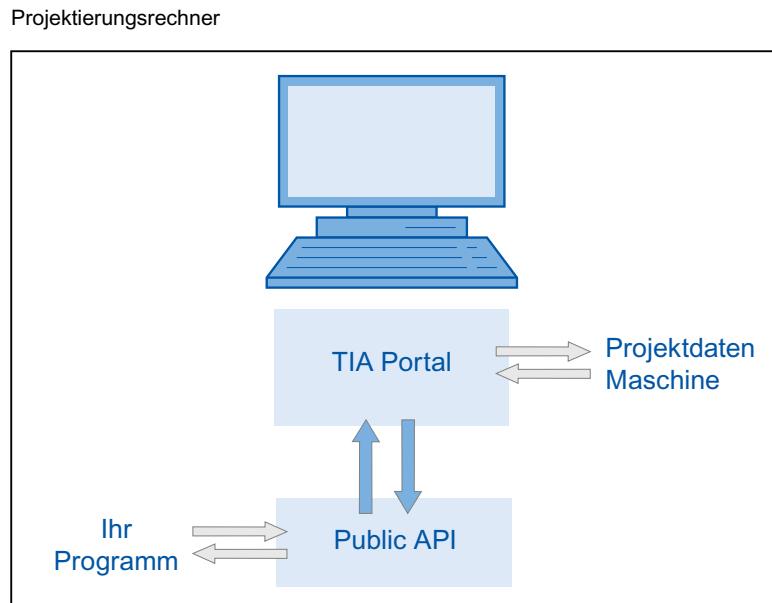
Hinweis

Sie müssen für PC2 ein ausführbares Programm „Ihr Programm 2“ entwickeln, beispielsweise „programm2.exe“. Das TIA Portal läuft mit diesem Programm im Hintergrund.

Import und Export der XML-Dateien erfolgt ausschließlich über die TIA Portal Openness API.

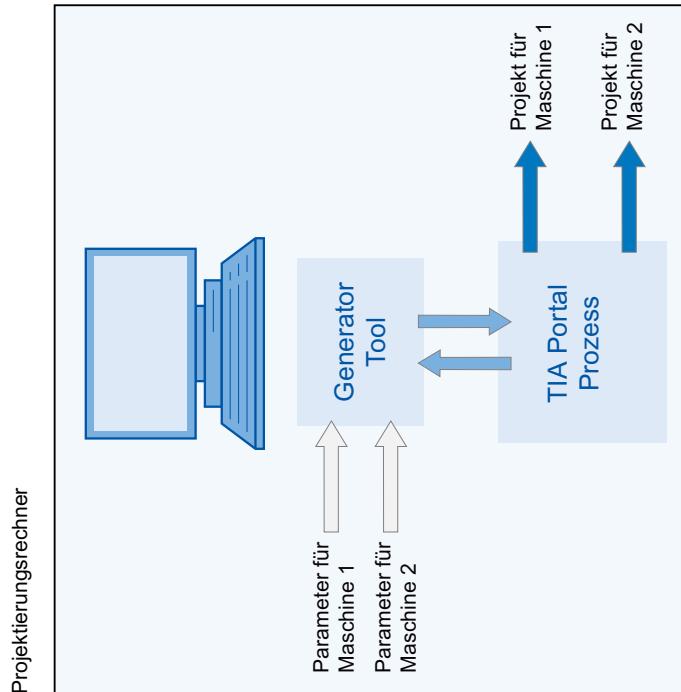
- Sie können ausgetauschte Dateien zu Verifizierungszwecken archivieren.
- Ausgetauschte Daten können an unterschiedlichen Standorten und zu unterschiedlichen Zeiten verarbeitet werden.

Die Anwendung und das TIA Portal befinden sich auf demselben Computer



- Ihr Programm ruft das TIA Portal entweder mit der oder ohne die Benutzeroberfläche auf. Ihr Programm öffnet, speichert und/oder schließt ein Projekt. Das Programm kann auch eine Verbindung zu einem laufenden TIA Portal herstellen.
- Sie können dann die TIA Portal-Funktionalität nutzen, um Projektdaten anzufordern, zu generieren und zu ändern oder um Import- und Exportvorgänge auszulösen.
- Die Daten werden unter der Kontrolle der TIA Portal-Verarbeitung erstellt und in den Projektdaten gespeichert.

Typischer Anwendungsbereich ist der modulare Maschinenbau



- Ein effizientes Automatisierungssystem ist auf ähnliche Maschinen anzuwenden.
- Ein Projekt ist im TIA Portal verfügbar, das die Komponenten aller Maschinenvarianten enthält.
- Das Generator-Tool steuert die Erstellung des Projekts für eine bestimmte Maschinenvariante.
- Das Generator-Tool ruft die Standardwerte ab, indem es die Parameter für die angeforderte Maschinenvariante einliest.
- Das Generator-Tool filtert die betreffenden Elemente aus dem TIA Portal-Gesamtprojekt heraus, ändert sie gegebenenfalls und generiert das angeforderte Maschinenprojekt.

TIA Portal Openness API

7.1 Einleitung

Übersicht

TIA Portal Openness unterstützt eine Auswahl von Funktionen für definierte Aufgaben, die Sie außerhalb des TIA Portals über TIA Portal Openness API aufrufen können.

Hinweis

Wenn bereits eine Vorgängerversion von TIA Portal Openness installiert ist, wird die aktuelle Version parallel installiert.

Die nachstehenden Abschnitte vermitteln Ihnen einen Überblick über die typischen Schritte der Programmierung. Sie erfahren, wie die einzelnen Codeabschnitte interagieren und wie sich die jeweiligen Funktionen zu einem vollständigen Programm verzahnen. Sie erhalten auch einen Überblick über die Codekomponenten, die der jeweiligen Aufgabe angepasst werden müssen.

Beispielprogramm

Die einzelnen Schritte der Programmierung werden mit der Funktion "API-Zugriff in einer Konsolenanwendung" als Beispiel erläutert. Sie als Benutzer integrieren die zur Verfügung gestellten Funktionen in diesen Programmcode und passen die jeweiligen Codekomponenten an diese Aufgabe an.

Funktionen

Im nachstehenden Abschnitt sind die Funktionen für definierte Aufgaben aufgeführt, die Sie mit TIA Portal Openness außerhalb des TIA Portals aufrufen können.

7.2 Programmierschritte

Übersicht

TIA Portal Openness erfordert die folgenden Schritte der Programmierung für den Zugriff mittels TIA Portal Openness API:

1. TIA Portal in der Entwicklungsumgebung bekannt machen
2. Programmzugriff auf das TIA Portal einrichten
3. Programmzugriff auf das TIA Portal aktivieren
4. TIA Portal veröffentlichen und starten

5. Projekt öffnen
6. Befehle ausführen
7. Projekt speichern und schließen
8. Verbindung zum TIA Portal beenden

Hinweis

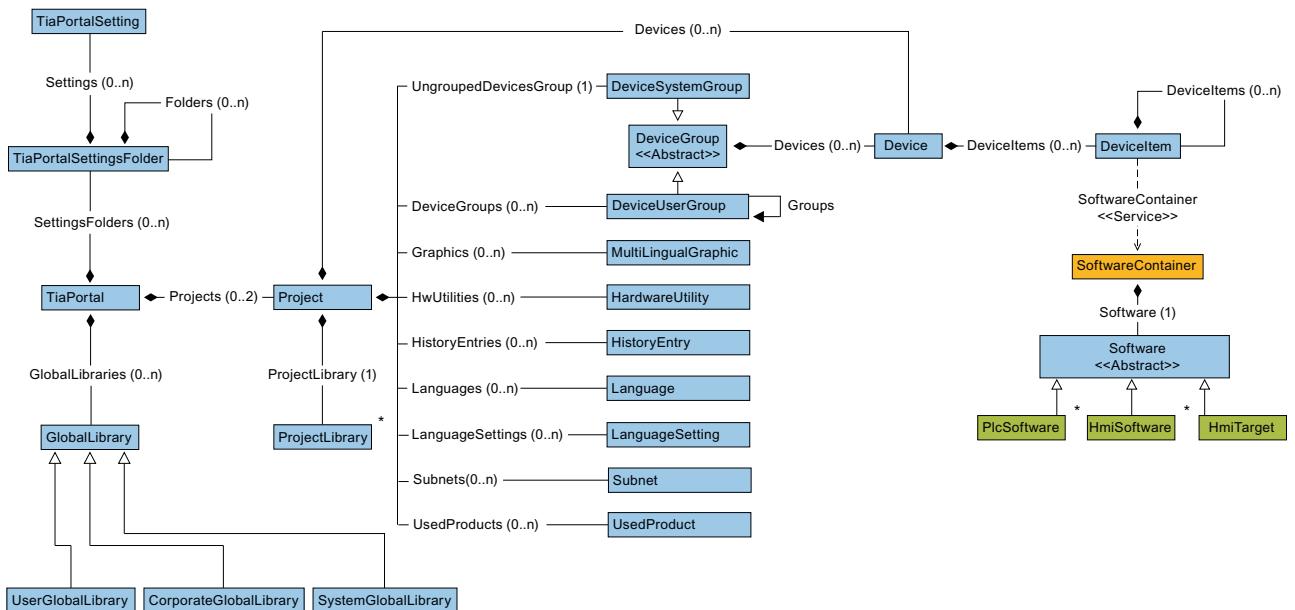
Zulässige Strings

In Strings im TIA Portal sind nur bestimmte Zeichen zulässig. Alle über die TIA Portal Openness-Anwendung an das TIA Portal übergebene Strings unterliegen diesen Regeln. Wenn Sie ein ungültiges Zeichen in einem Parameter übergeben, wird eine Ausnahme angezeigt.

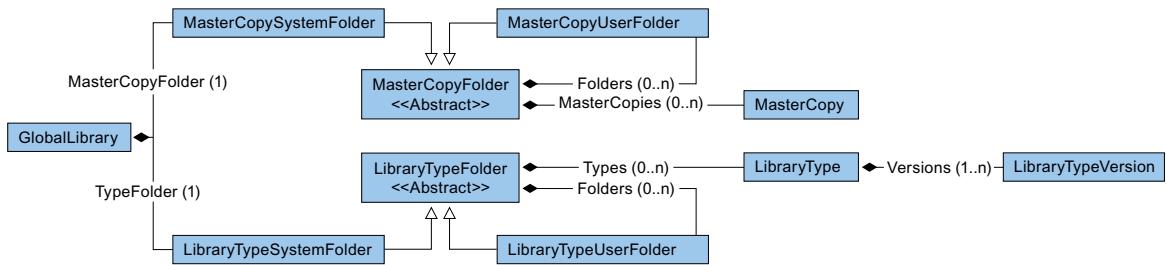
7.3 TIA Portal Openness-Objektmodell

Übersicht

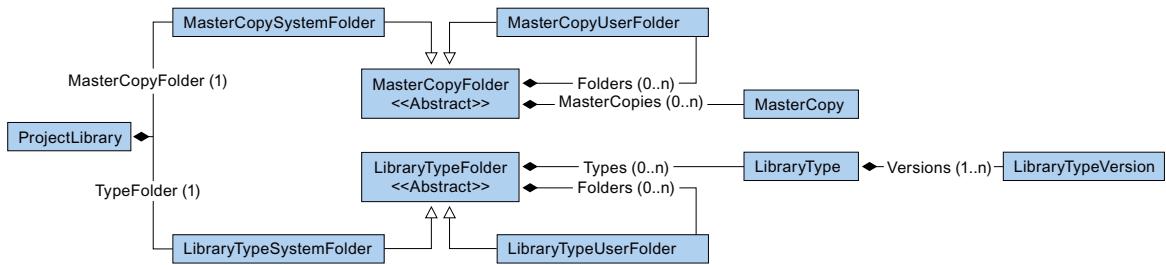
Das folgende Diagramm beschreibt die höchste Ebene des TIA Portal Openness-Objektmodells:



Das folgende Diagramm beschreibt die Objekte, die sich unter **GlobalLibrary** befinden.

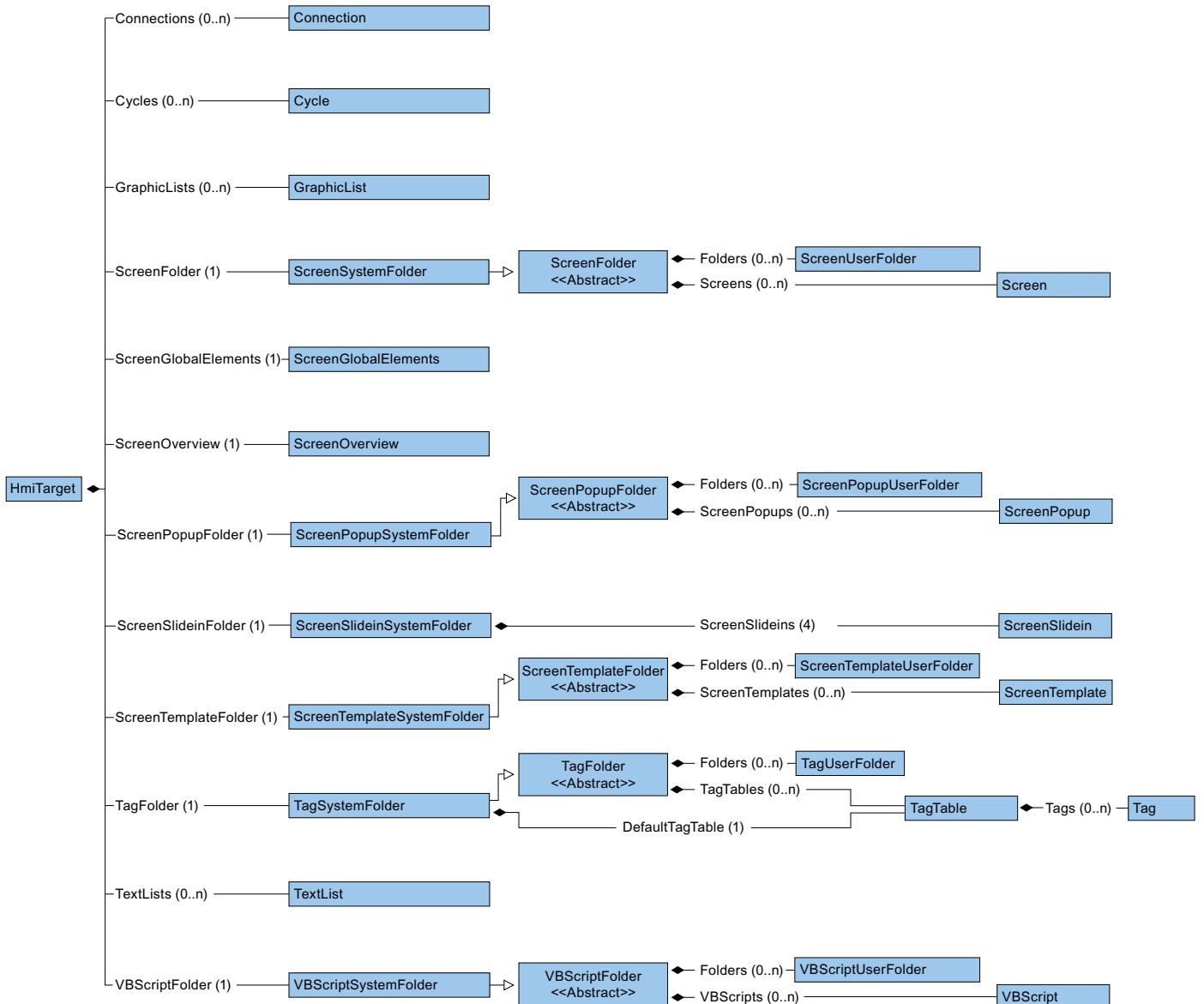


Das folgende Diagramm beschreibt die Objekte, die sich unter ProjectLibrary befinden.

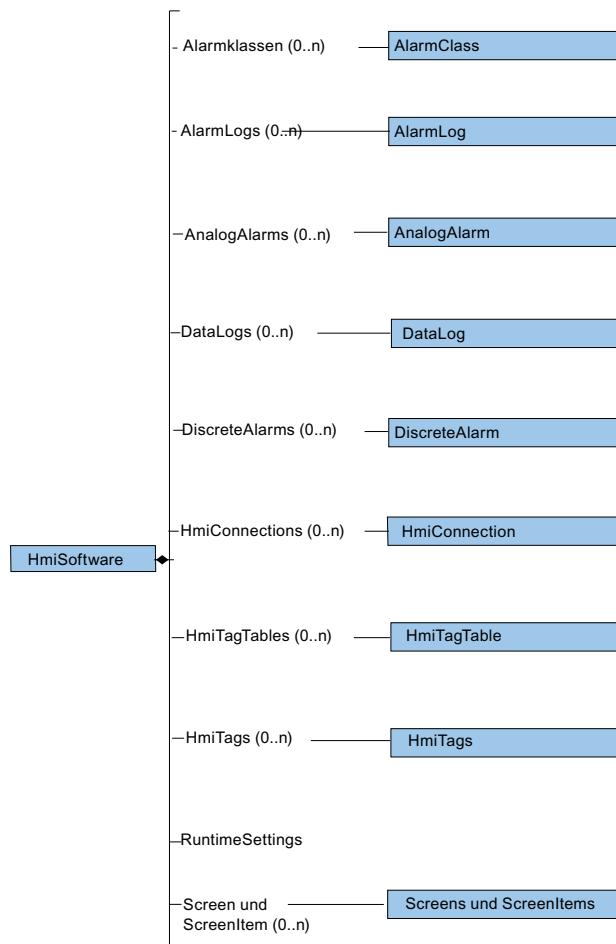


Das folgende Diagramm beschreibt die Objekte, die sich unter HmiTarget befinden.

7.3 TIA Portal Openness-Objektmodell

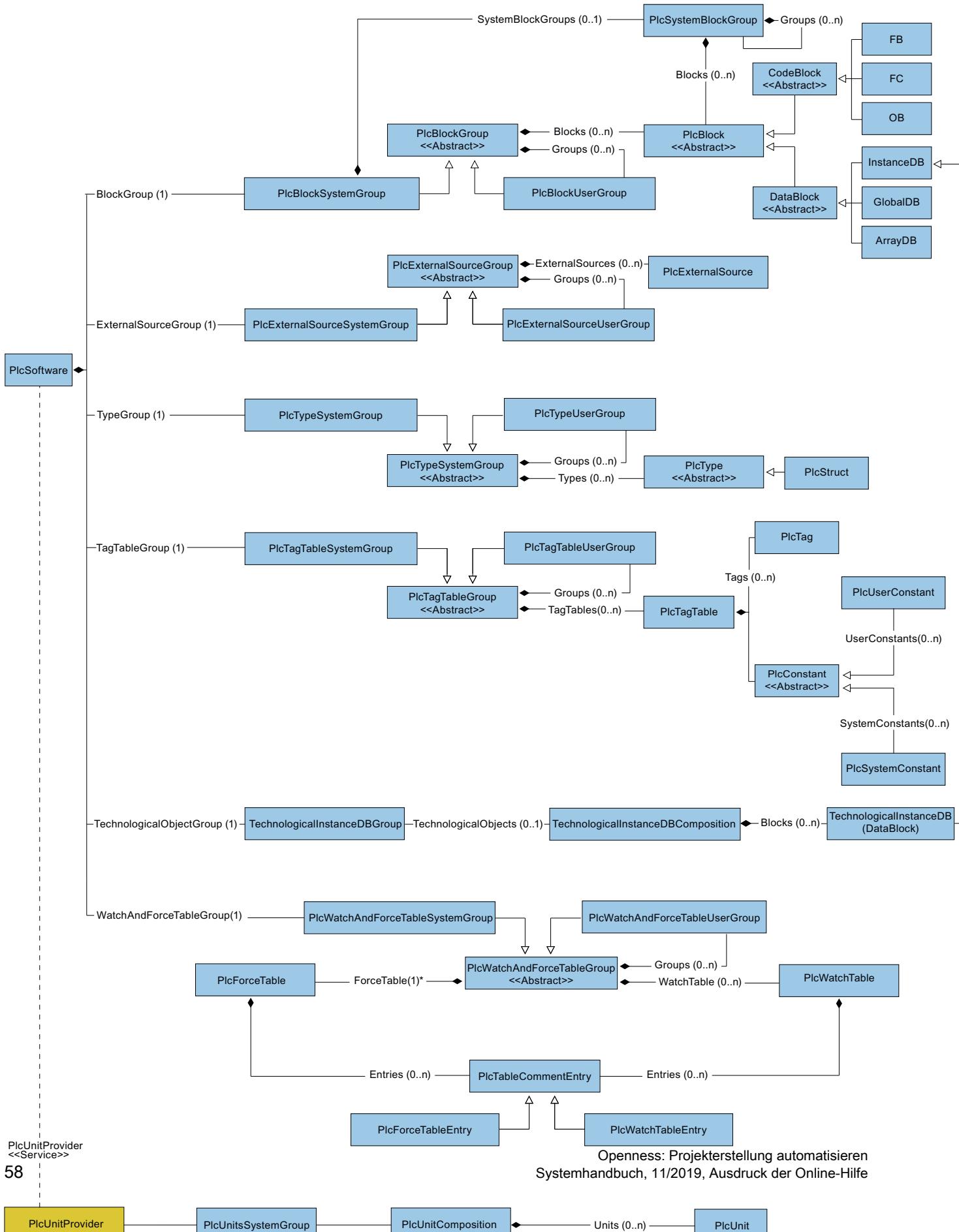


Das folgende Diagramm beschreibt die Objekte, die sich unter HmiSoftware befinden.



Das folgende Diagramm beschreibt die Objekte, die sich unter PlcSoftware befinden.

7.3 TIA Portal Openness-Objektmodell



Hinweis

* Die Forcetabelle muss sich in PlcWatch und ForceTableSystemGroup befinden

Auf Objekte in Listen zugreifen

Sie haben die folgenden Optionen zum Adressieren eines Objekts in einer Liste:

- Adressieren Sie über den Index. Die Zählung innerhalb der Listen beginnt mit 0.
- Verwenden Sie die Methode `Find`.

Verwenden Sie diese Methode zum Adressieren eines Objekts über seinen Namen. Sie können diese Methode für eine Zusammensetzung oder Liste verwenden. Die Methode `Find` ist nicht rekursiv.

Beispiel:

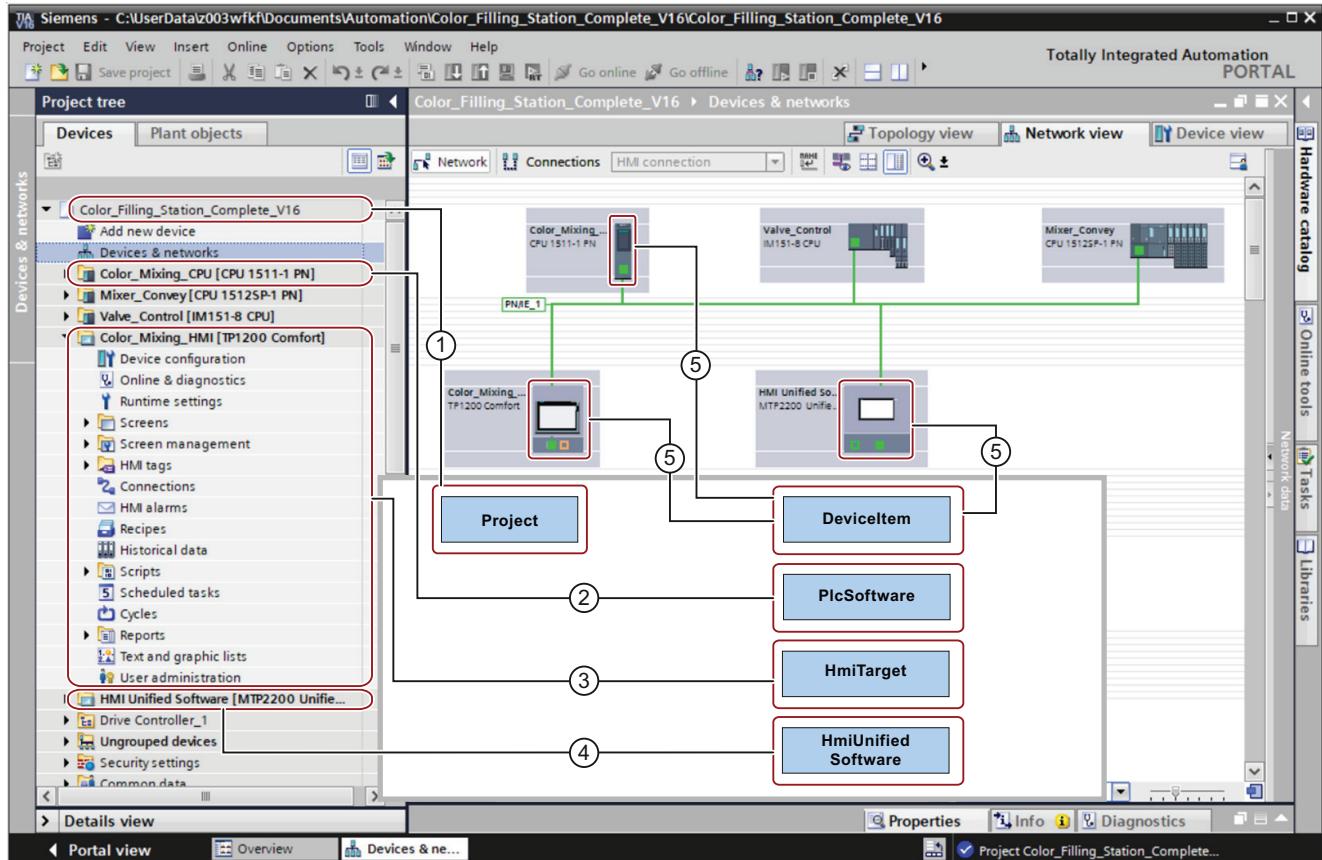
```
ScreenComposition screens = folder.Screens;  
Screen screen = screens.Find("myScreen");
```

- Verwenden Sie symbolische Namen.

7.3 TIA Portal Openness-Objektmodell

Beziehung zwischen TIA Portal und TIA Portal Openness-Objektmodell

Das Bild unten zeigt die Beziehung zwischen dem Objektmodell und einem Projekt im TIA Portal:

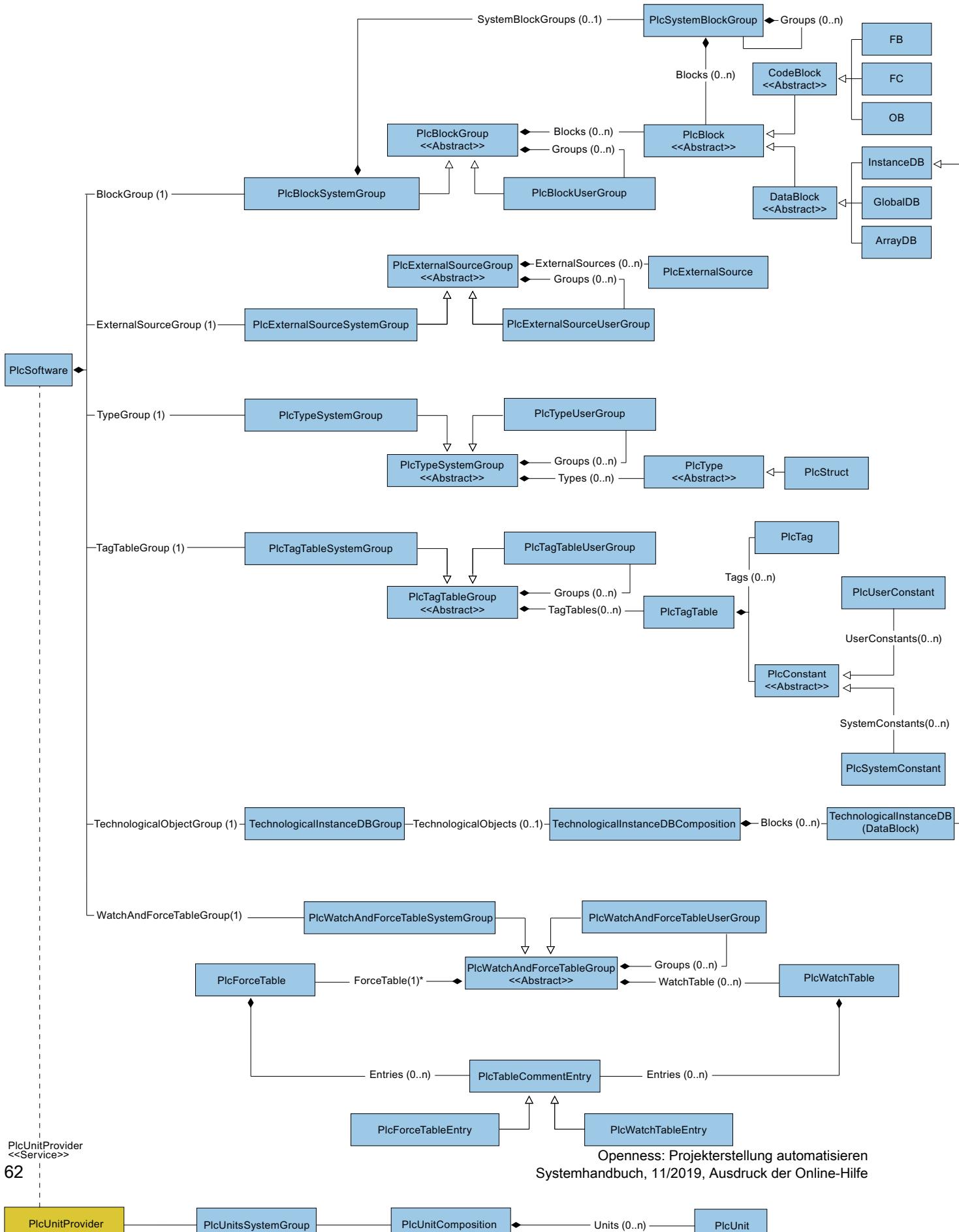


7.4 Bausteine und Typen des TIA Portal Openness-Objektmodells

Einleitung

Das folgende Diagramm beschreibt das Domainenmodell der PLCs, um einen Überblick über die aktuelle Modellierung in TIA Portal Openness zu geben.

7.4 Bausteine und Typen des TIA Portal Openness-Objektmodells



Hinweis

* Die Forcetabelle muss sich in PlcWatch und ForceTableSystemGroup befinden

Darstellung von Bausteinen und Typen in der TIA Portal Openness API

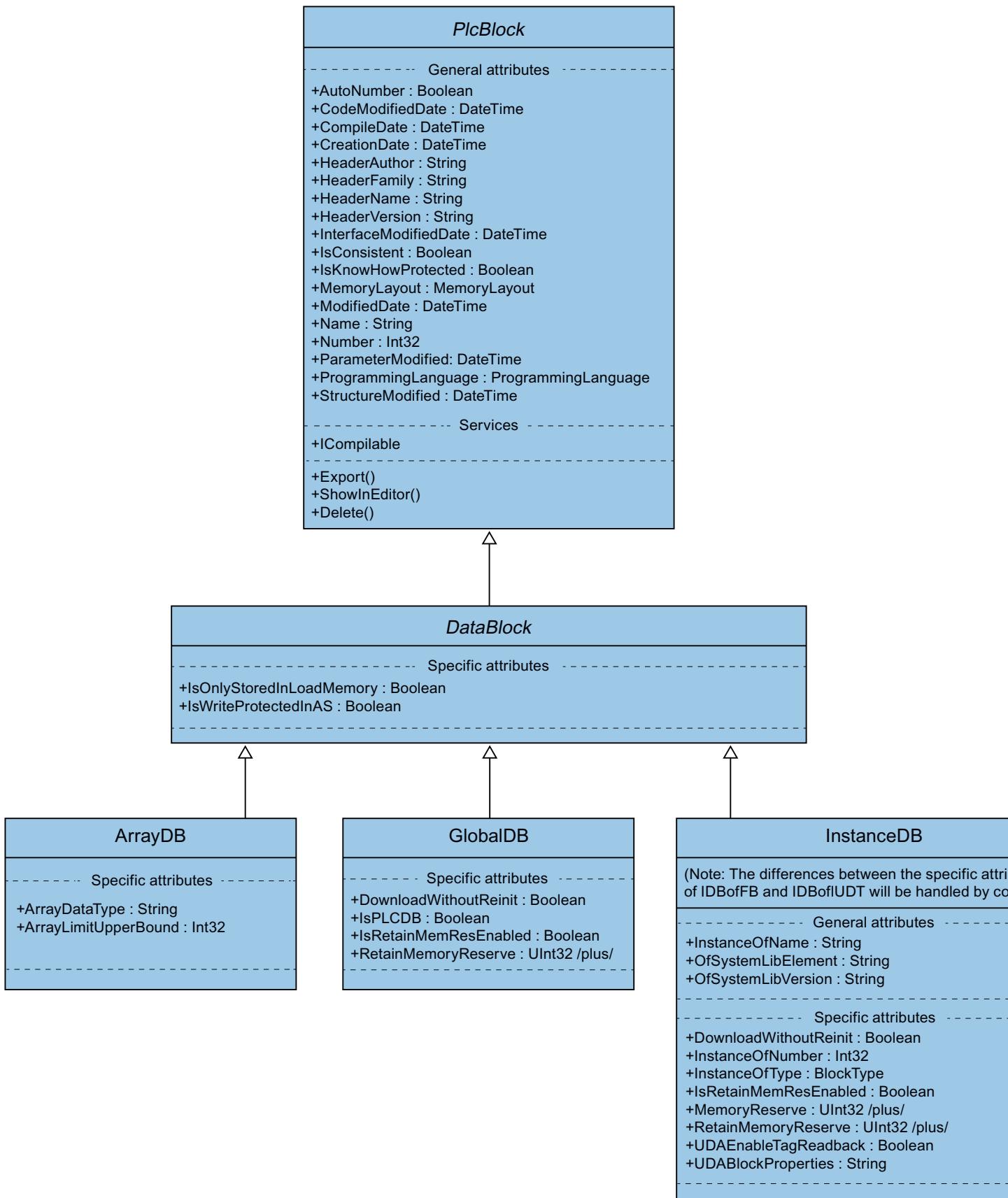
Der vereinfachte Modellteil von Bausteinen und für die Struktur basiert auf den Attributen in der TIA Portal Openness API. Diese Klassen bieten die Exportfunktion und für Bausteine auch die Übersetzungsfunktion.

Klassendiagramme

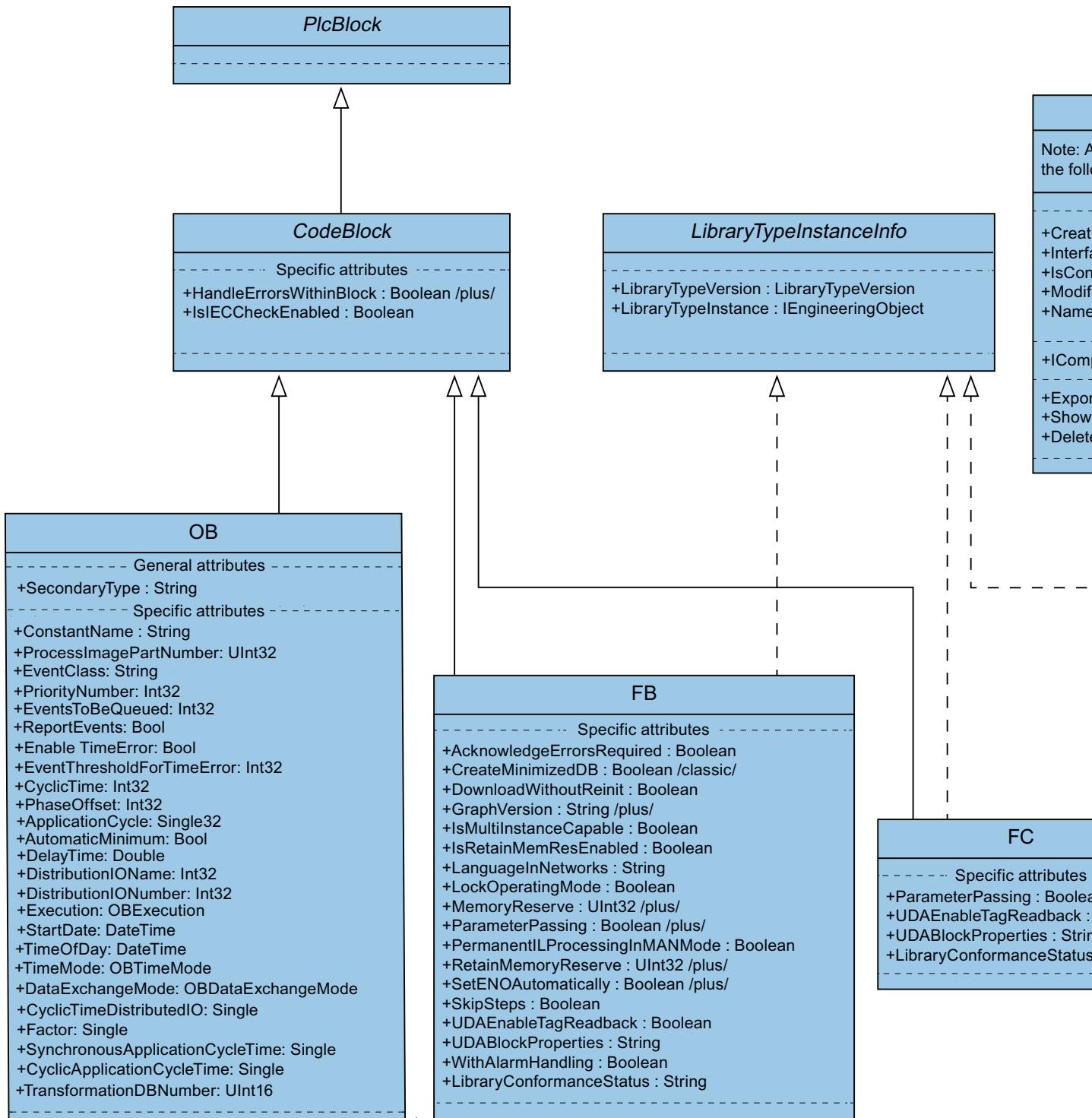
Im TIA Portal Openness-Objektmodell sind alle nicht direkt instanzierten Klassen als abstrakt definiert.

Daten

7.4 Bausteine und Typen des TIA Portal Openness-Objektmodells



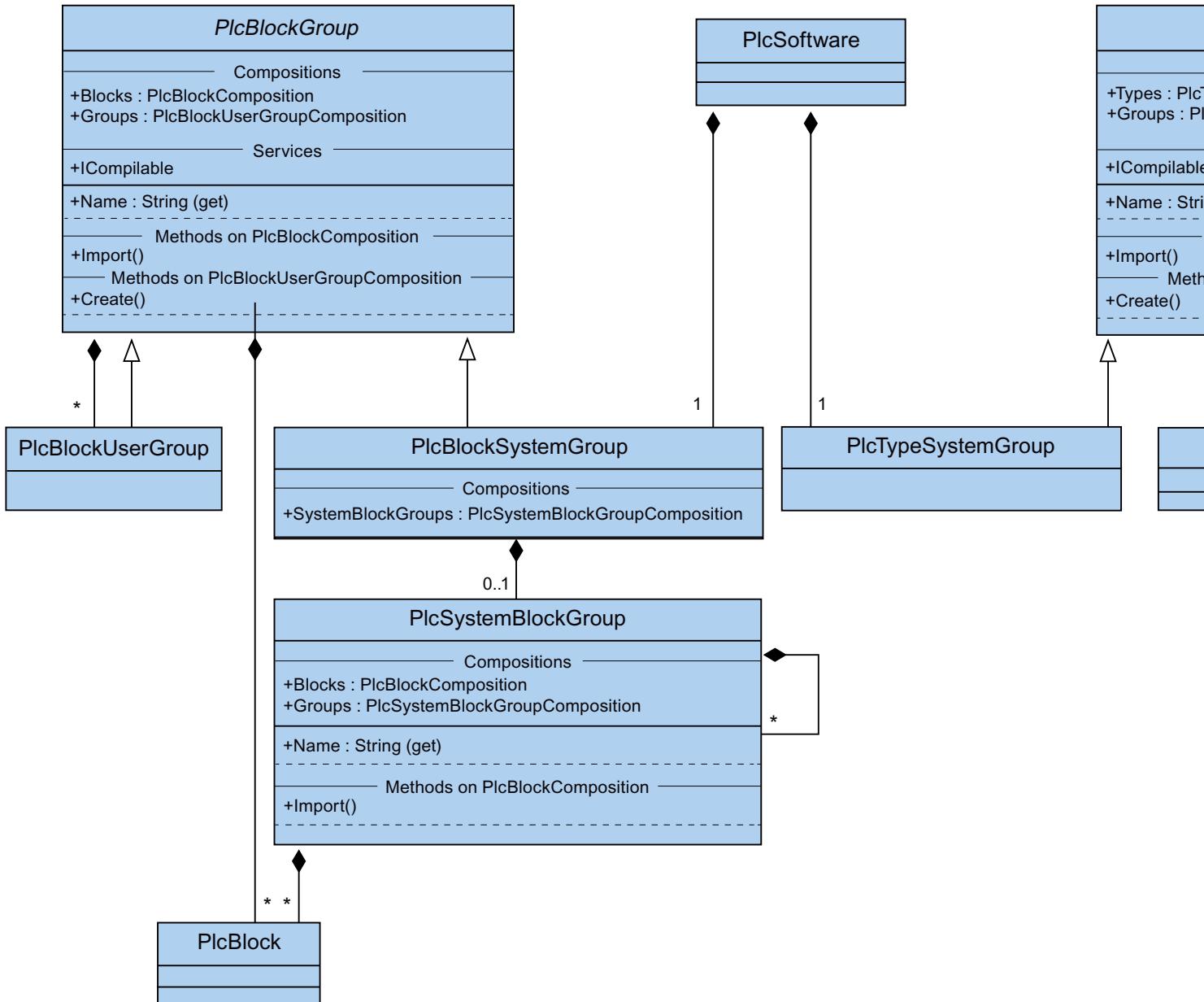
Code und Typ



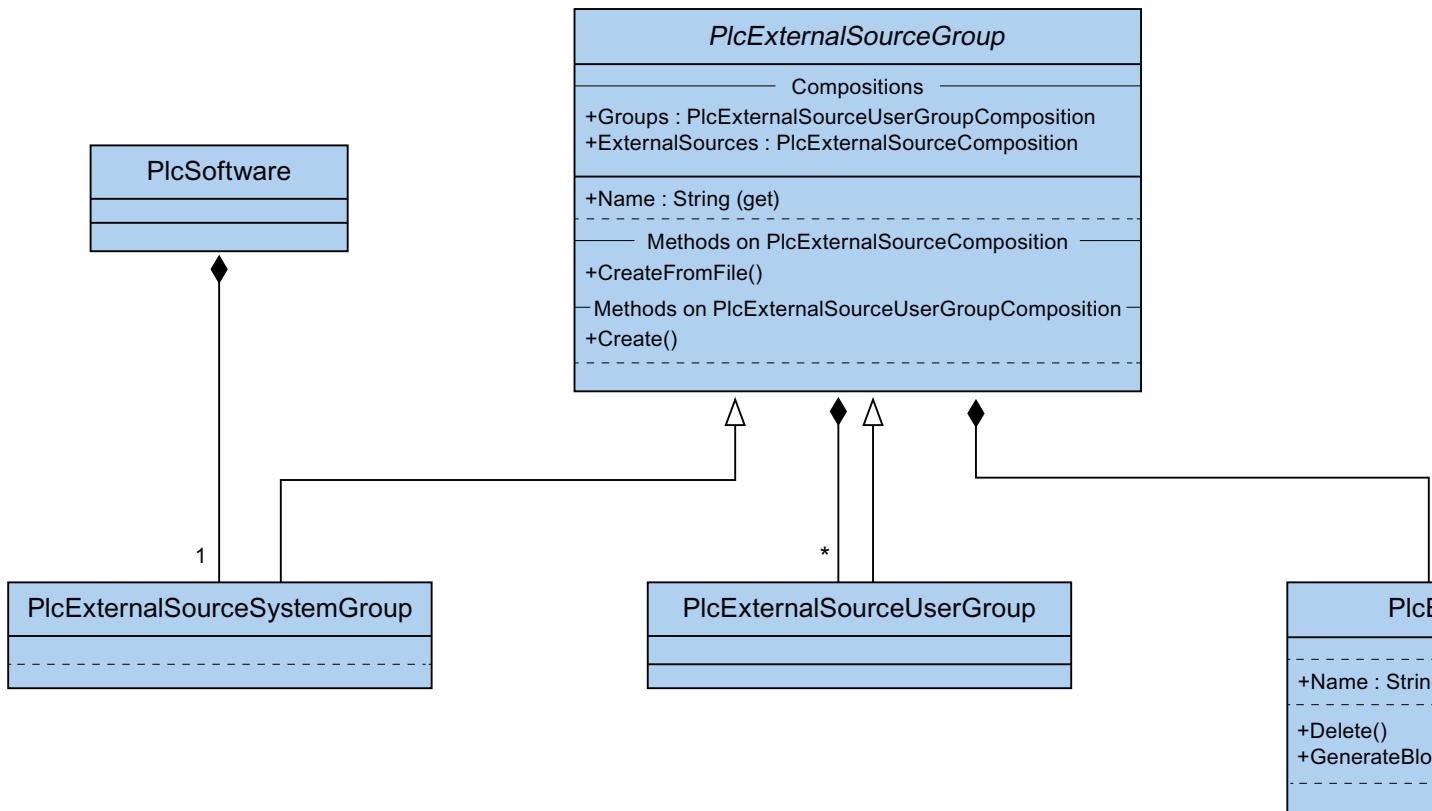
Darstellung von Gruppen für Bausteine und Typen in der TIA Portal Openness API

Die zwei Gruppen "PlcBlocks" der oberen Ebene ("Programmbausteine" in der Benutzeroberfläche des TIA Portals) und "PlcTypes" ("PLC-Datentypen" in der Benutzeroberfläche des TIA Portals) enthalten Bausteine und Typdefinitionen. Diese Gruppen stellen die Import- und die Übersetzungsfunktion für Bausteine zur Verfügung. Aufgrund der Tatsache, dass die meisten Methoden von Funktionalitäten der Gruppen nur über Sammlungen erreichbar sind, gibt es eine "eingebettete" oder "verdichtete" Darstellung der Sammlungen und ihrer Methoden in den "Host"-Klassen.

Bausteine und Typen



Externe Quellen



7.5 Hierarchie von Hardware-Objekten des Objektmodells

Beziehung zwischen den sichtbaren Elementen im TIA Portal und den modellierten Elementen im Objektmodell

Hardware-Objekt	Erläuterung
Gerät (Device)	Das Behälterobjekt für eine zentrale oder dezentrale Konfiguration
Geräteelement (DeviceItem)	Jedes Geräteelementobjekt hat ein Behälterobjekt. Die logische Beziehung ist "Elemente".

Die Behälterbeziehung ist vergleichbar mit der Beziehung der Module für die Geräteelementobjekte.

Beispiel: Ein Gerät umfasst einen oder mehrere Steckplätze. Ein Steckplatz umfasst Module. Ein Modul umfasst Submodule.

Hierbei handelt es sich um die Beziehung, die der Darstellung in Netzsicht und Gerätesicht des TIA Portals ähnelt. Das Attribut "PositionNumber" eines Geräteelements ist im Elementebereich innerhalb eines Behälters eindeutig.

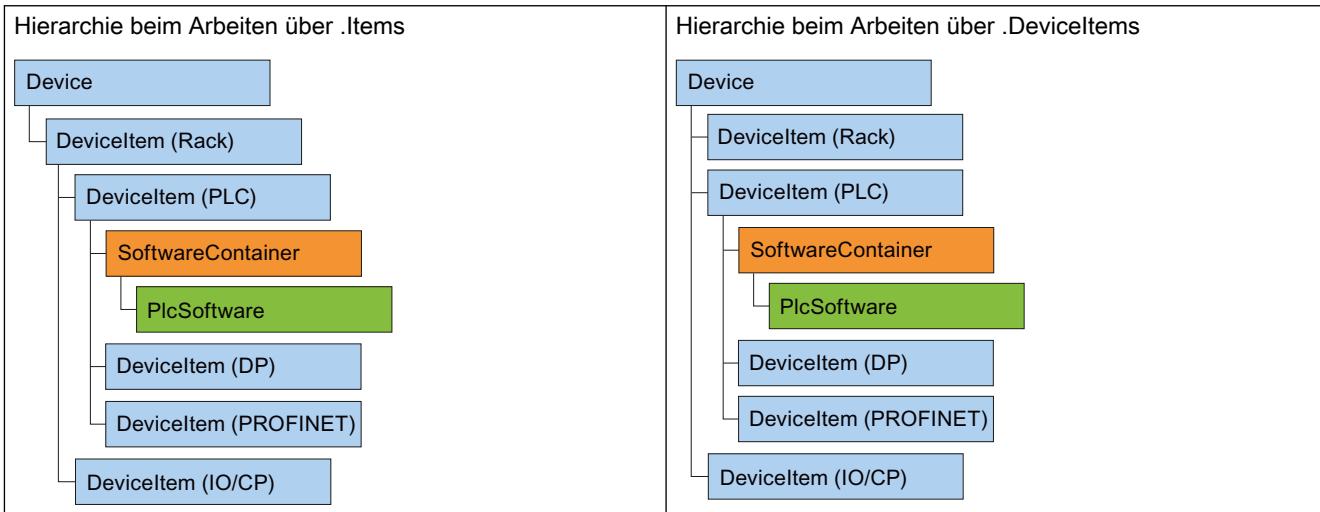
7.5 Hierarchie von Hardware-Objekten des Objektmodells

Die Eltern-Kind-Beziehung zwischen Geräteelementobjekten ist eine rein logische Beziehung im Objektmodell. Ein Kind kann nicht ohne seine Eltern existieren.

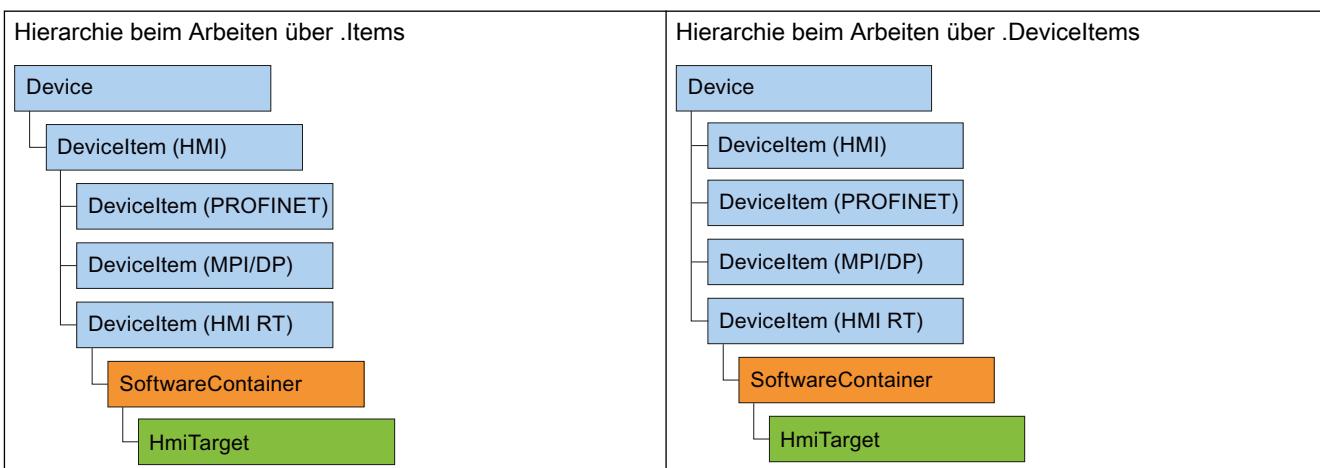
- Wenn ein Submodul als Teil eines Moduls modelliert wird (als Kind), kann das Submodul nicht ohne das Modul entfernt werden.
- Wenn Sie einem Modul ein Submodul hinzufügen und es dann vom Modul entfernen, hat dieses Kind die gleichen Eltern wie das Modul.

Das Diagramm unten zeigt die Hierarchie zwischen Geräten und Geräteelementen von PLC- und HMI-Geräten.

Hierarchie zwischen PLC-Geräten



Hierarchie zwischen HMI-Geräten



7.6 Informationen über installierte TIA Portal Openness-Versionen

Voraussetzung

- TIA Portal Openness und TIA Portal sind installiert

Verwendung

Ab TIA Portal Openness V14 hat jede installierte Version einen Registrierungsschlüssel, der Informationen über die Version enthält. Das ermöglicht die automatische Erzeugung von Anwendungskonfigurationsdatei für jede installierte Version von TIA Portal Openness.

Den Registrierungsschlüssel finden Sie unter dem folgendem Pfad:

HKEY_LOCAL_MACHINE\Software\Siemens\Automation\Openness\Vxx.x
\PublicAPI

Hinweis

Die Versionsnummer in diesem Pfad ist immer die Nummer der gegenwärtig installierten Version des TIA Portals. Bei mehreren parallelen Installationen gibt es mehrere Sätze von Einträgen für TIA Portal Openness in der Registrierung.

Es gibt einen einzigen Schlüssel für jede Version von TIA Portal Openness. Die Namen der Versionen sind die gleichen wie in der beschriebenen Assembly, zum Beispiel die Registrierungseinträge für TIA Portal Openness:

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

Hinweis

Wenn Sie eine Anwendungskonfigurationsdatei erzeugen möchten, können Sie den Pfad für Siemens.Engineering.dll, Siemens.Engineering.Hmi.dll und Token des öffentlichen Schlüssels dem Registrierungsschlüssel entnehmen.

7.7 Anwendungsbeispiel: API-Zugriff in einer Windows-Forms Applikation erstellen

Anwendungsbeispiel: API-Zugriff in einer Anwendung erstellen

Der vollständige Programmcode des Anwendungsbeispiels wird nachstehend aufgeführt. Die typischen Schritte für die Programmierung werden nachfolgend anhand dieses Beispiels beschrieben.

Hinweis

Für das Anwendungsbeispiel ist eine Anwendungskonfigurationsdatei erforderlich.

7.7 Anwendungsbeispiel: API-Zugriff in einer Windows-Forms Applikation erstellen

```

using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.HW.Features;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler;
using Siemens.Engineering.Library;
using System.IO;
namespace HelloTIA
{
    internal class Program
    {
        private static void Main(string[] args)
        {
            RunTiaPortal();
        }

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

                Console.WriteLine("Opening Project...");
                // please adapt the path and the extension apx to the installed version of
TIA Portal
                FileInfo projectPath = new FileInfo("C:\Demo\AnyCompanyProject.apx"); //edit
the path according to your project
                Project project = null;
                try
                {
                    project = projects.OpenWithUpgrade(projectPath);
                }
                catch (Exception)
                {
                    Console.WriteLine(String.Format("Could not open project {0}", projectPath.FullName));
                    Console.WriteLine("Demo complete hit enter to exit");
                    Console.ReadLine();
                }
            }
        }
    }
}

```

```
        return;
    }

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

    IterateThroughDevices(project);

    project.Close();

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

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

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

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

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

Vorgehensweise in Einzelschritten

1. TIA Portal in der Entwicklungsumgebung bekannt machen

Erzeugen Sie in Ihrer Entwicklungsumgebung einen Verweis auf alle "dll-Dateien" im Verzeichnis "C:\Program Files\Siemens\Automation\PortalV..\PublicAPI\V..".

Nachstehend finden Sie eine Beschreibung dieses Vorgangs anhand der Datei "Siemens.Engineering.dll" als Beispiel.

Die Datei "Siemens.Engineering.dll" befindet sich im Verzeichnis "C:\Program Files\Siemens\Automation\PortalV..\PublicAPI\V..". Erzeugen Sie in Ihrer Entwicklungsumgebung einen Verweis auf die Datei "Siemens.Engineering.dll".

Hinweis

Weisen Sie dem Parameter "CopyLocal" in den Referenzattributen den Wert "False" zu.

2. Namensraum für das TIA Portal veröffentlichen

Fügen Sie den folgenden Code hinzu:

```
using Siemens.Engineering;
```

3. TIA Portal veröffentlichen und starten

Um das TIA Portal zu veröffentlichen und zu starten, fügen Sie den folgenden Code ein:

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

4. Projekt öffnen

Um ein Projekt zu öffnen, können Sie beispielsweise den folgenden Code verwenden:

```
ProjectComposition projects = tiaPortal.Projects;
Console.WriteLine("Opening Project...");
//please adapt the path and the extension apx to the installed version of TIA Portal
FileInfo projectPath = new FileInfo("C:\Demo\AnyCompanyProject.apx");
Project project = null;
try
{
    project = projects.Open(projectPath);
}
catch (Exception)
{
    Console.WriteLine(String.Format("Could not open project {0}", projectPath.FullName));
    Console.WriteLine("Demo complete hit enter to exit");
    Console.ReadLine();
    return;
}
Console.WriteLine(String.Format("Project {0} is open", project.Path.FullName));
```

5. Geräte eines Projekts enumerieren

Um alle Geräte des Projekts zu enumerieren, fügen Sie den folgenden Code ein:

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

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

6. Projekt speichern und schließen

Fügen Sie den folgenden Code ein und speichern und schließen Sie das Projekt:

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

7.8 Einsatz der Codebeispiele

Aufbau der Codeausschnitte

Jeder Codeausschnitt in dieser Dokumentation wird als Funktion ohne Rückgabewert mit einem Objektverweis als Übertragungsparameter implementiert. Auf das Beseitigen von Objekten wird zum Zweck der besseren Lesbarkeit verzichtet. Objekte aus dem TIA Portal werden mit ihrem Namen mithilfe der Methode `Find` adressiert.

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

Zum Ausführen dieses Codeausschnitts benötigen Sie Folgendes:

- Ein WinCC-Projekt mit einem HMI-Gerät, das eine Gruppe mit mindestens einem Bild beinhaltet.
- Eine Funktion, die das Bediengerät instanziert.

Hinweis

Wenn Sie Verzeichnispfade angeben, verwenden Sie den absoluten Verzeichnispfad, zum Beispiel "C:/path/file.txt".

Relative Verzeichnispfade sind nur in den XML-Dateien für Import und Export zulässig, zum Beispiel "file.txt" oder "C:/path01/.../path02/file.txt".

Beispiel für Ausführung des Codeausschnitts

Verwenden Sie das folgende Beispiel zum Ausführen des Codeausschnitts "DeleteScreenFromFolder" als Teil des Beispielprogramms "Hello TIA":

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

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

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

7.9 Allgemeine Funktionen

7.9.1 Unterstützung von IntelliSense durch TIA Portal Openness

Verwendung

Der Intellisense-Support für TIA Portal Openness hilft Ihnen bei verfügbaren Attributen oder Methoden über Tooltip-Informationen. Er kann Aufschluss über Anzahl, Namen und Typen der erforderlichen Parameter geben. Im folgenden Beispiel gibt der fett gedruckte Parameter in der ersten Zeile den nächsten beim Eingeben der Funktion erforderlichen Parameter an.

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

Project ProjectComposition.OpenWithUpgrade(FileInfo path)
Open Action with project update is necessary

Sie können Informationen über Parameter manuell abrufen. Dies geschieht durch Klicken auf "Edit IntelliSense/Parameter Info", durch Betätigen der Tastenkombination CTRL+SHIFT +SPACE oder durch Klicken auf die Schaltfläche "Parameter Info" in der Symbolleiste des Editors.

7.9.2 Verbindung zum TIA Portal aufbauen

Einleitung

Sie starten das TIA Portal mit TIA Portal Openness oder stellen die Verbindung zu einem bereits ausgeführten TIA Portal her. Wenn Sie das TIA Portal mit einer TIA Portal Openness-Anwendung starten, geben Sie an, ob das TIA Portal mit oder ohne grafische Benutzeroberfläche gestartet werden soll. Wenn Sie mit dem TIA Portal ohne Benutzeroberfläche arbeiten, wird das TIA Portal nur als Prozess des Betriebssystems gestartet. Sie erstellen mit einer TIA Portal Openness-Anwendung bei Bedarf mehrere Instanzen des TIA Portals.

Hinweis

Wenn Sie über die TIA Portal Openness-Anwendung auf die Oberfläche des TIA Portals zugreifen, können Sie keinen HMI-Editor verwenden. Sie können den Editor „Geräte & Netze“ oder den Programmiereditor manuell oder mit TIA Portal Openness API öffnen.

Zum Starten des TIA Portals mit einer TIA Portal Openness-Anwendung stehen Ihnen folgende Optionen zur Verfügung.

- Verwenden Sie eine Anwendungskonfigurationsdatei (in den meisten Fällen zu empfehlen).
- Verwenden Sie die Methode "AssemblyResolve" (zu empfehlen beim Kopieren usw.).
- Kopieren Sie die Siemens.Engineering.dll in das Verzeichnis der TIA Portal Openness-Anwendung.

Hinweis

Es empfiehlt sich, die Siemens.Engineering.dll mit Hilfe der Anwendungskonfigurationsdatei zu laden. Bei Verwendung dieser Methode werden die starken Namen berücksichtigt und schädliche Änderungen an der Engineering.dll führen zu einem Ladefehler. Bei Verwendung der Methode AssemblyResolve ist dies nicht erkennbar.

Starten des TIA Portals mit einer Anwendungskonfigurationsdatei

Erzeugen Sie in der Anwendungskonfigurationsdatei Verweise auf alle erforderlichen Programmbibliotheken. Die Anwendungskonfigurationsdatei verteilen Sie zusammen mit der TIA Portal Openness-Anwendung.

Speichern Sie die Anwendungskonfigurationsdatei "app.config" im selben Verzeichnis wie die TIA Portal Openness-Anwendung und beziehen Sie diese Datei in Ihre Anwendung ein. Prüfen Sie, ob der Dateipfad in jedem Code mit dem Installationspfad des TIA Portals übereinstimmt.

Für die Anwendungskonfigurationsdatei können Sie den folgenden Codeausschnitt verwenden:

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

Verwenden Sie den folgenden Programmcode zum Öffnen einer neuen Instanz des TIA Portals über die Anwendungskonfigurationsdatei:

```
//Connect a TIA Portal Openness application via API using
using System;
using System.IO;
using Siemens.Engineering;

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

Starten des TIA Portals mit der Methode "AssemblyResolve"

Bauen Sie den Programmcode von TIA Portal Openness so auf, dass die Registrierung auf das Ereignis "AssemblyResolve" so früh wie möglich erfolgt. Verkapseln Sie den Zugriff auf das TIA Portal in einem zusätzlichen Objekt oder einer zusätzlichen Methode.

Vorsicht ist beim Auflösen der Engineering Assembly mit einer Assembly-Resolver-Methode geboten. Wenn Typen aus der Engineering Assembly verwendet werden, bevor der Assembly Resolver ausgeführt wurde, stürzt das Programm ab. Der Grund dafür ist, dass der Just-in-time-Übersetzer (JIT-Übersetzer) eine Methode erst dann übersetzt, wenn er sie ausführen muss. Wenn Typen einer Engineering Assembly beispielsweise in "Main" verwendet werden, versucht der JIT-Übersetzer, "Main" bei laufendem Programm zu übersetzen. Das schlägt fehl, weil der JIT-Übersetzer nicht weiß, wo die Engineering Assembly zu finden ist. Die Registrierung des Assembly Resolver in Main ändert daran nichts. Die Methode muss vor der Registrierung des Assembly Resolver laufen und übersetzt werden, bevor der Assembly Resolver ausgeführt werden kann. Die Lösung für dieses Problem besteht darin, die Business Logic, die Typen aus der Engineering Assembly verwendet, in einer separaten Methode unterzubringen. Dabei verwendet die separate Methode nur Typen, die der JIT-Übersetzer bereits versteht. Das vorliegende Beispiel verwendet eine Methode, die "void" zurückgibt, keine Parameter hat und sämtliche Business Logic enthält. Jetzt wird "Main" vom JIT-Übersetzer erfolgreich übersetzt, weil er alle Typen in Main versteht. Wenn RunTiaPortal während der Laufzeit aufgerufen wird, ist der Assembly Resolver bereits registriert. Somit weiß

der JIT-Übersetzer bei dem Versuch, die Typen der Business Logic zu finden, wo sich die Engineering Assembly befindet.

Verwenden Sie den folgenden Programmcode zum Öffnen einer neuen Instanz des TIA Portals.

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

namespace UserProgram
{
    static class MyProgram
    {
        public static void Main(string[] args)
        {
            AppDomain.CurrentDomain.AssemblyResolve += MyResolver;
            RunTiaPortal();
        }

        private static void RunTiaPortal()
        {
            // To start TIA Portal with user interface:
            // using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            //
            // To start TIA Portal without user interface:
            // using (TiaPortal tiaPortal = new
            TiaPortal(TiaPortalMode.WithoutUserInterface))
            using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            {
                //begin of code for further implementation
                //...
                //end of code
            }
        }

        private static Assembly MyResolver(object sender, ResolveEventArgs args)
        {
            int index = args.Name.IndexOf(',');
            if (index == -1)
            {
                return null;
            }

            string name = args.Name.Substring(0, index) + ".dll";
            // Edit the following path according to your installed version of TIA Portal
            string path = Path.Combine(@"C:\Program Files\Siemens\Automation\Portal Vxx
\PublicAPI\Vxx\", name);
            string fullPath = Path.GetFullPath(path);
            if (File.Exists(fullPath))
            {
                return Assembly.LoadFrom(fullPath);
            }
            return null;
        }
    }
}
```

Auf aktive Instanzen des TIA Portals zugreifen

Um die Verbindung zu einer aktiven Instanz des TIA Portals mit einer TIA Portal Openness-Anwendung herstellen zu können, listen Sie zunächst die Instanzen des TIA Portals auf. Innerhalb einer Windows-Sitzung können Sie Verbindungen zu mehreren Instanzen herstellen. Die aktive Instanz kann das TIA Portal mit oder ohne gestartete Benutzeroberfläche sein:

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

Wenn Sie die Prozess-ID der Instanz des TIA Portals kennen, verwenden Sie die Prozess-ID für den Zugriff auf das Objekt. Der Startvorgang des TIA Portals benötigt eine gewisse Zeit, bevor Sie TIA Portal Openness mit dem TIA Portal verbinden können.

Beim Herstellen einer Verbindung zu einer aktiven Instanz des TIA Portals erscheint eine Verbindungsaufforderung der TIA Portal Openness-Firewall. Hier können Sie für die Verbindung Folgendes angeben:

- Verbindung einmal erlauben
 - Verbindung nicht erlauben
 - Verbindungen von dieser Applikation immer erlauben
- Weitere Informationen hierzu siehe TIA Portal Openness-Firewall (Seite 84).

Hinweis

Wird die Registry-Eingabeaufforderung dreimal zurückgewiesen, löst das System eine Ausnahme vom Typ `EngineeringSecurityException` aus.

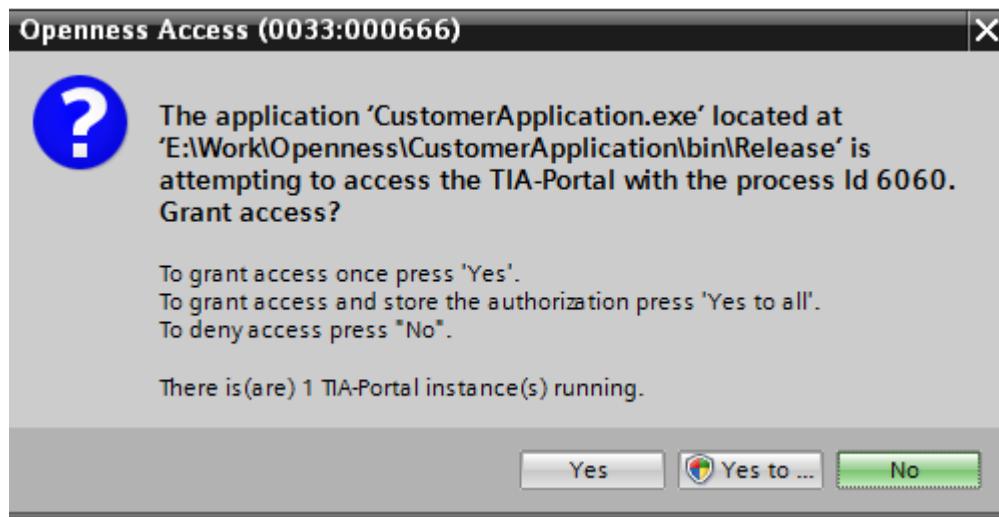
Nach Herstellung der Verbindung zum Prozess können Sie mit Hilfe eines der folgenden Attribute Informationen zu den Instanzen des TIA Portals abrufen:

Attribut	Information
InstalledSoftware as IList<TiaPortalProduct>	Gibt Informationen über die installierten Produkte zurück.
Mode as TiaPortalMode	Gibt den Modus zurück, in dem das TIA Portal gestartet wurde (<code>WithoutUserInterface</code> / <code>WithUserInterface</code>).
AttachedSessions as IList<TiaPortalSession>	Gibt eine Liste von Anwendungen zurück, die mit dem TIA Portal verbunden sind.
ProjectPath as FileInfo	Gibt den Dateinamen des im TIA Portal geöffneten Projekts einschließlich des Ordners zurück, z. B. "D:\WinCCProjects\ColorMixing\ColorMixing.ap*" Wenn kein Projekt geöffnet ist, wird eine leere Zeichenfolge zurückgegeben.
ID as int	Gibt die Prozess-ID der Instanz des TIA Portals zurück.
Path as FileInfo	Gibt den Pfad zur ausführbaren Datei des TIA Portals zurück.

7.9.3 TIA Portal Openness-Firewall

Eingabeaufforderung der TIA Portal Openness-Firewall

Wenn Sie versuchen, über TIA Portal Openness eine Verbindung zu einem laufenden TIA Portal herzustellen, fordert das TIA Portal Sie auf, die Verbindung zu akzeptieren oder zurückzuweisen, wie im folgenden Screenshot dargestellt.



Verbindung zum TIA Portal einmal erlauben

Wenn Sie Ihre TIA Portal Openness-Anwendung nur einmal mit dem TIA Portal verbinden möchten, klicken Sie bei der Eingabeaufforderung auf "Yes". Wenn Ihre TIA Portal Openness-Anwendung das nächste Mal versucht, eine Verbindung zum TIA Portal herzustellen, wird die Eingabeaufforderung wieder angezeigt.

Whitelist-Eintrag durch Verbinden mit dem TIA Portal hinzufügen

Zum Erzeugen eines Whitelist-Eintrags für Ihre TIA Portal Openness-Anwendung befolgen Sie diese Schritte:

1. Wenn Sie bei der Eingabeaufforderung auf "Yes to all" klicken, wird ein Dialog zur Benutzerkontensteuerung angezeigt.
2. Klicken Sie im Dialog zur Benutzerkontensteuerung auf "Yes", um Ihre Anwendung zur Whitelist in der Windows-Registrierdatenbank hinzuzufügen und um die Anwendung mit dem TIA Portal zu verknüpfen.

Whitelist-Eintrag ohne TIA Portal hinzufügen

Wenn Sie der Whitelist einen Eintrag hinzufügen möchten, ohne dafür das TIA Portal zu verwenden, können Sie eine Registrierungsdatei wie diese erzeugen:

```
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation\Openness\Vxx.x\Whitelist
\CustomerApplication.exe]
[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation\Openness\Vxx.x\Whitelist
\CustomerApplication.exe\Entry]
"Path"="E:\\Work\\Openness\\CustomerApplication\\bin\\Release\\CustomerApplication.exe"
"DateModified"="2014/06/10 15:09:44.406"
"FileHash"="0rXRKUCNzMWHOMFrT52OwXzqJef1oran4UykTeBraaY="
```

Das folgende Beispiel zeigt, wie sie den Filehash-Wert und das Datum der letzten Änderung berechnen können:

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

7.9.4 Event-Handler

Ereignis-Handler in der TIA Portal Openness-Anwendung

Eine Instanz des TIA Portals bietet die folgenden Ereignisse, auf die Sie mit einem Ereignis-Handler in einer TIA Portal Openness-Anwendung reagieren können. Sie können auf die Attribute von Benachrichtigungen zugreifen und die Antworten entsprechend definieren.

Ereignis	Antwort
Disposed	Mit diesem Ereignis reagieren Sie auf das Schließen des TIA Portals mit einer TIA Portal Openness-Anwendung.
Notification	Mit diesem Ereignis reagieren Sie auf Benachrichtigungen des TIA Portals mit einer TIA Portal Openness-Anwendung. Benachrichtigungen müssen lediglich quittiert werden, zum Beispiel mit "OK".
Confirmation	Mit diesem Ereignis reagieren Sie auf Bestätigungen des TIA Portals mit einer TIA Portal Openness-Anwendung. Bestätigungen erfordern immer eine Entscheidung, zum Beispiel "Möchten Sie das Projekt speichern?".

Programmcode

Ändern Sie den folgenden Programmcode, um Ereignis-Handler in einer TIA Portal Openness-Anwendung zu registrieren:

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

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

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

private static void TiaPortal_Notification(object sender, NotificationEventArgs e)

{
    .....
}

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

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

Attribute von Benachrichtigungen des TIA Portals

Benachrichtigungen des TIA Portals haben die folgenden Attribute:

Attribut	Beschreibung
Caption	Gibt den Namen der Bestätigung zurück.
DetailText	Gibt den Detailtext der Bestätigung zurück.
Icon	Gibt das Symbol der Bestätigung zurück.
IsHandled	Gibt die Bestätigung zurück oder gibt an, ob die Bestätigung noch aussteht.
Text	Gibt den Text der Bestätigung zurück.

Attribute von Bestätigungen

Bestätigungen haben die folgenden Attribute:

Attribut	Beschreibung
Caption	Gibt den Namen der Bestätigung zurück.
Choices	Gibt die Option zum Quittieren der Bestätigung zurück.
DetailText	Gibt den Detailtext der Bestätigung zurück.
Icon	Gibt das Symbol der Bestätigung zurück.
IsHandled	Gibt die Bestätigung zurück oder gibt an, ob die Bestätigung noch aussteht.
Result	Gibt das Ergebnis der Quittierung zurück oder gibt es an.
Text	Gibt den Text der Bestätigung zurück.

Siehe auch

Dialoge mit Systemmeldungen programmgesteuert bestätigen (Seite 87)

7.9.5 Dialoge mit Systemmeldungen programmgesteuert bestätigen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Ereignis-Handler sind registriert.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)

Verwendung

Wenn Sie das TIA Portal mit der Benutzeroberfläche bedienen, werden für einige Programmabläufe Dialoge mit Systemereignissen angezeigt. Anhand dieser Systemereignisse entscheiden Sie, wie Sie fortfahren möchten.

Wenn mit einer TIA Portal Openness-Anwendung auf das TIA Portal zugegriffen wird, müssen diese Systemereignisse über entsprechende „.NET“-Ereignisse quittiert werden.

Die zulässigen Bestätigungen sind in der Liste `Choices` enthalten:

- Abort
- Cancel
- Ignore
- No
- NoToAll
- None

- OK
- Retry
- Yes
- YesToAll

Der Wert von ConfirmationEventArgs.Result muss einer der oben aufgeführten Einträge sein. Andernfalls wird eine Ausnahme ausgelöst.

Programmcode

Ändern Sie den folgenden Programmcode, um auf ein Bestätigungsereignis zu reagieren:

```
...
    tiaPortal.Confirmation += TiaPortalConfirmation;
...
private void TiaPortalConfirmation(object sender, ConfirmationEventArgs e)
{
    ...
}
```

Ändern Sie den folgenden Programmcode, um den Projekteur über ausgeführte Aktionen einer TIA Portal Openness-Anwendung zu benachrichtigen:

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

7.9.6 Verbindung zum TIA Portal beenden

Einleitung

Wenn Sie die TIA Portal-Instanz ohne eine Benutzeroberfläche gestartet haben und Ihre Anwendung der einzige mit dem TIA Portal verknüpfte TIA Portal Openness-Client ist, können Sie die Instanz des TIA Portals mit der TIA Portal Openness-Anwendung schließen. Andernfalls trennen Sie die TIA Portal Openness-Anwendung von der TIA Portal-Instanz.

Trennen oder schließen Sie die aktive Instanz des TIA Portals mit der Methode `IDisposable.Dispose()`.

Die Methode `IDisposable.Dispose()` können Sie wie folgt verwenden:

- Mit einem using-Statement.
- Umgeben Sie die Objektbeschreibung mit einem try-finally-Block und rufen Sie die Methode `IDisposable.Dispose()` innerhalb des finally-Blocks auf.

Wenn Sie die aktive Instanz des TIA Portals beenden, können Sie nicht mehr auf das TIA Portal zugreifen.

Hinweis

Wenn ein Projektor die TIA Portal-Instanz trotz laufenden Zugriffs einer TIA Portal Openness-Anwendung schließt, wird beim nächsten Zugriff in der TIA Portal Openness-Anwendung eine Ausnahme der Klasse "NonRecoverableException" ausgelöst. Sie können ein Disposed-Ereignis abonnieren, um einen Aufruf beim Schließen des TIA Portals zu erhalten.

Programmcode

Ändern Sie den folgenden Programmcode, um die Verbindung zum TIA Portal zu trennen oder zu schließen:

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

Siehe auch

Event-Handler (Seite 85)

7.9.7 Diagnoseschnittstellen im TIA Portal

Verwendung

Sie können bestimmte Diagnoseinformationen von laufenden Instanzen des TIA Portals über eine statische Methode abrufen. Die Diagnoseschnittstelle ist im Objekt `TiaPortalProcess` implementiert und dieses Objekt kann für jede aktuell laufende Instanz des TIA Portals abgerufen werden.

Die Diagnoseschnittstelle ist nicht gesperrt. Sie können also auf das Objekt `TiaPortalProcess` abrufen und auf seine Mitglieder zugreifen, und zwar unabhängig davon, ob das TIA Portal belegt ist oder nicht. Die Diagnoseschnittstelle umfasst folgende Mitglieder:

Klasse TiaPortalProcess

Mitglied	Typ	Funktion
AcquisitionTime	DateTime	Zeitpunkt, zu dem das TiaPortalProcess-Objekt erfasst wurde. Da das Objekt TiaPortalProcess eine vollkommen statische Momentaufnahme vom Zustand des TIA Portals zu einem gegebenen Zeitpunkt darstellt, können darin enthaltene Informationen veraltet sein.
Attach	TiaPortal	Hängt sich dem gegebenen TiaPortalProcess an und gibt eine Instanz des TIA Portals zurück.
AttachedSessions	IList<TiaPortalSession>	Eine Sammlung aller anderen Sitzungen, die gegenwärtig an dasselbe TIA Portal angehängt sind. Diese Sammlung kann leer sein. Jede Sitzung wird durch ein nachstehend beschriebenes TiaPortalSession-Objekt repräsentiert.
Attaching	EventHandler<AttachingEventArgs>	Dieses Ereignis ermöglicht einer Anwendung, Versuche zum Anhängen an das TIA Portal zuzulassen. Wenn eine andere Anwendung versucht, sich dem TIA Portal anzuhängen, werden die Teilnehmer dieses Ereignisses benachrichtigt. Die Teilnehmer haben dann 10 Sekunden Zeit, um dieser Anbindung zuzustimmen. Wenn ein Teilnehmer dieses Ereignis ignoriert oder nicht rechtzeitig reagiert, gilt das als Ablehnung und der anderen Anwendung wird das Anhängen nicht erlaubt. Abgestürzte Anwendungen, die nicht in der Lage sind, auf dieses Ereignis zu reagieren, können einer Anwendung die Erlaubnis zum Anhängen nicht verweigern.
Dispose	void	Schließt die betreffende Instanz des TIA Portals.
Id	int	Die Prozess-ID des TIA Portals.
InstalledSoftware	IList<TiaPortalProduct>	Eine Sammlung aller gegenwärtig installierten Produkte als Teil des TIA Portals. Jedes Produkt wird durch ein nachstehend beschriebenes TiaPortalProduct-Objekt dargestellt.

Mitglied	Typ	Funktion
Mode	TiaPortalMode	Gibt den Modus zurück, in dem das TIA Portal gestartet wurde. Die aktuellen Werte sind WithUserInterface und WithoutUserInterface.
Path	FileInfo	Der Pfad zur ausführbaren Datei des TIA Portals.
ProjectPath	FileInfo	Der Pfad zu dem Projekt, das gegenwärtig im TIA Portal geöffnet ist. Wenn kein Projekt geöffnet ist, hat dieses Attribut den Wert Null.

Klasse TiaPortalSession

Mitglied	Typ	Funktion
AccessLevel	TiaPortalAccessLevel	Zugriffsebene der Sitzung. Wird dargestellt als Merker-Enumeration, wobei mehrere Zugriffsebenen möglich sind. TiaPortalAccessLevel wird nachstehend ausführlich beschrieben.
AttachTime	DateTime	Der Zeitpunkt, zu dem die Verbindung mit dem TIA Portal hergestellt wurde.
Id	int	Die ID der aktuellen Sitzung.
IsActive	bool	Gibt "wahr" zurück, wenn das TIA Portal gegenwärtig einen Aufruf aus dieser Sitzung verarbeitet.
Dispose	void	Trennt die Verbindung des Prozesses zum TIA Portal. Diese Methode erzwingt nicht die Beendigung des Prozesses selbst, so wie es mit System.Diagnostics.Process.Kill der Fall wäre. Die Anwendung, deren Verbindung beendet wird, erhält dennoch ein Disposed-Ereignis. Es erfolgt aber kein weiterer Hinweis darauf, weshalb die Verbindung beendet wurde.
ProcessId	int	Die Prozess-ID des angehängten Prozesses.
ProcessPath	FileInfo	Der Pfad der ausführbaren Datei des angehängten Prozesses.

Mitglied	Typ	Funktion
TrustAuthority	TiaPortalTrustAuthority	Gibt an, ob die aktuelle Sitzung von einem signierten Prozess gestartet wurde und ob es ein TIA Portal Openness-Zertifikat ist oder nicht. TrustAuthority ist eine Merker-Enumeration und wird nachstehend beschrieben.
UtilizationTime	TimeSpan	Zeitraum, den der Prozess aktiv im TIA Portal verbracht hat. Kombiniert mit dem Attribut Attach-Time kann dies dazu genutzt werden, prozentuale Nutzungs-daueranteile oder ähnliche Da-ten zu ermitteln.
Version	string	Die Version der Siemens.Engineering.dll, der die Sitzung ange-hängt wird.

Enum TiaPortalAccessLevel

Enumerationswert	Funktion
None	Dies ist kein gültiger Wert. Der Wert ist aufgeführt, weil TiaPortalAccessLevel eine Merkerenumeration ist und als solche einen entsprechenden "Nullwert" benötigt, um darzustellen, dass keine Merker gesetzt sind. Aber der Wert erscheint nie im tat-sächlichen Gebrauch, weil keine Sitzung ohne Zu-griff gestartet werden kann.
Published	Die Sitzung hat Zugriff auf publizierte Funktionali-tät.
Modify	Die Sitzung hat Änderungszugriff.

Enum TiaPortalTrustAuthority

Enumerationswert	Funktion
None	Das Hauptmodul des angehängten Prozesses ist nicht mit einem Zertifikat signiert.
Signed	Das Hauptmodul ist mit einem Zertifikat signiert, es ist aber kein TIA Portal Openness-Zertifikat.
Certified	Das Hauptmodul ist mit einem TIA Portal Open-ness-Zertifikat signiert.
CertifiedWithExpiration	Das Hauptmodul ist mit einem TIA Portal Open-ness-Zertifikat signiert, das seine Gültigkeit mit Ablauf seiner Lebensdauer verliert.

Klasse TiaPortalProduct

Mitglied	Typ	Funktion
Name	string	Der Name des Produkts (z. B. STEP 7 Professional).
Options	IList<TiaPortalProduct>	Eine Sammlung aller zum verbundenen TIA Portal gehörenden Optionspakete, dargestellt als TiaPortalProduct-Objekte. Wenn ein Optionspaket selbst Optionspakete hat, kann sich diese Schachtelung fortsetzen.
Version	string	Der Versionsstring des Produkts.

Der folgende Codeausschnitt bietet ein Beispiel dafür, wie die Diagnoseschnittstelle zum Abfragen von Informationen verwendet wird und wie diese Informationen anschließend in Ihrer Applikation genutzt werden.

```

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

```

Sicherheitsrelevante Information

Aufgrund der Tatsache, dass für die Nutzung der Diagnoseschnittstelle keine Verbindung zum TIA Portal gebraucht wird, ist es möglich, einen Windows-Dienst zu schreiben, der das sich anhängende Ereignis dazu nutzt, jede Anwendung, die sich an ein TIA Portal anzuhängen versucht, zu prüfen. So kann zum Beispiel festgelegt werden, dass nur Anwendungen, die mit dem Namen Ihres Unternehmens beginnen, sich anhängen dürfen. Eine andere Option könnte sein, immer Zugriff zu gewähren, aber Informationen über sich anhängende Prozesse in ein Protokoll zu schreiben. Der folgende Programmcode ist ein Beispiel für einen Ereignis-Handler zum Prüfen eingehender Verbindungen:

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

7.9.8 Ausschließlicher Zugriff

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Zum Einrichten eines exklusiven Prozesses auf einen angehängten Prozess des TIA Portals bietet die Klasse "TIA Portal" die Methode "ExclusiveAccess(String text)". Die Nutzung eines exklusiven Zugriffs wird dringend empfohlen, auch wenn er nicht obligatorisch ist.

Verwenden Sie "ExclusiveAccess" in einer "using"-Anweisung, um sicherzustellen, dass der Zugriff ordnungsgemäß beendet wird, wenn Ausnahmen auftreten oder die Anwendung heruntergefahren wird.

Hinweis

Jeder Versuch, einen zweiten exklusiven Zugriff innerhalb des Geltungsbereichs eines geöffneten exklusiven Zugriffs zu erzeugen, resultiert in der Auslösung einer wiederherstellbaren Ausnahme.

Ändern Sie das folgende Beispiel, um "ExclusiveAccess" auf eine Instanz zu erhalten:

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

Nach dem Erfassen einer "ExclusiveAccess"-Instanz für einen gegebenen Prozess des TIA Portals wird ein Dialog angezeigt. Dieser Dialog zeigt die bei der Instanziierung vorgesehene Meldung an. Darüber hinaus werden die folgenden Informationen über die Client-Anwendung angezeigt:

- der Assembly-Titel aus den Manifestdaten, wenn verfügbar, andernfalls der Prozessname
- die Prozess-ID
- die SID (Openness-Session-ID der TIA-Portal-Instanz der Client-Anwendung)

Hinweis

Für eine gegebene TIA Portal Openness-Client-Anwendung können mehrere Sitzungen aktiv sein, weil es mehrere Instanzen des TIA Portals gibt, wobei diese jeweils demselben TIA Portal-Prozess zugeordnet sind.

Die Client-Anwendung kann den angezeigten Inhalt des Dialogs für exklusiven Zugriffs auch aktualisieren, indem das Attribut "Text" mit neuen Werten besetzt wird. Ändern Sie den folgenden Programmcode, um dieses Verhalten herbeizuführen:

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

Sie können verlangen, dass der exklusive Zugriff mit der Schaltfläche „Cancel“/„Abbrechen“ aufgehoben wird. Ändern Sie den folgenden Programmcode, um dieses Verhalten herbeizuführen:

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

7.9.9 Dynamisches Verhalten

Dynamisches Verhalten in Openness

Die Openness-Benutzer können Objekte und ihre Attribute während der Laufzeit festlegen. Sie können außerdem Aktionen aufrufen und bidirektional in der Hierarchie zu zugehörigen Objekten navigieren. Um dynamisches Verhalten zu unterstützen, werden von Openness-Objekten die folgenden Schnittstellen bereitgestellt und implementiert:

IEngineeringInstance

Die Schnittstelle, die im Objektmodell Aufwärtsnavigation ermöglicht.

Eigenschaften:

- IEngineeringObject Parent – übergeordnetes Objekt einer Instanz.

IEngineeringCompositionOrObject

Die Schnittstelle wird von Engineering-Objekten und Zusammenstellungen implementiert.

IEngineeringObject

Die Schnittstelle wird von der Mehrheit der Objekte implementiert, die für die Openness-Benutzer verfügbar sind. Die Schnittstelle IEngineeringObject wird von IEngineeringCompositionOrObject und IEngineeringInstance abgeleitet:

Methoden:

- `IList<EngineeringCreateInfo> GetCreationInfos(string compositionName)` – ruft die Liste der für das Objekt verfügbaren Zusammenstellungsinfos ab.
- `IEngineeringObject Create(string compositionName, Type type, IEnumerable<KeyValuePair<string, object>> parameters)` – erstellt ein IEngineeringObject des angegebenen Typs, das mit Werten wie dem angegebenen compositionName innerhalb der Parameter initialisiert wird.
- `object GetAttribute(string name)` – ruft ein Attribut mit dem angegebenen Namen ab.
- `IList<EngineeringAttributeInfo> GetAttributeInfos()` – gibt eine Sammlung von EngineeringAttributeInfo-Objekten zurück, die die verschiedenen Attribute zu diesem Objekt beschreiben.
- `IList<object> GetAttributes(IEnumerable<string> names)` – ruft eine Liste von Attributen für die angegebenen Namen ab.
- `IEngineeringCompositionOrObject GetComposition(string name)` – ruft ein IEngineeringCompositionOrObject mit dem angegebenen Namen ab.
- `IList<EngineeringInvocationInfo> GetInvocationInfos()` – gibt eine Sammlung von EngineeringAttributeInfo-Objekten zurück, die die verschiedenen Aktionen zu diesem Objekt beschreiben.
- `object Invoke(string name, IEnumerable<KeyValuePair<Type, object>> parameters)` – ruft die von der aktuellen Instanz repräsentierte Methode unter Verwendung der angegebenen Parameter auf.
- `void SetAttribute(string name, object value)` – legt ein Attribut mit dem angegebenen Namen und dem angegebenen Wert fest.
- `void SetAttributes(IEnumerable<KeyValuePair<string, object>> attributes)` – legt die Attribute mit den angegebenen Namen und den angegebenen Werten wie in den Attributen angegeben fest.
- `EngineeringObjectHandle GetHandle()` – ruft den eindeutigen Handle des Objekts ab.

IEngineeringRoot

Diese Schnittstelle wird von TiaPortal-Objekten implementiert und gibt an, dass der Benutzer am obersten Objekt in der Hierarchie ist. IEngineeringRoot wird von IEngineeringObject, IEngineeringCompositionOrObject und IEngineeringInstance abgeleitet.

Methoden:

- `IEngineeringObject GetObject(EngineeringObjectHandle objectHandle)` – ruft das Objekt aus einem Handle ab.

IEngineeringComposition

Die Schnittstelle wird von der Mehrheit der Objekte implementiert, die für die Openness-Benutzer verfügbar sind. Die Schnittstelle IEngineeringObject wird von IEngineeringCompositionOrObject und IEngineeringInstance abgeleitet: Neben den unter Mit Zusammenstellungen arbeiten (Seite 100) beschriebenen Eigenschaften und Methoden werden die folgenden Methoden unterstützt.

Methoden:

- IEngineeringObject Create(Type type, IEnumerable<KeyValuePair<string, object>> parameters) – erstellt ein IEngineeringObject des angegebenen Typs, das mit Werten wie in den Parametern angegeben initialisiert wird.
- IList<EngineeringCreateInfo> GetCreationInfos() – ruft die Sammlung von EngineeringCreateInfo-Objekten ab, die die verschiedenen CreateInfos zu diesem Objekt beschreiben.
- IList<EngineeringInvocationInfo> GetInvocationInfos() – gibt eine Sammlung von EngineeringAttributeInfo-Objekten zurück, die die verschiedenen Aktionen zu diesem Objekt beschreiben.
- object Invoke(string name, IEnumerable<KeyValuePair<Type, object>> parameters) – ruft die von der aktuellen Instanz repräsentierte Methode unter Verwendung der angegebenen Parameter auf.

IEngineeringObjectAssociation

Diese Schnittstelle wird von der Sammlung zugehöriger Engineering-Objekte implementiert.
 Diese Schnittstelle wird von IEngineeringAssociation und IEngineeringInstance abgeleitet.
 Siehe Mit Zuweisungen arbeiten (Seite 100)

IEngineeringServiceProvider

Die Schnittstelle wird von Engineering-Objekten implementiert, die Dienste bereitstellen können.

Methoden:

- T GetService<T>() where T : class, IEngineeringService – ruft eine Instanz vom Typ T ab.
- IList<EngineeringServiceInfo> GetServiceInfos() – gibt eine Sammlung von EngineeringServiceInfo-Objekten zurück, die die verschiedenen Dienste zu diesem Objekt beschreiben.

IEngineeringService

Die Schnittstelle stellt einen Engineering-Dienst dar, der von IEngineeringServiceProvider bereitgestellt wird.

EngineeringObjectHandle

Die Struktur, die einen eindeutigen Objekt-Handle liefert.

Siehe auch

Mit Zusammensetzungen arbeiten (Seite 100)

Arbeiten mit Assoziationen (Seite 100)

7.9.10 Arbeiten mit Assoziationen

Auf Zuordnungen zugreifen

Eine Zuordnung beschreibt die Beziehung zwischen zwei oder mehr Objekten auf Typebene.

TIA Portal Openess unterstützt den Zugriff auf Zuordnungen über den Index und "foreach"-Schleifen. Direktzugriff, beispielsweise über string name, wird nicht unterstützt.

Attribute

Die folgenden Attribute sind verfügbar:

- int Count
- bool IsReadonly
- IEngineeringObject Parent
- retType this [int index] { get; }

Methoden

TIA Portal Openess unterstützt die folgenden Methoden:

- int IndexOf (type): Gibt den Index in der Zuordnung für eine übertragene Instanz zurück.
- bool Contains (type): Ermittelt, ob die übertragene Instanz in der Zuordnung enthalten ist.
- IEnumator GetEnumator <retType>(): Kommt in foreach-Schleifen zum Einsatz und ermöglicht den Zugriff auf ein Objekt.
- void Add (type)¹: Fügt die übertragene Instanz der Zuordnung hinzu.
- void Remove (type)¹: Entfernt die übertragene Instanz aus der Zuordnung.

¹: Wird nicht von allen Zuordnungen unterstützt.

7.9.11 Mit Zusammensetzungen arbeiten

Zusammensetzungen aufrufen

Eine Zusammensetzung ist der Sonderfall einer Zuordnung. Eine Zusammensetzung drückt eine semantische Beziehung zwischen zwei Objekten aus, wobei eines der Objekte ein Teil des anderen ist.

Attribute

Die folgenden Attribute sind verfügbar:

- int Count
- bool Isreadonly
- IEngineeringObject Parent
- retType this [int index] {get; }: Indizierter Zugriff auf ein Objekt der Zusammensetzung.
Diese Art von Zugriff darf nur gezielt genutzt werden, da jede indizierte Zugriffsoperation über Prozessgrenzen hinausgeht.

Methoden

TIA Portal Openness unterstützt die folgenden Methoden:

- retType Create (id, ...): Erzeugt eine neue Instanz und fügt diese Instanz der Zusammensetzung hinzu.
Die Signatur der Methode ist abhängig von der Art und Weise, in der die Instanz erzeugt wird. Diese Methode wird nicht von allen Zusammensetzungen unterstützt.
- type Find (id, ...): Durchsucht eine Zusammensetzung nach der Instanz mit der übertragenen ID.
Die Suche ist nicht rekursiv. Die Signatur der Methode ist abhängig von der Weise, in der nach der Instanz gesucht wird. Diese Methode wird nicht von allen Zusammensetzungen unterstützt.
- IEnumator<retType> GetEnumerator (): Kommt in foreach-Schleifen zum Einsatz und ermöglicht den Zugriff auf ein Objekt.
- Delete (type)¹: Löscht die von der aktuellen Objektreferenz spezifizierte Instanz.
- int IndexOf (type): Gibt den Index in der Zusammensetzung für eine übertragene Instanz zurück.
- bool Contains (type): Ermittelt, ob die übertragene Instanz in der Zusammensetzung enthalten ist.
- void Import(string path, ImportOptions importOptions)¹: Wird für jede Zusammensetzung verwendet, die importierbare Typen enthält.
Jede Importsignatur enthält einen Konfigurationsparameter des Typs "ImportOptions (Seite 780)" ("None", "Overwrite"), mit dem der Benutzer das Importverhalten steuert.

¹: Nicht von allen Zusammensetzungen unterstützt.

7.9.12 Objektgleichheit prüfen

Verwendung

Als Anwender einer TIA Portal Openness API können Sie mittels Programmcode Objekte auf Gleichheit prüfen:

- Sie prüfen dabei mit dem Operator `==`, ob zwei Objektreferenzen gleich sind.
- Mit der Methode `System.Object.Equals()` prüfen Sie, ob beide Objekte in Bezug auf das TIA Portal wirklich identisch sind.

Programmcode

Um auf Objektreferenztypen zu prüfen, ändern Sie den folgenden Programmcode:

```
...
//Composition
DeviceComposition sameCompA = project.Devices;
DeviceComposition sameCompB = project.Devices;
if (sameCompA.Equals(sameCompB))
{
    Console.WriteLine("sameCompA is equal to sameCompB");
}
if (!(sameCompA == sameCompB))
{
    Console.WriteLine("sameCompA is not reference equal to sameCompB");
}
DeviceComposition sameCompAsA = sameCompA;
if (sameCompAsA.Equals(sameCompA))
{
    Console.WriteLine("sameCompAsA is equal to sameCompA");
}
if (sameCompAsA == sameCompA)
{
    Console.WriteLine("sameCompAsA is reference equal to sameCompA");
}
MultiLingualGraphicComposition notSameComp = project.Graphics;
if (!sameCompA.Equals(notSameComp))
{
    Console.WriteLine("sameCompA is not equal to notSameComp");
}
```

7.9.13 Lese-Operationen für Attribute

Gruppenoperationen und Standardleseoperationen für Attribute

TIA Portal Openness unterstützt den Zugriff auf Attribute über die folgenden Methoden, die auf Objektebene verfügbar sind:

- Gruppenoperation für Lesezugriff
- Standardleseoperationen

Programmcode für Gruppenoperationen

```
//Exercise GetAttributes and GetAttributeNames
//get all available attributes for a device,
//then get the names for those attributes, then display the results.
private static void DynamicTest(Project project)
{
    Device device = project.Devices[0];
    IList<string> attributeNames = new List<string>();
    IList<EngineeringAttributeInfo> attributes =
((IEngineeringObject)device).GetAttributeInfos();
    foreach (EngineeringAttributeInfo engineeringAttributeInfo in attributes)
    {
        string name = engineeringAttributeInfo.Name;
        attributeNames.Add(name);
    }
    IList<object> values = ((IEngineeringObject)device).GetAttributes(attributeNames);
    for (int i = 0; i < attributes.Count; i++)
    {
        Console.WriteLine("attribute name: " + attributeNames[i] + " value: " + values[i]);
    }
}
```

Gruppenoperation für Lesezugriff

Diese Methode ist für jedes Objekt verfügbar:

```
public abstract IList<object> GetAttributes(IEnumerable<string>
names);
```

Standardleseoperationen

Die folgenden Operationen sind verfügbar:

- Namen der verfügbaren Attribute abrufen:
Verwenden Sie die Methode GetAttributeInfos() (Seite 107) auf einem IEngineeringObject.
- Generische Methode zum Lesen eines Attributs
`public abstract object GetAttribute(string name);`

Hinweis

Dynamische Attribute werden in IntelliSense nicht angezeigt, weil ihre Verfügbarkeit vom Status der Objektinstanz abhängig ist.

7.9.14 Transaktionsbehandlung

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Operation

Eine innerhalb eines zugehörigen TIA Portal-Prozesses geöffnete Persistenz (Projekt, Bibliothek usw.) kann durch eine TIA Portal Openness-Client-Anwendung geändert werden. Sie können diese Änderung durch eine einzelne Operation oder durch eine Reihe von Operationen herbeiführen. Während der Aktivität ist es sinnvoll, diese Operationen in einer einzelnen Undo-Einheit zu gruppieren, um logischere Arbeitsabläufe zu erhalten. Darüber hinaus ergeben sich durch das Gruppieren von Operationen in einer einzelnen Undo-Einheit auch Leistungsvorteile. Um dies zu unterstützen, bietet die Klasse "ExclusiveAccess" die Methode "Transaction(ITransactionSupport persistence, string undoDescription)". Der Aufruf aus dieser Methode führt zur Instanziierung eines neuen beendbaren Objekts des Typs "Transaction". Sie müssen eine Beschreibung des Inhalts der Transaktion liefern (das Textattribut darf nicht Null oder leer sein). Solange diese Instanz nicht beendet wurde, werden alle Operationen von Client-Anwendungen in einer einzelnen Undo-Einheit innerhalb des zugehörigen TIA Portal-Prozesses gruppiert.

Ändern Sie den folgenden Programmcode, um eine Instanz des Typs "Transaction" zu erfassen:

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

Hinweis

Verwenden Sie zum Instanziieren einer "Transaction" eine "using"-Anweisung. So stellen Sie sicher, dass die Transaktion selbst beim Auftreten von Ausnahmen ordnungsgemäß beendet wird, um so die Transaktion rückgängig zu machen.

Konsistente Umsetzung oder Rückgängigmachung

Mithilfe einer "Transaction" in einer Client-Anwendung stellen Sie sicher, dass es einen vorhersehbaren Weg gibt, einen Satz von Änderungen wirksam werden zu lassen oder rückgängig zu machen. Ihre Client-Anwendung muss entscheiden, ob Änderungen an einer Persistenz wirksam werden sollen oder nicht. Hierzu muss Ihre Anwendung anfordern, dass die Änderungen im Geltungsbereich einer geöffneten Transaktion wirksam werden, wenn die Transaktion durch Aufrufen der Methode "Transaction.CommitOnDispose()" beendet wird. Wenn diese Methode im Codeablauf nie aufgerufen wird, werden die Änderungen im Geltungsbereich der geöffneten Transaktion bei Beendigung der Transaktion automatisch rückgängig gemacht.

Wenn nach Stellung dieser Anforderung eine Ausnahme auftritt, werden alle Änderungen im Geltungsbereich einer geöffneten Transaktion bei Beendigung der Transaktion dennoch rückgängig gemacht.

Ändern Sie den folgenden Programmcode, um eine einzelne Undo-Einheit in dem angehängten TIA Portal mit zwei "Create"-Änderungen zu erzeugen:

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

Einschränkungen

Folgende Aktionen sind innerhalb einer Transaktion nicht zulässig. Der Aufruf dieser Aktionen führt zu einer wiederherstellbaren Ausnahme:

- Compile
- Go Online
- Go Offline
- ProjectText Import
- ProjectText Export
- Open Global Library
- Close Global Library
- Project Create
- Project Open
- Project OpenWithUpgrade
- Project Save
- Project SaveAs
- Project Close

Undo-Verhalten

Von einer TIA Portal Openess-Client-Anwendung ausgeführte Aktionen können in Undo-Einheiten innerhalb des angehängten TIA Portal-Prozesses resultieren. Jeder dieser Undo-Einträge wird unter einem Ortseintrag gruppiert. Dieser Ortseintrag setzt sich aus den folgenden Informationen aus der Client-Anwendung zusammen:

- der Assembly-Titel aus den Manifestdaten, wenn verfügbar, andernfalls der Prozessname
- die Prozess-ID
- die SID (Openess-Session-ID der TIA-Portal-Instanz der Client-Anwendung)
- optional ein Hinweis darauf, dass der Client-Prozess noch läuft

Diese Einträge gehören zu einer der folgenden beiden Arten:

1. Die Operationen, die als Resultat der Verwendung einer "Transaction" in einer Undo-Transaktion zusammengefasst werden, haben die Beschreibung wie von der Client-Anwendung bereitgestellt, als die "Transaction" instanziert wurde.

- Undo-Eintrag für eine laufende Client-Anwendung:

 MySuperApplication ,PID: 4704, SID: 4, running
 My Operation

- Undo-Eintrag für eine gestoppte Client-Anwendung:

 MySuperApplication, PID: 4704, SID: 4
 My Operation

2. Für die Operationen, die individuell ausgeführt werden, existieren individuelle Undo-Einträge zur Beschreibung der Operation, wie im entsprechenden Metadatenbefehl definiert.

- Undo-Eintrag für eine laufende Client-Anwendung:

 MySuperApplication ,PID: 11224, SID: 3, running
 Create Folder "My Folder 2"
 Create Folder "My Folder 1"

- Undo-Eintrag für eine gestoppte Client-Anwendung:

 MySuperApplication, PID: 11224, SID: 3
 Create Folder "My Folder 2"
 Create Folder "My Folder 1"

7.9.15 Ein Objekt DirectoryInfo/FileInfo erstellen

Anwendung

Die Instanzen der Klassen `DirectoryInfo` und `FileInfo` müssen einen absoluten Pfad enthalten. Andernfalls führen die Methoden, die die Objekte `DirectoryInfo` oder `FileInfo` verwenden, zu einer Ausnahme.

Programmcode

Um ein Objekt `DirectoryInfo` oder `FileInfo` zu erstellen, ändern Sie den folgenden Programmcode:

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

//Create a FileInfo object
//please adapt the path and the extension apx to the installed version of TIA Portal
string fileName = @"D:\Test\Project 1\Project 1.apx";
FileInfo fileInfo = new FileInfo(fileName);
...
```

7.9.16 Unterstützung der Selbstbeschreibung für Attribute, Navigatoren, Aktionen und Dienste

Verwendung

In TIA Portal Openness beschreibt jeder `IEngineeringServiceProvider` der TIA Portal Openness API seine Fähigkeiten für potentielle Aufrufe.

Unterstützung der Selbstbeschreibung für IEngineeringObject

Methodename	Rückgabewerte
GetCompositionInfos	Gibt eine Sammlung von Objekten des Typs EngineeringCompositionInfo zurück, die die verschiedenen Zusammensetzungen dieser Objekte beschreiben. EngineeringCompositionInfo wird nachstehend beschrieben.
GetAttributeInfos	Gibt eine Sammlung von Objekten des Typs EngineeringAttributeInfo zurück, die die verschiedenen Attribute dieser Objekte beschreiben. EngineeringAttributeInfo wird nachstehend beschrieben.
GetInvocationInfos	Gibt eine Sammlung von Objekten des Typs EngineeringInvocationInfo zurück, die die verschiedenen Aktionen dieser Objekte beschreiben. EngineeringInvocationInfo wird nachstehend beschrieben.

Unterstützung der Selbstbeschreibung für IEngineeringServiceProvider

Methodename	Rückgabewerte
GetServiceInfos	Gibt eine Sammlung von Objekten des Typs EngineeringServiceInfo zurück, die die verschiedenen Dienste dieser Objekte beschreiben. EngineeringServiceInfo wird nachstehend beschrieben.

Unterstützung der Selbstbeschreibung für IEngineeringComposition

Methodename	Rückgabewerte
GetCreationInfos	Gibt eine Sammlung von Objekten des Typs EngineeringCreateInfo zurück, die die verschiedenen Aktionen dieser Objekte beschreiben. EngineeringCreateInfo wird nachstehend beschrieben.

Klasse EngineeringCompositionInfo

Attributname	Rückgabewerte
Name	Name der Zusammensetzung

Klasse EngineeringAttributeInfo

Attributname	Rückgabewerte
AccessMode	Die vom Attribut unterstützte Zugriffsebene. Dieses Attribut ist kombinierbar und wird nachstehend ausführlich beschrieben.
Name	Name des Attributs.

Klasse EngineeringInvocationInfo

Attributname	Rückgabewerte
Name	Name der Aktion.
ParameterInfos	Eine Sammlung von Objekten des Typs EngineeringInvocationParameterInfo, die von der Aktion möglicherweise benötigte Parameter beschreiben. EngineeringInvocationParameterInfo wird nachstehend beschrieben.

Klasse EngineeringServiceInfo

Attributname	Rückgabewerte
Type	Der Diensttyp als System.Type-Objekt.

Enum AccessMode

Enumerationswert	Rückgabewerte
None	Keine gültige Option.
Read	Das Attribut kann gelesen werden.
Write	Das Attribut kann geschrieben werden.

Klasse EngineeringInvocationParameterInfo

Attributname	Rückgabewerte
Name	Name des Parameters
Type	Typ des Parameters als System.Type-Objekt

Klasse EngineeringCreationInfo

Attributname	Rückgabewerte
Type	Die Eigenschaft Type gibt ein Objekt System.Type zurück.
ParameterInfos	Die Eigenschaft ParameterInfos gibt eine Sammlung von Objekten EngineeringCreationParameterInfo zurück.

Klasse EngineeringCreationParameterInfo

Attributname	Rückgabewerte
Name	Name des betreffenden Parameters

Programmcode

AccessMode ist eine Merker-Enumeration, ihre Werte können wie der folgende Programmcode kombiniert werden:

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

Ändern Sie den folgenden Programmcode, um alle Attribute eines IEngineeringObject zu suchen und Änderungen am Zugriffsmodus dieser Attribute vorzunehmen.

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

```

Ändern Sie folgenden Programmcode dafür, wie Create und GetCreationInfos gemeinsam verwendet werden sollen.

```
...
Project project = tiaPortal.Projects.Open(projectPath);
IEngineeringComposition deviceGroupComposition = project.DeviceGroups;
EngineeringCreateInfo deviceGroupCreateInfo = deviceGroupComposition.GetCreationInfos()
[0];
EngineeringCreationParameterInfo deviceGroupCreateInfo =
deviceGroupCreateInfo.ParameterInfos[0];
string deviceGroupCreationParameterName = deviceGroupCreateInfo.Name;
string groupName = "Example";
IDictionary<string, object> createParameters = new Dictionary<string, object>
{
    {deviceGroupCreationParameterName, groupName }
};
IEngineeringObject group = deviceGroupComposition.Create(deviceGroupCreateInfo.Type,
createParameters);
...

```

7.10 Funktionen der Projekte und Projektdaten

7.10.1 Projekt öffnen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden. Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Das zu öffnende Projekt ist in keiner anderen Instanz des TIA Portals geöffnet.

Hinweis

Rückgängigmachen eines Projekt-Upgrades

Wenn Sie das Upgrade eines Projekts auf V14 SP1 rückgängig machen, nachdem Sie das Projekt mit TIA Portal Openness verbunden haben, treten Konflikte auf.

Verwendung

Zum Öffnen eines Projekts verwenden Sie die Methode `Projects.Open`. Geben Sie in der Methode `Projects.Open` einen Pfad zu dem gewünschten Projekt ein.

Die Methode `Projects.Open` greift ausschließlich auf Projekte zu, die mit der aktuellen Version des TIA Portal s angelegt oder auf die aktuelle Version hochgerüstet wurden. Wenn Sie mit der Methode `Projects.Open` auf ein Projekt einer Vorgängerversion zugreifen, wird eine Ausnahme zurückgegeben. Verwenden Sie die Methode `OpenWithUpgrade` zum Öffnen von Projekten, die mit älteren Versionen von TIA Portal angelegt wurden.

Hinweis

Kein Zugriff auf schreibgeschützte Projekte

TIA Portal Openness kann nur auf Projekte mit Lese- und Schreibrechten zugreifen.

Programmcode

Um ein Projekt zu öffnen, ändern Sie den folgenden Programmcode:

```
Project project =tiaPortal.Projects.Open(new FileInfo(@"D:\Project_1\Project_1.apXX"));
if (project != null)
{
    try
    {
        ...
    }
    finally
    {
        project.Close();
    }
}
```

Öffnen eines UMAC-geschützten Projekts

Sie können auch ein UMAC-geschütztes Projekt öffnen. Die Überlast der Funktion Open benötigt einen zusätzlichen Parameter vom Typ UmacDelegate. Dieser zusätzliche Parameter ermöglicht es dem Aufrufer, einen Handler anzugeben, der während der UMAC-Authentifizierung verwendet werden soll. Der neue UmacDelegate wird mit einer Methode implementiert, die einen Parameter vom Typ UmacCredentials enthält. UmacCredentials hat zwei Eigenschaften, 'Name' vom Typ String und 'Type' vom Typ UmacUserType sowie eine Methode SetPassword mit einem Parameter vom Typ SecureString. Durch die Verwendung von UmacUserType.Project wird auf einen UMAC-Projektumfang hingewiesen, während durch die Verwendung von UmacUserType.Global auf einen UMAC-Anwendungsumfang hingewiesen wird (d.h. gesteuert von einem UMAC-Server).

Programmcode

```

...
    Siemens.Engineering.Project project = tiaPortal.Projects.Open(new
    FileInfo(@"D:\Project_3\Project_2.apXX"), MyUmacDelegate);
    if (project != null)
    {
        try
        {
            ...
        }
        finally
        {
            project.Close();
        }
    }
    ...
private static void MyUmacDelegate(UmacCredentials umacCredentials)
{
    SecureString password = ...; // Get password from a secure location
    umacCredentials.Type = UmacUserType.Project;
    umacCredentials.Name = "SomeUser";
    umacCredentials.SetPassword(password);
}
...
}

```

Mehrere Projekte öffnen

Sie können in einer Instanz des TIA Portals ein primäres Projekt und mehrere sekundäre Projekte gleichzeitig öffnen. Sie müssen dann entscheiden, ob ein Projekt als primäres oder sekundäres Projekt geöffnet werden soll. Wird ein Projekt als primäres Projekt geöffnet, wird es in der Projektnavigation angezeigt, wenn die Openness-Anwendung mit einem TIA Portal verknüpft ist. Wird ein Projekt als sekundäres Projekt geöffnet, wird es in der Benutzeroberfläche nicht angezeigt. Sekundäre Projekte werden stets schreibgeschützt geöffnet. Deshalb hat ein Anwender mit Lese- und Schreibrechten für UMAC-geschützte Projekte nur Leserechte, wenn das Projekt als sekundäres Projekt geöffnet wird. Es ist nicht notwendig, dass ein primäres Projekt geöffnet ist, damit ein sekundäres Projekt geöffnet werden kann.

Jedes geöffnete Projekt kann mit Hilfe der Projektzusammensetzung in der Instanz des TIA Portals enumeriert werden. Die Reihenfolge der Projekte in der Zusammensetzung wird durch die Reihenfolge, in der sie geöffnet wurden, bestimmt. Wird ein Projekt geschlossen, wird der Index aller Projekte neu berechnet.

Programmcode

```
TiaPortal tiaPortal = ...;
Project project1 = tiaPortal.Projects.Open(new
FileInfo(@"D:\Project_1\Project_1.apXX"), null, ProjectOpenMode.Primary);
Project project3 = tiaPortal.Projects.Open(new
FileInfo(@"D:\Project_3\Project_3.apXX"), null, ProjectOpenMode.Secondary);
bool isPrimary = project3.IsPrimary
```

Mit älteren Versionen angelegte Projekte öffnen

Verwenden Sie die Methode `OpenWithUpgrade` zum Öffnen eines Projekts, das mit der älteren Version des TIA Portals angelegt wurde. Die Methode legt ein neues, aktualisiertes Projekt an und öffnet es.

Wenn Sie auf ein mit einer älteren Version erstelltes Projekt zugreifen, wird eine Ausnahme zurückgegeben.

Hinweis

Wenn Sie auf ein mit der aktuellen Version erstelltes Projekt zugreifen, wird das Projekt geöffnet.

Programmcode

Ändern Sie den folgenden Programmcode zum Öffnen eines Projekts über die Methode `OpenWithUpgrade`:

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

Programmcode für ein UMAC-geschütztes Projekt

Sie können auch ein UMAC-geschütztes Projekt öffnen, das mit einer Vorgängerversion von TIA Portal angelegt wurde. Eine Überlastfunktion von `OpenWithUpgrade` benötigt einen zusätzlichen Parameter vom Typ `UmacDelegate`. `OpenWithUpgrade` wird auch bei sekundären Projekten unterstützt.

```

...
Siemens.Engineering.Project project = tiaPortal.Projects.OpenWithUpgrade(new
FileInfo(@"D:\Project_1\Project.apXX"), MyUmacDelegate);
...

```

7.10.2 Projekt anlegen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)

Verwendung

Projekte können über die TIA Portal Openness API erstellt werden

- durch Aufrufen der Methode Create auf ProjectComposition
- durch Aufrufen der Methode Create auf IEngineeringComposition

ProjectComposition.Create

Ändern Sie folgenden Programmcode:

```

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

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

```

Entsprechend diesem Beispiel

- wird ein Ordner "D:\TiaProjects\MyProject" erzeugt.
- wird eine Projektdatei "D:\TiaProjects\MyProject\MyProject.aPXX" erstellt.

Hinweis

Parameter targetDirectory

Der Parameter targetDirectory kann auch einen UNC (Universal Naming Convention)-Pfad darstellen, von daher kann ein Projekt auch auf einem freigegebenen Laufwerk im Netzwerk erstellt werden.

IEngineeringComposition.Create

Ändern Sie folgenden Programmcode:

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

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

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

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

Entsprechend diesem Beispiel

- wird ein Ordner "D:\TiaProjects\MyProject" erzeugt.
- wird eine Projektdatei "D:\TiaProjects\MyProject\MyProject.aPXX" mit den Projektattributen Autor gleich "Bob" und Kommentar gleich "This project was created with openness" erstellt.

Parameter zum Erstellen eines Projekts mit optionalen Projektattributen

Parameter	Datentyp	Obligato- risch	Beschreibung
Author	String	Nein	Autor eines Projekts.
Comment	String	Nein	Kommentar zum Projekt.
Name	String	Ja	Name eines Projekts.
TargetDirecto- ry	DirectoryInfo	Ja	Verzeichnis, in dem der erstellte Projektordner ent- halten ist.

7.10.3 Auf allgemeine Einstellungen des TIA Portals zugreifen**Voraussetzung**

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Über TIA Portal Openness können Sie auf allgemeine Einstellungen des TIA Portals zugreifen:

- Aktuelle Sprache der Benutzeroberfläche
- Option "In Projekt suchen", um den Suchindex zu erstellen, der für die Suche in einem Projekt benötigt wird.

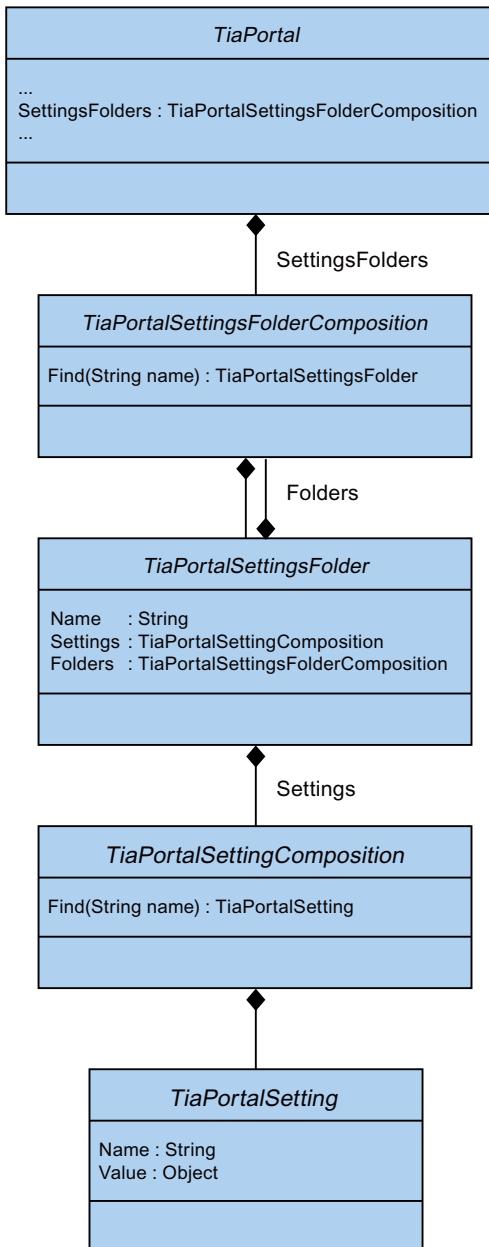
Die folgende Tabelle zeigt die Details der zugänglichen Einstellungen im Abschnitt "Allgemein" der Einstellungen im TIA Portal. Die Instanz `TiaPortalSettingsFolder` hat den Namen "Allgemein".

Name	Datentyp	Schreibbar	Beschreibung
"SearchInProject"	System.Boolean	r/w	Aktiviert oder deaktiviert die Erstellung des Suchindex zum Suchen innerhalb eines Projekts.
"UserInterfaceLanguage"	System.Globalization.CultureInfo	r/w	Zeigt die aktive Sprache der Benutzeroberfläche des TIA Portals oder die Angabe der aktiven Sprache der Benutzeroberfläche an.

Der Zugriff auf diese Einstellungen erfolgt über die Klasse `TiaPortalSettingsFolder`. Die Klasse `TiaPortalSettingsFolder` ist über das Attribut `Settings` der Klasse `TiaPortal` zugänglich.

Die folgende Abbildung zeigt die spezifischen Einstellungen in TIA Portal Openness:

7.10 Funktionen der Projekte und Projektdaten



Programmcode: Im Projekt suchen

Um die Option "In Projekt suchen" zu aktivieren/deaktivieren, ändern Sie den folgenden Programmcode.

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

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

Programmcode: Sprache der Benutzeroberfläche

Um auf die aktuelle Sprache der Benutzeroberfläche zuzugreifen, ändern Sie den folgenden Programmcode:

```
private static void SetUILanguage(Project project)
{
    TiaPortalSettingsFolder generalSettingsFolder =
tiaPortal.SettingsFolders.Find("General");

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

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

Siehe auch

[Projekt öffnen \(Seite 111\)](#)

7.10.4 Projekt archivieren und abrufen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Ein Projekt ist gespeichert.
Siehe Projekt speichern (Seite 137)

Anwendung

Mit dem TIA Portal Openness können Sie den Zustand des geöffneten und gespeicherten Projekts vor dem Durchführen weiterer Änderungen archivieren und später das archivierte Projekt abrufen.

Projekt archivieren

Sie können ein TIA Portal-Projekt mit der am Objekt Siemens.Engineering.Project verfügbaren TIA Openness API archivieren.

```
public void Archive(System.IO.DirectoryInfo targetDirectory, string targetName,  
Siemens.Engineering.ProjectArchivationMode archivationMode)
```

'targetName' ist der Name der erstellten Datei, unabhängig davon, ob sie archiviert oder nicht archiviert ist. Diese Datei kann Dateierweiterungen oder keine Dateierweiterungen enthalten. Für den Wert ProjectArchivationMode als Compressed und DiscardRestorableDataAndCompressed, lautet der archivierte Dateiname wie von Ihnen angegeben. Wenn Sie keine Erweiterung oder außer "zap15_1" bzw. "zap14" usw. keine Erweiterung angeben, kann dieses archivierte Projekt nicht außerhalb der Openness API vom TIA Portal abgerufen werden. Wenn Sie weder Compressed noch DiscardRestorableDataAndCompressed wählen, ist das Ergebnis die Standarddateistruktur eines Projekt der aktuellen Version.

Hinweis

Sie müssen das Projekt vor dem Aufrufen der Archivierungs-API speichern. Wenn das Projekt nicht gespeicherte Änderungen enthält, gibt die Archivierung eine EngineeringTargetInvocationException zurück.

ProjectArchivationMode

Die Enumeration **ProjectArchivationMode** hat vier Werte.

ProjectArchivationMode	Beschreibung
None	<ul style="list-style-type: none"> Mit den Originaldateien werden keine speziellen Aktionen durchgeführt. Der Modus ähnelt dem Vorgang "Speichern unter". In diesem Modus wird keine komprimierte ZIP-Datei erstellt. Der Unterschied zu SaveAs besteht in diesem Fall darin, dass das Archiv den persistenten Speicherort nicht auf den neuen Ordner Archived umstellt, was jedoch bei SaveAs der Fall ist.
DiscardRestorableData	<ul style="list-style-type: none"> Bei der Dateispeicherung wird das Projekt in einer internen Datendatei abgelegt. Diese Datei wird anschließend bei jeder Änderung an den Projektdaten größer. Beim Modus DiscardRestorableData wird diese Datendatei reorganisiert (dabei wird nur die neueste Version der Objekte gespeichert und der Verlauf aus der Datei gelöscht). Zwischenzeitlich angefallene Daten, die Dateien des IM-Verzeichnisses und des Verzeichnisses tmp werden nicht an den Archivspeicherort kopiert. In diesem Modus wird keine komprimierte ZIP-Datei erstellt.
Compressed	<ul style="list-style-type: none"> Die vom Archivierungsprozess erstellte Struktur des TMP-Projektordners wird zu einem ZIP-kompatiblen Archiv komprimiert. Nach Erstellung der ZIP-Datei wird die TMP-Ordnerstruktur gelöscht.
DiscardRestorableDataAndCompressed	<ul style="list-style-type: none"> In der vom Archivierungsprozess erstellten Struktur des TMP-Projektordners werden die wiederherstellbaren Daten verworfen und dann zu einem ZIP-kompatiblen Archiv komprimiert. Nach Erstellung der ZIP-Datei wird die TMP-Ordnerstruktur gelöscht.

Programmcode: Projekt archivieren

Ändern Sie den folgenden Programmcode, um ohne Erweiterung auf das Archivprojekt zuzugreifen:

```
var tiaPortal = new TiaPortal(TiaPortalMode.WithoutUserInterface);
var projectFilePath = @"E:\Sample1\Sample1.ap15_1";
var project = tiaPortal.Projects.Open(new FileInfo(projectFilePath));
var archiveDirectory = @"E:\Archive";
project.Archive(new DirectoryInfo(archiveDirectory), "ArchiveProjectNameWithoutExtension",
ProjectArchivationMode.Compressed);
```

7.10 Funktionen der Projekte und Projektdaten

Ändern Sie den folgenden Programmcode, um mit der Erweiterung '.archive' auf das Archivprojekt zuzugreifen:

```
var tiaPortal = new TiaPortal(TiaPortalMode.WithoutUserInterface);
var projectFilePath = @"E:\Sample1\Sample1.ap15_1";
var project = tiaPortal.Projects.Open(new FileInfo(projectFilePath));
var archiveDirectory = @"E:\Archive";
project.Archive(new DirectoryInfo(archiveDirectory), "ArchiveProjectName.archive",
ProjectArchivationMode.Compressed);
```

Hinweis

Sie können jede beliebige Erweiterung verwenden. Das Verhalten des Archivs ist mit und ohne Erweiterung identisch.

Projekt abrufen

Über die Openness API können Sie ein archiviertes TIA Portal-Projekt abrufen. Der Abruf ist nur mit einem komprimierten Archiv möglich. Bei archivierten Projekten mit 'ProjectArchivationMode.None' oder 'ProjectArchivationMode.DiscardRestorableData' kann der Enumerationswert nicht von dieser API abgerufen werden. Diese API ist am Objekt "Siemens.Engineering.ProjectComposition" verfügbar.

```
public Siemens.Engineering.Project Retrieve(System.IO.FileInfo sourcePath,
System.IO.DirectoryInfo targetDirectory)
```

Zum Abrufen eines UMAC-geschützten Projekts steht zum Abfragen der Anmeldedaten eine überlastete API mit UmacDelegate bereit. Dann wird die folgende Definition verwendet:

```
public Siemens.Engineering.Project Retrieve(System.IO.FileInfo sourcePath,
System.IO.DirectoryInfo targetDirectory, Siemens.Engineering.UmacDelegate umacDelegate)
```

Beim Abrufen wird außerdem ProjectOpenMode unterstützt, für das "Primary" oder "Secondary" angegeben werden kann. Dann wird die folgende API-Definition verwendet:

```
public Siemens.Engineering.Project Retrieve(System.IO.FileInfo sourcePath,
System.IO.DirectoryInfo targetDirectory, Siemens.Engineering.UmacDelegate umacDelegate,
Siemens.Engineering.ProjectOpenMode projectOpenMode)
```

Hinweis

UmacDelegate kann als null übergeben werden, wenn das Projekt nicht geschützt ist.

Wenn das archivierte Projekt aus einer früheren TIA Portal-Version stammt, müssen Sie die RetrieveWithUpgrade API, aufrufen. Dann wird die folgende API-Definition verwendet:

```
public Siemens.Engineering.Project RetrieveWithUpgrade(System.IO.FileInfo sourcePath,
System.IO.DirectoryInfo targetDirectory);
```

Wenn das archivierte Projekt aus einer früheren TIA Portal-Version stammt und UMAC-geschützt ist, steht eine andere überlastete API mit UmacDelegate zur Verfügung. Dann wird die folgende API-Definition verwendet:

```
public Siemens.Engineering.Project RetrieveWithUpgrade(System.IO.FileInfo sourcePath,
System.IO.DirectoryInfo targetDirectory, Siemens.Engineering.UmacDelegate umacDelegate)
```

Wenn das archivierte Projekt aus einer früheren TIA Portal-Version stammt und auch der Projektöffnungsmodus angegeben werden muss, wird die folgende API-Definition verwendet:

```
public Siemens.Engineering.Project RetrieveWithUpgrade(System.IO.FileInfo sourcePath,
System.IO.DirectoryInfo targetDirectory, Siemens.Engineering.UmacDelegate umacDelegate,
Siemens.Engineering.ProjectOpenMode projectOpenMode)
```

Hinweis

UmacDelegate kann als null übergeben werden, wenn das Projekt nicht geschützt ist. Wenn das Projekt geschützt ist, sind die Benutzerdaten eines Administrators erforderlich, um die Aktion RetrieveWithUpgrade durchzuführen.

Programmcode: Projekt abrufen

Um auf das abzurufende Projekt zuzugreifen, ändern Sie folgenden Programmcode:

```
var archivePath = @"E:\Archive\Sample1.zap15_1";
var retrievedProjectsDirectory = @"E:\RetrievedProjects";
var tiaPortal = new TiaPortal(TiaPortalMode.WithoutUserInterface);
tiaPortal.Projects.Retrieve(new FileInfo(archivePath), new
DirectoryInfo(retrievedProjectsDirectory));
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[Projekt speichern \(Seite 137\)](#)

7.10.5 Schreibgeschütztes TIA Portal-Projekt öffnen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Mithilfe von TIA Portal Openness können auch bei einem schreibgeschützten TIA Portal-Projekt bestimmte Vorgänge ausgeführt werden. Sie können auf ein schreibgeschütztes Projekt zugreifen, können jedoch nicht den vollständigen Funktionsumfang nutzen, der einem Benutzer mit Lese-/Schreibzugriff zur Verfügung steht. Beispielsweise kann ein Benutzer, der nur über Leserechte verfügt, mit Openness ein UMAC-geschütztes Projekt wie in Öffnen eines Projekts (Seite 111) beschrieben öffnen. Die Nutzbarkeit dieser Funktion gilt jedoch nicht für Referenzprojekte.

Die Liste der Openness-Funktionen, die Ihnen beim Zugriff auf ein schreibgeschütztes Projekt zur Verfügung stehen, lässt sich in zwei Gruppen unterteilen: in integrierte Aktionen und in aktivierte nicht-verändernde Aktionen.

Integrierte Funktionen

- GetAttribute(s) oder Verwendung der Abruffunktion für ein beliebiges Attribut eines beliebigen Objekts
- GetComposition bei einem Objekt, auf das Zugriff möglich ist
- GetService bei einem Objekt, auf das Zugriff möglich ist
- Find-Aktionen bei einem Objekt, auf das Zugriff möglich ist
- Navigation bei einem Objekt, auf das Zugriff möglich ist
- Feststellen des Vorhandenseins von Objekten, auf die Zugriff möglich ist, und Zugriff auf diese Objekte in Zusammensetzungen und Zuordnungen
- System.Object-Methoden bei einem Objekt, auf das Zugriff möglich ist

Aktivierte nicht-verändernde Aktionen

- Project.Close (...)
- PlcBlock.ShowInEditor ()
- CaxProvider.Export (Device,...)
- CaxProvider.Export (Project,...)

Siehe auch

Projekt öffnen (Seite 111)

7.10.6 Auf Sprachen zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

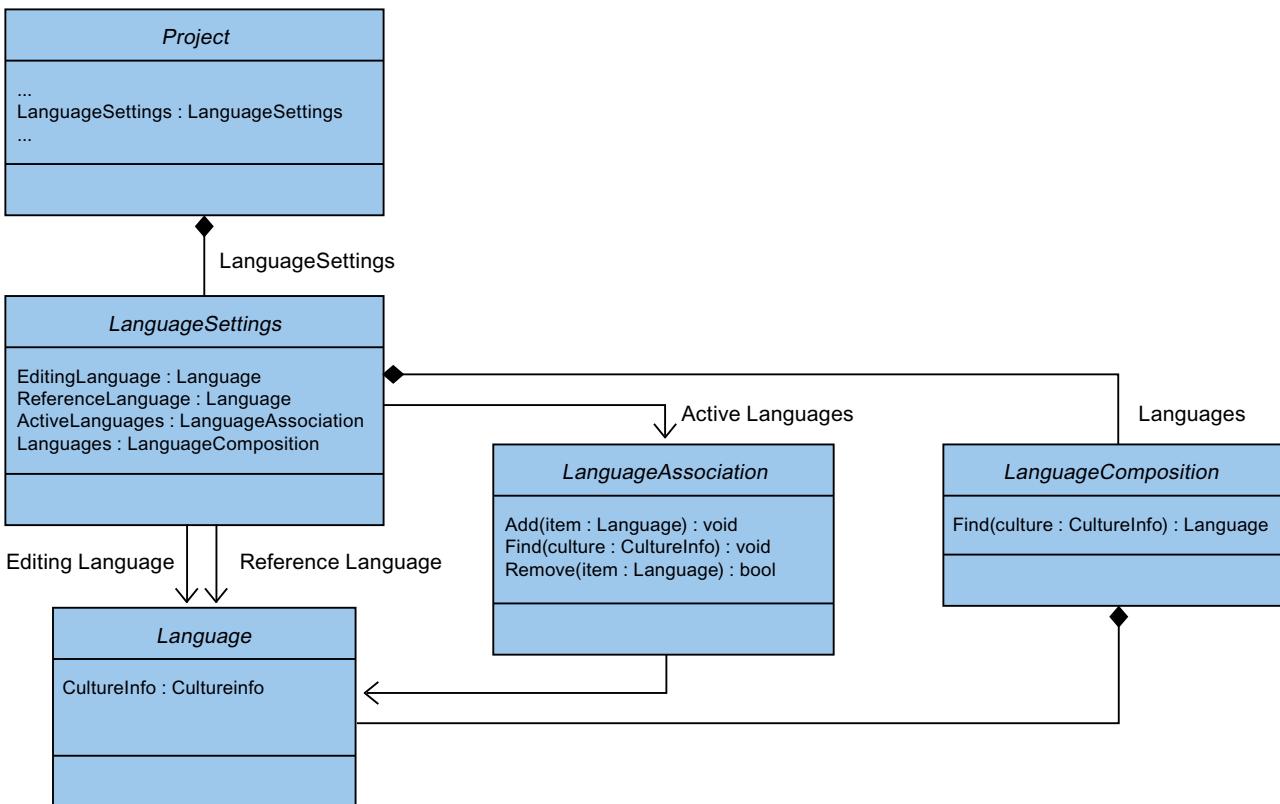
Im TIA Portal können Sie die Projektsprache im Editor "Projektsprachen" festlegen und verwalten.

TIA Portal Openness unterstützt den folgenden Zugriff auf die Projektsprachen:

- Iteration durch unterstützte Sprachen.
- Suchen in der Sammlung unterstützter Sprachen mit Hilfe von `System.Globalization.CultureInfo`.
- Zugriff auf einzelne Sprachen. Jedes Sprachobjekt enthält ein einziges schreibgeschütztes Attribut `Culture` vom Typ `System.Globalization.CultureInfo`..
- Zugriff auf eine Sammlung aktiver Sprachen.
- Suchen in der Sammlung aktiver Sprachen über mit Hilfe von `System.Globalization.CultureInfo`.
- Hinzufügen einer Sprache zu einer Sammlung aktiver Sprachen.
- Entfernen einer Sprache aus einer Sammlung aktiver Sprachen.
- Festlegen einer Bearbeitungssprache.
- Festlegen einer Referenzsprache.

Die Funktionalitäten werden vom Objekt `LanguageSettings` bereitgestellt. Die folgende Abbildung zeigt das Modell in TIA Portal Openness:

7.10 Funktionen der Projekte und Projektdaten

**Programmcode: Sprachen festlegen**

Ändern Sie den folgenden Programmcode, um eine Sprache festzulegen. Wenn Sie über TIA Portal Openness eine inaktive Sprache festlegen, wird die Sprache der Sammlung aktiver Sprachen hinzugefügt.

```

Project project = ...;

LanguageSettings languageSettings = project.LanguageSettings;

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

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

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

Programmcode: Eine aktive Sprache deaktivieren

Um eine aktive Sprache zu deaktivieren, ändern Sie den folgenden Programmcode. Wenn Sie eine Sprache deaktivieren, die als Referenz- oder Bearbeitungssprache verwendet wird, ist die ausgewählte Sprache konsistent mit dem Verhalten in der Benutzeroberfläche.

```
Project project = ...;

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

7.10.7 Objektstruktur und -Attribute ermitteln

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Sie haben mit Ihrer TIA Portal Openness-Anwendung ein Projekt geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Die Navigationsstruktur in der Objekthierarchie können Sie mit der Schnittstelle `IEngineeringObject` ermitteln. Das Ergebnis wird als Liste zurückgegeben:

- Untergeordnete Objekte
- Kindzusammensetzungen
- Alle Attribute

Signatur

Verwenden Sie zum Ermitteln von Attributen die Methode `GetAttributeInfos`.

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

Programmcode: Objekte oder Zusammensetzungen ermitteln

Um alle Namen von Zusammensetzungen anzuzeigen, verwenden Sie den folgenden Programmcode:

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

Wenn Sie den Rückgabewert kennen, ändern Sie den folgenden Programmcode:

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

Programmcode: Attribute ermitteln

Um Attribute eines Objekts mit bestimmten Zugriffsrechten in einer Liste zurückzugeben, ändern Sie folgenden Programmcode:

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

public static string GetNameAttribute(IEngineeringObject obj)
{
    Object nameAttribute = obj.GetAttribute("Name");
    return (string)nameAttribute;
}
```

7.10.8 Auf Software-Ziel zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode

Ändern Sie den folgenden Programmcode, um ein Software-Ziel verfügbar zu machen:

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

Ändern Sie den folgenden Programmcode, um auf die Software-Attribute zuzugreifen:

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

7.10.9 Auf mehrsprachige Texte zugreifen und sie enumerieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Mehrsprachige Texte im TIA Portal sind beispielsweise Project.Comment, PlcTag.Comment usw. In TIA Portal Openness werden die mehrsprachigen Texte vom Objekt MultilingualText dargestellt. Ein Objekt MultilingualText besteht aus MultilingualTextItemComposition.

MultilingualTextItemComposition unterstützt die folgende Find-Methode:

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

Jedes MultilingualTextItem hat die folgenden Attribute:

Attributname	Datentyp	Schreib-bar	Beschreibung
Language	Siemens.Engineering.Language	r/o	Sprache dieses Elements
Text	System.String	r/w	Für diese Sprache angegebener Text

Programmcode: Mehrsprachige Texte festlegen

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

```

Programmcode: Mehrsprachige Texte für Geräte festlegen

Ändern Sie folgenden Programmcode, um mehrsprachige Texte für Geräte und Geräteelemente festzulegen:

```
...
var mltObject = device.GetAttribute("CommentML");
MultilingualText multilingualText = mltObject as MultilingualText;
if (multilingualText != null)
{
    Language englishLanguage = project.LanguageSettings.Languages.Find(new CultureInfo("en-US"));
    MultilingualTextItem multilingualTextItem =
multilingualText.Items.Find(englishLanguage);
    if (multilingualTextItem != null)
    {
        multilingualTextItem.Text = comment;
    }
}
...

```

7.10.10 Aktualisierung der Projekteigenschaft "Simulationsunterstützung"

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Mithilfe von TIA Portal Openness können Sie die Projekteigenschaft "Simulationsunterstützung" aktualisieren, sodass Sie PLC-Programme erstellen können, die auch in einer virtuellen SINUMERIK-Steuerung eingesetzt werden.

7.10 Funktionen der Projekte und Projektdaten

Openness unterstützt die folgenden Attribute, mit deren Hilfe es die Eigenschaft "Simulationsunterstützung" eines TIA Portal-Projekts liest/schreibt. Die API ist am Objekt "Siemens.Engineering.Project" verfügbar.

Attributname	Datentyp	Schreiben	Beschreibung
IsSimulationDuringBlockCompilationEnabled	System.Boolean	ja	Gibt an, ob die Unterstützung für die Simulation bei der Bausteinübersetzung für das Projekt aktiviert ist.

Programmcode

Ändern und verwenden Sie den folgenden Programmcode, um den Attributwert zu lesen/zuschreiben:

```
Project project = ...;
// Read the attribute value
var attributeValue = project.IsSimulationDuringBlockCompilationEnabled;
//Write the attribute value to false
project.IsSimulationDuringBlockCompilationEnabled = false;
```

Siehe auch

[Projekt öffnen \(Seite 111\)](#)

7.10.11 Projektbezogene Attribute lesen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Mit dieser Funktion können Sie projektbezogene Attribute aus der TIA Portal Openness API erhalten. Die gelieferten Informationen umfassen Projektattribute, Projekthistorie und vom Projekt genutzte Produkte.

Projektattribute

Die Projektattribute liefern die folgenden Informationen:

Attributname	Datentyp	Schreibbar	Beschreibung
Author	System.String	r/o	Autor des Projekts
Comment	Siemens.Engineering.MultilingualText	r/o	Kommentar des Projekts

Attributname	Datentyp	Schreibbar	Beschreibung
Copyright	System.String	r/o	Copyright-Hinweis des Projekts
CreationTime	System.DateTime	r/o	Zeitpunkt, zu dem das Projekt angelegt wurde
Family	System.String	r/o	Familie des Projekts
IsModified	System.Boolean	r/o	Gibt wahr aus, wenn das Projekt geändert wurde
LanguageSettings	Siemens.Engineering.LanguageSettings	r/o	Handhabt Projektsprachen
LastModified	System.DateTime	r/o	Zeitpunkt, zu dem das Projekt zuletzt geändert wurde
LastModifiedBy	System.String	r/o	Autor der letzten Änderung
Name	System.String	r/o	Name des Projekts
Path	System.IO.FileInfo	r/o	Absoluter Pfad des Projekts
Size	System.Int64	r/o	Größe des Projekts in KB
Version	System.String	r/o	Version des Projekts

Ändern Sie den folgenden Programmcode, um auf projektbezogene Attribute zuzugreifen:

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

Ändern Sie das folgende Programm, um die Projektsprachen zu enumerieren:

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

Ändern Sie den folgenden Programmcode, um Kommentartext zu erhalten:

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

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

Projekthistorie

Die Projekthistorie ist eine Zusammensetzung von Objekten des Typs `HistoryEntry`, die folgende Informationen enthalten:

Attributname	Datentyp	Schreibbar	Beschreibung
Text	System.String	r/o	Ereignisbeschreibung
DateTime	System.DateTime	r/o	Zeitpunkt, zu dem das Ereignis aufgetreten ist

Ändern Sie den folgenden Programmcode, um `HistoryEntries` zu enumerieren und auf deren Attribute zuzugreifen:

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

Hinweis

Das Textattribut von `HistoryEntry` enthält einen String in der gleichen Sprache wie die Benutzeroberfläche. Wenn eine TIA Portal Openness-Anwendung an ein TIA Portal ohne Benutzeroberfläche angehängt wird, liegt der String immer auf Englisch vor.

Verwendete Produkte

Das Objekt `UsedProduct` enthält die folgenden Informationen:

Attributname	Datentyp	Schreibbar	Beschreibung
Name	System.String	r/o	Name des verwendeten Projekts
Version	System.String	r/o	Version des Produkts

Ändern Sie den folgenden Programmcode, um `UsedProduct` zu enumerieren und auf die Attribute zuzugreifen.

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

7.10.12 Projektgrafik löschen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode

Ändern Sie den folgenden Programmcode, um eine Grafiksammlung zu löschen:

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

7.10.13 Projekt übersetzen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Alle Geräte sind "offline".

Anwendung

Die API-Schnittstelle unterstützt die Übersetzung von Geräten und Programmbausteinen. Das Übersetzungsergebnis wird als Objekt zurückgegeben. In Abhängigkeit vom Objekttyp wird die HW- oder SW- oder HW/SW-Übersetzung zur Verfügung gestellt. Die folgenden Objekttypen werden unterstützt:

- Device - HW & SW
 - Device mit fehlersicherer CPU - SW mit ausgeschalteter F-Aktivierung
- DeviceItem - HW
- CodeBlock - SW
- DataBlock - SW
- HmiTarget - SW

7.10 Funktionen der Projekte und Projektdaten

- PlcSoftware – SW
- PlcType – SW
- PlcBlockSystemGroup – SW
- PlcBlockUserGroup – SW
- PlcTypeSystemGroup – SW
- PlcTypeUserGroup – SW

Hinweis

Format des Zeitstempels

Alle Zeitstempel sind in UTC. Wenn Sie die lokale Uhrzeit anzeigen möchten, können Sie `DateTime.ToLocalTime()`.

Signatur

Verwenden Sie für die Übersetzung die Methode `ICompilable`.

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

Hinweis

Alle Geräte müssen vor dem Start der Übersetzung „offline“ sein.

Programmcode

Um die Softwareänderungen eines Objekts vom Typ `HmiTarget` zu übersetzen, ändern Sie folgenden Programmcode:

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

Um die Softwareänderungen eines Objekts vom Typ `PlcSoftware` zu übersetzen, ändern Sie folgenden Programmcode:

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

Um die Softwareänderungen eines Objekts vom Typ `CodeBlock` zu übersetzen, ändern Sie folgenden Programmcode:

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

Um das Übersetzungsergebnis auszuwerten, ändern Sie folgenden Programmcode:

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

7.10.14 Projekt speichern

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Ein Projekt speichern

- Zum Speichern eines Projekts verwenden Sie die Methode `Save()`.
- Um ein Projekt mit einem anderen Namen oder in einem anderen Verzeichnis zu speichern, verwenden Sie die Methode `SaveAs()`.

Programmcode

Ändern Sie folgenden Programmcode, um ein Projekt zu öffnen und zu speichern:

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

Ändern Sie folgenden Programmcode, um ein Projekt mit einem anderen Namen oder an einem anderen Ort zu speichern:

```
...
    TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
    FileInfo fileInfoExistingProject = new FileInfo(@"D:\SampleProjects
\SampleProject.apXX");
    DirectoryInfo dirInfoSaveAsProject = new DirectoryInfo(@"D:\SampleProjects
\SampleProjectSaveAs");
    Project sampleProject = portal.Projects.Open(fileInfoExistingProject );
    sampleProject.SaveAs(dirInfoSaveAsProject);
...
...
```

7.10.15 Projekt schließen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Sie haben mit Ihrer TIA Portal Openness-Anwendung ein Projekt geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode

Um ein Projekt zu schließen, ändern Sie folgenden Programmcode:

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

7.11 Funktionen für Verbindungen

7.11.1 Konfigurierbare Attribute einer Port-zu-Port-Verbindung

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Projekt öffnen (Seite 111)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Die Attribute einer Portverschaltung befinden sich im Port-Geräteelement. Der Lese- und Schreibzugriff von Attributen über TIA Portal Openness ist mit dem in der UI identisch.

Einstellungen der Portschnittstelle

Die folgenden Attribute werden für die Einstellungen der Portschnittstelle bereitgestellt:

Attributname	Datentyp	Schreibbar	Zugriff	Beschreibung
MediumAttachmentType	MediumAttachment-Type	r/o	Dynamisches Attribut	
CableName	CableName	r/w	Dynamisches Attribut	

7.11 Funktionen für Verbindungen

Attributname	Datentyp	Schreibbar	Zugriff	Beschreibung
AlternativePartnerPorts	Bool	r/w	Dynamisches Attribut	Nur verfügbar, wenn die Funktionalität des Werkzeugwechslers unterstützt wird, z. B. bei der CPU1516.
SignalDelaySelection	SignalDelaySelection	r/w	Dynamisches Attribut	
CableLength	CableLength	r/w	Dynamisches Attribut	
SignalDelayTime	Double	r/w	Dynamisches Attribut	

Die folgenden ENUM-Werte werden für das Attribut `MediumAttachmentType` bereitgestellt:

Wert	Beschreibung
<code>MediumAttachmentType.None</code>	Anbindungstyp kann nicht ermittelt werden.
<code>MediumAttachmentType.Copper</code>	Anbindungstyp ist Kupfer.
<code>MediumAttachmentType.FibreOptic</code>	Anbindungstyp ist Glasfaser.

Die folgenden ENUM-Werte werden für das Attribut `CableName` bereitgestellt:

Wert	Beschreibung
<code>CableName.None</code>	Kein Kabelname angegeben
<code>CableName.FO_Standard_Cable_9</code>	FO-Standardkabel GP (9 µm)
<code>CableName.Flexible_FO_Cable_9</code>	Flexibles FO-Kabel (9 µm)
<code>CableName.FO_Standard_Cable_GP_50</code>	FO-Standardkabel GP (50 µm)
<code>CableName.FO_Trailing_Cable_GP</code>	FO-Schleppkabel / GP
<code>CableName.FO_Ground_Cable</code>	FO-Erdungskabel
<code>CableName.FO_Standard_Cable_62_5</code>	FO-Standardkabel (62,5 µm)
<code>CableName.Flexible_FO_Cable_62_5</code>	Flexibles FO-Kabel (62,5 µm)
<code>CableName.POF_Standard_Cable_GP</code>	POF-Standardkabel GP
<code>CableName.POF_Trailing_Cable</code>	POF-Schleppkabel
<code>CableName.PCF_Standard_Cable_GP</code>	PCF-Standardkabel GP
<code>CableName.PCF_Trailing_Cable_GP</code>	PCF-Schleppkabel / GP
<code>CableName.GI_POF_Standard_Cable</code>	GI-POF-Standardkabel
<code>CableName.GI_POF_Trailing_Cable</code>	GI-POF-Schleppkabel
<code>CableName.GI_PCF_Standard_Cable</code>	GI-PCF-Standardkabel
<code>CableName.GI_PCF_Trailing_Cable</code>	GI-PCF-Schleppkabel

Die folgenden ENUM-Werte werden für das Attribut `SignalDelaySelection` bereitgestellt:

Wert	Beschreibung
<code>SignalDelaySelection.None</code>	
<code>SignalDelaySelection.CableLength</code>	CableLength dient zur Festlegung der Signalverzögerung.
<code>SignalDelaySelection.SignalDelayTime</code>	SignalDelayTime dient zur Festlegung der Signalverzögerung.

Die folgenden ENUM-Werte werden für das Attribut `CableLength` bereitgestellt:

Wert	Beschreibung
<code>CableLength.None</code>	Kabellänge nicht angegeben
<code>CableLength.Length20m</code>	Kabellänge ist 20 m.
<code>CableLength.Length50m</code>	Kabellänge ist 50 m.
<code>CableLength.Length100m</code>	Kabellänge ist 100 m.
<code>CableLength.Length1000m</code>	Kabellänge ist 1000 m.
<code>CableLength.Length3000m</code>	Kabellänge ist 3000 m.

Portoptionen

Die folgenden Attribute werden für Portoptionen bereitgestellt:

Attributname	Datentyp	Schreibbar	Zugriff
<code>PortActivation</code>	Bool	r/w	Dynamisches Attribut
<code>TransmissionRateAndDuplex</code>	<code>TransmissionRateAndDuplex</code>	r/w	Dynamisches Attribut
<code>PortMonitoring</code>	Bool	r/w	Dynamisches Attribut
<code>TransmissionRateAutoNegotiation</code>	Bool	r/w	Dynamisches Attribut
<code>EndOfDetectionOfAccessibleDevices</code>	Bool	r/w	Dynamisches Attribut
<code>EndOfTopologyDiscovery</code>	Bool	r/w	Dynamisches Attribut
<code>EndOfSyncDomain</code>	Bool	r/w	Dynamisches Attribut

Die folgenden ENUM-Werte werden für das Attribut `TransmissionRateAndDuplex` bereitgestellt:

Wert	Beschreibung
<code>TransmissionRateAndDuplex.None</code>	
<code>TransmissionRateAndDuplex.Automatic</code>	Automatisch
<code>TransmissionRateAndDuplex.AUI10Mbps</code>	10 Mbps AUI
<code>TransmissionRateAndDuplex.TP10MbpsHalfDuplex</code>	TP 10 Mbps Halbduplex
<code>TransmissionRateAndDuplex.TP10MbpsFullDuplex</code>	TP 10 Mbps Vollduplex
<code>TransmissionRateAndDuplex.AsyncFiber10MbpsHalfDuplex</code>	Asynchrone Glasfaser 10 Mbit/s Halbduplexmodus
<code>TransmissionRateAndDuplex.AsyncFiber10MbpsFullDuplex</code>	Asynchrone Glasfaser 10 Mbit/s Voll duplexmodus
<code>TransmissionRateAndDuplex.TP100MbpsHalfDuplex</code>	TP 100 Mbps Halbduplex
<code>TransmissionRateAndDuplex.TP100MbpsFullDuplex</code>	TP 100 Mbps Voll duplex
<code>TransmissionRateAndDuplex.FO100MbpsFullDuplex</code>	FO 100 Mbps Voll duplex

Wert	Beschreibung
TransmissionRateAndDuplex.X1000MbpsFullDuplex	X1000 Mbps Vollduplex
TransmissionRateAndDuplex.FO1000MbpsFullDuplexLD	FO 1000 Mbps Vollduplex LD
TransmissionRateAndDuplex.FO1000MbpsFullDuplex	FO 1000 Mbps Vollduplex
TransmissionRateAndDuplex.TP1000MbpsFullDuplex	TP 1000 Mbps Vollduplex
TransmissionRateAndDuplex.FO10000MbpsFullDuplex	FO 10000 Mbps Vollduplex
TransmissionRateAndDuplex.FO100MbpsFullDuplexLD	FO 100 Mbps Vollduplex LD
TransmissionRateAndDuplex.POFPCF100MbpsFullDuplex	POF/PCF 100 Mbps Vollduplex

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

7.12 Funktionen in Bibliotheken

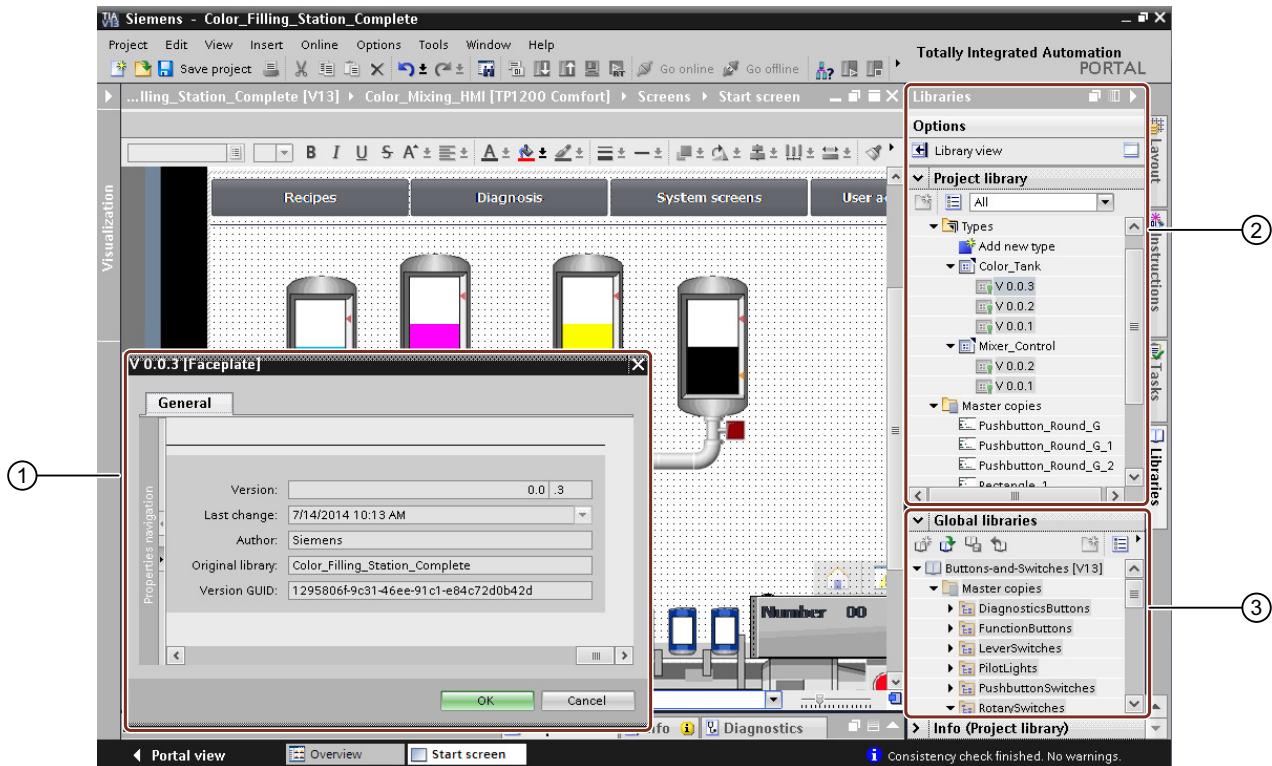
7.12.1 Funktionen auf Objekte und Instanzen

Zugriff auf Typen und Instanzen

Mit der Schnittstelle TIA Portal Openness API können Sie auf Typen, Typversionen und Masterkopien in der Projektbibliothek oder in globalen Bibliotheken zugreifen. Sie können Verbindungen zwischen Typversionen und Instanzen bestimmen. Sie können auch Instanzen im Projekt aktualisieren und Änderungen zwischen einer globalen Bibliothek und der Projektbibliothek synchronisieren. Die Schnittstelle TIA Portal Openness API unterstützt auch den Vergleich von Typversionen und Instanzen.

Funktionen für Objekte und Instanzen

Mit der Schnittstelle TIA Portal Openess API haben Sie Zugriff auf die folgenden Funktionen für Typen, Typversionen, Masterkopien und Instanzen:



- ① Attribute von Typen, Typversionen, Masterkopien und Instanzen anzeigen
- ② Die folgenden Funktionen sind in der Projektbibliothek verfügbar:
 - Instanzen von Typen aktualisieren
 - Typversionen im Projekt instanziieren
 - Innerhalb der Bibliotheksgruppe navigieren
 - Gruppe, Typen, Typversionen und Masterkopien löschen
- ③ Die folgenden Funktionen sind in der globalen Bibliothek verfügbar:
 - Instanzen von Typen aktualisieren
 - Typversion im Projekt instanziieren
 - Innerhalb der Bibliotheksgruppe navigieren

7.12.2 Auf globale Bibliotheken zugreifen

Voraussetzung

- Die Anwendung TIA Portal Openess ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)

Verwendung

Es gibt drei Arten von globalen Bibliotheken.

- Globale Systembibliothek (Siemens.Engineering.Library.SystemGlobalLibrary): Diese globalen Bibliotheken sind Bestandteil der Installation des TIA Portals und verwenden die Dateierweiterung *.asx. Alle globalen Systembibliotheken sind schreibgeschützt.
- Globale Unternehmensbibliothek (Siemens.Engineering.Library.CorporateGlobalLibrary): Diese globalen Bibliotheken wurden von einem Administrator ausgewählt und werden beim Starten des TIA Portals vorab geladen. Alle globalen Unternehmensbibliotheken sind schreibgeschützt.
- Globale Anwenderbibliothek (Siemens.Engineering.Library.UserGlobalLibrary): Diese globalen Bibliotheken wurden von Benutzern des TIA Portals erstellt. Globale Anwenderbibliotheken können entweder nur schreibgeschützt oder mit Lese- und Schreibrechten geöffnet werden.

Wenn eine globale Anwenderbibliothek in einem bestimmten Modus bereits geöffnet ist, kann diese globale Anwenderbibliothek nicht in einem anderen Modus geöffnet werden. Globale Anwenderbibliotheken aus Vorgängerversionen können nur schreibgeschützt geöffnet werden.

Mit TIA Portal Openness geöffnete globale Bibliotheken werden außerdem der Sammlung der globalen Bibliotheken der TIA Portal-Benutzeroberfläche hinzugefügt und in der Oberfläche im TIA Portal angezeigt, sofern die Oberfläche vorhanden ist.

Programmcode: Verfügbare globale Bibliotheken

Um Informationen über sämtliche globale Bibliotheken abzurufen, ändern Sie folgenden Programmcode:

```
TiaPortal tia = ...;
var availableLibraries = tia.GlobalLibraries.GetGlobalLibraryInfos();
foreach (GlobalLibraryInfo info in availableLibraries)
{
    Console.WriteLine(" Library Name: {0}", info.Name);
    Console.WriteLine(" Library Path: {0}", info.Path);
    Console.WriteLine(" Library Type: {0}", info.LibraryType);
    Console.WriteLine(" Library IsOpen: {0}", info.IsOpen);
}
```

Attribute von GlobalLibrary

Wert	Datentyp	Beschreibung
Author	String	Autor der globalen Bibliothek.
Comment	MultilingualText	Kommentar der globalen Bibliothek.
IsReadOnly	Boolean	Wahr, wenn die globale Bibliothek schreibgeschützt ist.

Wert	Datentyp	Beschreibung
IsModified	Boolean	Wahr, wenn der Inhalt der globalen Bibliothek geändert wurde.
Name	String	Name der globalen Bibliothek.
Path	FileInfo	Pfad der globalen Bibliothek.

Attribute von GlobalLibraryInfo

Wert	Rückgabetyp	Beschreibung
IsReadOnly	Boolean	Wahr, wenn die globale Bibliothek schreibgeschützt ist.
isOpen	Boolean	Wahr, wenn die globale Bibliothek bereits geöffnet ist.
LibraryType	GlobalLibraryType	Typ der globalen Bibliothek: <ul style="list-style-type: none"> • System: Globale Systembibliothek • Corporate: Globale Unternehmensbibliothek • User: Globale Anwenderbibliothek
Name	String	Name der globalen Bibliothek.
Path	FileInfo	Pfad der globalen Bibliothek.

Siehe auch

Auf Ordner in einer Bibliothek zugreifen (Seite 155)

7.12.3 Zugriff auf Sprachen der globalen Bibliothek

Voraussetzungen

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Eine Bibliothek ist geöffnet.
Siehe Bibliotheken öffnen (Seite 147)

Anwendung

Sie können mithilfe des Navigators für die Spracheinstellungen auf die Sprachen der globalen Bibliothek zugreifen und sie verwalten.

TIA Portal Openness unterstützt den folgenden Zugriff auf die Sprachen der globalen Bibliothek:

- In unterstützten Sprachen iterativ suchen
- Suchen in der Sammlung unterstützter Sprachen mithilfe von `System.Globalization.CultureInfo`.

- Zugriff auf einzelne Sprachen. Jedes Sprachobjekt enthält ein einziges schreibgeschütztes Attribut Culture vom Typ System.Globalization.CultureInfo.
- Zugriff auf eine Sammlung aktiver Sprachen.
- Suchen in der Sammlung aktiver Sprachen mithilfe von System.Globalization.CultureInfo.
- Hinzufügen einer Sprache zu einer Sammlung aktiver Sprachen.
- Entfernen einer Sprache aus einer Sammlung aktiver Sprachen.
- Festlegen einer Bearbeitungssprache.
- Festlegen einer Referenzsprache.

Attribut von Sprachen der globalen Bibliothek

Sprachen der globalen Bibliothek haben die folgenden Attribute:

Attributname	Datentyp	Schreibbar	Beschreibung
LanguageSettings	Siemens.Engineering.LanguageSettings	schr.-gesch.	Verwaltet Sprachen der globalen Bibliothek

Programmcode: Sprachen der globalen Bibliothek enumerieren

Ändern Sie den folgenden Programmcode, um die Sprache der globalen Bibliothek zu enumerieren:

```
FileInfo m_GlobalLibrarypath = new FileInfo("bla");
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
var globalLibrary = portal.GlobalLibraries.Open(m_GlobalLibrarypath, OpenMode.ReadOnly);
LanguageSettings languageSettings = globalLibrary.LanguageSettings;
```

Programmcode: Auf Spracheinstellung zugreifen

Ändern Sie den folgenden Programmcode, um auf die Spracheinstellung der globalen Bibliothek zuzugreifen:

```
FileInfo m_GlobalLibrarypath = new FileInfo("bla");
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
var globalLibrary = portal.GlobalLibraries.Open(m_GlobalLibrarypath, OpenMode.ReadOnly);
LanguageComposition languages = globalLibrary.LanguageSettings.Languages;
foreach (Language language in languages)
{
// Work with this language
}
```

Programmcode: Festlegen der Sprachen der globalen Bibliothek

Ändern Sie den folgenden Programmcode, um Sprachen der globalen Bibliothek festzulegen. Wenn Sie über TIA Portal Openness eine neue unterstützte Sprache hinzufügen, wird die Sprache der Sammlung aktiver Sprachen hinzugefügt.

```
FileInfo m_GlobalLibrarypath = new FileInfo("bla");
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
var globalLibrary = portal.GlobalLibraries.Open(m_GlobalLibrarypath, OpenMode.ReadOnly);
LanguageSettings languageSettings = globalLibrary.LanguageSettings;
LanguageComposition supportedLanguages = languageSettings.Languages;
LanguageAssociation activeLanguages = languageSettings.ActiveLanguages;
Language supportedGermanLanguage = supportedLanguages.Find(CultureInfo.GetCultureInfo("de-DE"));
activeLanguages.Add(supportedGermanLanguage);
languageSettings.EditingLanguage = supportedGermanLanguage;
languageSettings.ReferenceLanguage = supportedGermanLanguage;
```

Hinweis

Wenn Sie der globalen Bibliothek Sprachen hinzufügen oder sie darin ändern, bewirkt dies keine Änderungen an Sprachen von veröffentlichten Versionen der globalen Bibliothek. Dies gilt auch für die Aktualisierung von Sprachen der globalen Bibliothek über die Benutzeroberfläche (Spracheditor) oder den Navigator für die Spracheinstellungen (LanguageSettings).

7.12.4 Bibliotheken öffnen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Sie haben mit Ihrer TIA Portal Openness-Anwendung ein Projekt geöffnet. Diese Voraussetzung gilt nur für den Zugriff auf Projektbibliotheken.
Siehe Projekt öffnen (Seite 111)

Verwendung

Eine globale Bibliothek kann über System.IO.FileInfo mit einem Pfad auf die Bibliotheksdatei auf einem lokalen Speichermedium oder einem Netzwerkspeicher geöffnet werden. Nur globale Anwenderbibliotheken können mittels Pfad geöffnet werden. Ein Pfad einer globalen Systembibliothek oder einer globalen Unternehmensbibliothek kann zum Öffnen der Bibliothek nicht verwendet werden.

Ab V14 SP1 können globale Bibliotheken mittels GlobalLibraryInfo geöffnet werden. Der OpenMode wird in der GlobalLibraryInfo angegeben.

Eine globale Anwenderbibliothek einer Vorgängerversion des TIA Portals kann hochgerüstet und mit der aktuellen Version des TIA Portals geöffnet werden. Eine globale Bibliothek der V13 oder einer Vorgängerversion kann nicht mittels Hochrüsten geöffnet werden. Diese Bibliotheken müssen zunächst auf V13 SP1 aktualisiert werden.

Mit TIA Portal Openness geöffnete globale Bibliotheken werden außerdem der Sammlung der globalen Bibliotheken im TIA Portal hinzugefügt und in der Benutzeroberfläche im TIA Portal angezeigt.

Programmcode: Bibliothek mittels System.IO.FileInfo öffnen

Ändern Sie folgenden Programmcode:

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

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

Programmcode: Bibliothek mittels GlobalLibraryInfo öffnen

Ändern Sie folgenden Programmcode:

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

Programmcode: Bibliothek hochrüsten

Ändern Sie folgenden Programmcode:

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

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

OpenMode

Wert	Beschreibung
ReadOnly	Schreibzugriff auf die Bibliothek.
ReadWrite	Lese- und Schreibzugriff auf die Bibliothek.

7.12.5 Offene Bibliotheken enumerieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)

Anwendung

Alle geöffneten globalen Bibliotheken im TIA Portal, unabhängig davon, ob sie über API oder die Benutzeroberfläche geöffnet wurden, können enumeriert werden.

Globale Bibliotheken aus Vorgängerversionen des TIA Portals werden nicht enumeriert, wenn sie mit Schreibzugriff geöffnet wurden.

Programmcode

Ändern Sie den folgenden Programmcode, um offene globale Bibliotheken zu enumerieren:

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

Siehe auch

Projekt öffnen (Seite 111)

7.12.6 Bibliotheken speichern und schließen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Eine Bibliothek ist geöffnet.
Siehe Bibliotheken öffnen (Seite 147)

Anwendung

Globale Anwenderbibliotheken können geschlossen oder gespeichert werden. Alle an der globalen Bibliothek vorgenommenen Änderungen werden nicht automatisch gespeichert. Alle nicht gespeicherten Änderungen werden ohne Benutzeraufforderung durch Schließen einer globalen Bibliothek verworfen.

Globale Systembibliotheken und globale Unternehmensbibliotheken können nicht geschlossen oder gespeichert werden.

Zum Speichern und Schließen einer globalen Bibliothek:

- Zum Speichern einer globalen Anwenderbibliothek verwenden Sie die Methode Save ().
- Verwenden Sie die Methode SaveAs (), wenn Sie eine globale Anwenderbibliothek in einem anderen Verzeichnis speichern möchten.
- Zum Schließen einer globalen Anwenderbibliothek verwenden Sie die Methode Close ().

Feature-Token für SaveAs () sind:

- Public API: Erforderlich für alle veröffentlichten Features
- DenyIfTransaction: Erforderlich, da SaveAs nicht innerhalb einer Transaktion zulässig sein darf, da es nicht rückgängig gemacht werden kann.

Programmcode

Ändern Sie folgenden Programmcode, um eine globale Anwenderbibliothek zu speichern:

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

Ändern Sie folgenden Programmcode, um eine globale Anwenderbibliothek an einem anderen Ort zu speichern:

```
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);  
GlobalLibraryComposition globalLibraryComposition = portal.GlobalLibraries;  
//please adapt the path and the extension alx to the installed version of TIA Portal  
FileInfo existingLibraryfileInfo = new FileInfo(@"D:\GlobalLibraries\MyGlobalLibrary  
\MyGlobalLibrary.alx");  
DirectoryInfo targetDirectoryInfo = new DirectoryInfo(@"D:\GlobalLibraries  
\GlobalLibrarySaveAs");  
UserGlobalLibrary userGlobalLibrary =  
globalLibraryComposition.Open(existingLibraryfileInfo, OpenMode.ReadWrite);  
userGlobalLibrary.SaveAs(targetDirectoryInfo);
```

Hinweis

Die Beständigkeit der Bibliothek wird nach der Operation SaveAs geändert. Von daher ist die neu gespeicherte Bibliothek nach Durchführung von SaveAs in der Bibliothekskarte verfügbar. Die ursprüngliche Bibliothek jedoch, mit der die Operation SaveAs durchgeführt wurde, ist nicht verfügbar. Die neu gespeicherte Bibliothek hat den gleichen Modus wie die ursprüngliche Bibliothek, für die SaveAs durchgeführt wurde. Das bedeutet, dass die neu gespeicherte Bibliothek im Zielpfad auch im Modus ReadOnly geöffnet wird, wenn die vorhandene Bibliothek im Modus ReadOnly geöffnet wurde. Ähnliches gilt für den Modus ReadWrite.

Ändern Sie folgenden Programmcode, um eine globale Anwenderbibliothek zu schließen:

```
UserGlobalLibrary userLib = ...
// close and discard changes
userLib.Close();
```

7.12.7 Bibliothek archivieren und abrufen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Eine Bibliothek ist geöffnet.
Siehe Bibliotheken öffnen (Seite 147)
- Eine Bibliothek ist gespeichert.
Siehe Bibliotheken speichern und schließen (Seite 149)

Anwendung

Eine geöffnete und gespeicherte Bibliothek kann archiviert werden, bevor weitere Änderungen daran vorgenommen werden. Auf diese Weise verhindern Sie unbeabsichtigte Änderungen und können die archivierte Bibliothek später wieder abrufen. Außerdem können Sie die archivierte Datei problemlos im Netzwerk teilen.

Bibliothek archivieren

Über die Schnittstelle TIA Portal Openness API können Sie eine globale Anwenderbibliothek archivieren. Die API ist am Objekt "Siemens.Engineering.UserGlobalLibrary" verfügbar.

```
public void Archive(System.IO.DirectoryInfo targetDirectory, string targetName,
Siemens.Engineering.LibraryArchivationMode archivationMode)
```

Hierbei ist 'targetName' der Name der Datei, die für die archivierten oder nicht-archivierten Bibliotheken erstellt wurde. Diese Datei kann Dateierweiterungen oder keine Dateierweiterungen enthalten. Wenn Sie keine Erweiterung oder eine andere Erweiterung als "zalx" oder "zal14" usw. angeben, kann die archivierte Datei nicht außerhalb der Openness API aus dem TIA Portal abgerufen werden.

Wenn der Wert für LibraryArchivationMode mit Compressed und DiscardRestorableDataAndCompressed angegeben wird, ist der Name der archivierten Datei identisch mit dem von Ihnen bereitgestellten Namen. Bei den Werten None und DiscardRestorableData für LibraryArchivationMode wird die Erweiterung der Bibliotheksdatei

automatisch von der Komponente Project Manager des TIA Portals anhand der aktuellen Version des TIA Portals festgelegt.

Hinweis

Vor dem Aufrufen der Archive API müssen Sie die Bibliothek gespeichert haben. Falls die Bibliothek nicht gespeicherte Änderungen enthält, löst das Archiv eine EngineeringTargetInvocationException aus.

Archivierungsmodus der Bibliothek

Für die Enumeration LibraryArchivationMode (Archivierungsmodus der Bibliothek) können vier Werte angegeben werden.

LibraryArchivation-Mode	Beschreibung
Ohne	<ul style="list-style-type: none"> Mit den Originaldateien werden keine speziellen Aktionen durchgeführt. Der Modus ähnelt dem Vorgang "Speichern unter". In diesem Modus wird keine komprimierte ZIP-Datei erstellt. Der Unterschied zu SaveAs besteht in diesem Fall darin, dass das Archiv den persistenten Speicherort nicht auf den neuen Ordner Archived umstellt, was jedoch bei SaveAs der Fall ist.
DiscardRestorable-Data	<ul style="list-style-type: none"> Bei der Dateispeicherung wird die Bibliothek in einer internen Datendatei abgelegt. Diese Datei wird anschließend bei jeder Änderung an den Bibliotheksdaten größer. Beim Modus DiscardRestorableData wird diese Datendatei reorganisiert (dabei wird nur die neueste Version der Objekte gespeichert und der Verlauf aus der Datei gelöscht). Zwischenzeitlich angefallene Daten, die Dateien des IM-Verzeichnisses und des Verzeichnisses tmp (siehe Struktur der Bibliotheksverzeichnisse) werden nicht an den Archiv-Speicherort kopiert. In diesem Modus wird keine komprimierte ZIP-Datei erstellt.
Compressed	Die vom Archivierungsprozess erstellte Struktur des TMP-Bibliotheksordners wird zu einem ZIP-kompatiblen Archiv komprimiert. Nach Erstellung der ZIP-Datei wird die TMP-Ordnerstruktur gelöscht.
DiscardRestorable-DataAndCompressed	Die vom Archivierungsprozess erstellte Struktur des TMP-Bibliotheksordners verwirft die wiederherstellbaren Daten und wird dann zu einem ZIP-kompatiblen Archiv komprimiert. Nach Erstellung der ZIP-Datei wird die TMP-Ordnerstruktur gelöscht.

Programmcode: Bibliothek archivieren

Ändern Sie folgenden Programmcode, um eine globale Anwenderbibliothek zu archivieren:

```
var tiaPortal = new TiaPortal(TiaPortalMode.WithoutUserInterface);
//Please adapt the path and the extension alx to the installed version of TIA Portal
var libraryFilePath = @"E:\Sample1\Sample1.alx";
var userGlobalLibrary = tiaPortal.GlobalLibraries.Open(new FileInfo(libraryFilePath),
OpenMode.ReadWrite);
var archivePath = @"E:\Archive";
var archiveFileName = "SampleArchive";
userGlobalLibrary.Archive(new DirectoryInfo(archivePath), archiveFileName,
LibraryArchivationMode.Compressed);
```

Bibliothek abrufen

Über die Schnittstelle TIA Portal Openness API können Sie eine archivierte TIA Portal-Bibliothek abrufen. Der Abruf ist nur mit einem komprimierten Archiv möglich. Die API ist am Objekt "Siemens.Engineering.GlobalLibraryComposition" verfügbar.

```
public Siemens.Engineering.Library.UserGlobalLibrary Retrieve(System.IO.FileInfo
sourcePath, System.IO.DirectoryInfo targetDirectory, Siemens.Engineering.OpenMode openMode)
```

Hinweis

Es ist nicht möglich, archivierte Bibliotheken mit den Enumerationswerten 'LibraryArchivationMode.None' oder 'LibraryArchivationMode.DiscardRestorableData' abzurufen.

Durch Aufrufen von RetrieveWithUpgrade API stellen Sie die archivierte Bibliothek einer früheren TIA Portal-Version wieder her. Die API-Definition lautet wie folgt:

```
public Siemens.Engineering.Library.UserGlobalLibrary
RetrieveWithUpgrade(System.IO.FileInfo sourcePath, System.IO.DirectoryInfo
targetDirectory, Siemens.Engineering.OpenMode openMode)
```

Öffnungsmodus

Die Enumeration OpenMode weist zwei Werte auf:

OpenMode	Beschreibung
ReadMode	<ul style="list-style-type: none"> Lesezugriff auf die Bibliothek. Daten können aus der Bibliothek gelesen werden.
ReadWrite	<ul style="list-style-type: none"> Schreibzugriff auf die Bibliothek. Daten können in die Bibliothek geschrieben werden.

Programmcode: Bibliothek abrufen

Ändern Sie den folgenden Programmcode, um eine Bibliothek abzurufen:

```
//Please adapt the path and the extension zalx to the installed version of TIA Portal
var archivePath = @"E:\Archive\Sample1.zalx";
var retrievedLibraryDirectory = @"E:\RetrievedLibraries";
var tiaPortal = new TiaPortal(TiaPortalMode.WithoutUserInterface);
tiaPortal.GlobalLibraries.Retrieve(new FileInfo(archivePath), new
DirectoryInfo(retrievedLibraryDirectory), OpenMode.ReadWrite));
```

Siehe auch

[Bibliotheken öffnen \(Seite 147\)](#)

[Bibliotheken speichern und schließen \(Seite 149\)](#)

7.12.8 Globale Bibliotheken erstellen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)

Anwendung

Globale Bibliotheken können über die TIA Portal Openness API erzeugt werden, indem die Methode Create auf GlobalLibraryComposition aufgerufen wird. Es wird eine globale Anwenderbibliothek ausgegeben.

GlobalLibraryComposition.Create

Ändern Sie folgenden Programmcode:

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

Entsprechend diesem Beispiel

- wird ein Ordner "D:\GlobalLibraries\Library1" erstellt
- wird eine globale Bibliotheksdatei "D:\GlobalLibraries\Library1\Library1.alXX" erstellt

Parameter zum Erstellen globaler Bibliotheken

Parameter	Datentyp	Typ	Beschreibung
Author	String	Obligatorisch	Autor einer globalen Bibliothek.
Comment	String	Optional	Kommentar einer globalen Bibliothek.
Name	String	Optional	Name einer globalen Bibliothek.
TargetDirectory	DirectoryInfo	Obligatorisch	Verzeichnis, das den Ordner mit der globalen Bibliothek enthalten wird.

Siehe auch

Projekt öffnen (Seite 111)

7.12.9 Auf Ordner in einer Bibliothek zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Sie haben mit Ihrer TIA Portal Openness-Anwendung ein Projekt geöffnet.
Siehe Projekt öffnen (Seite 111)
- Sie haben Zugriff auf die erforderliche Bibliothek.
Siehe Auf globale Bibliotheken zugreifen (Seite 143).

Verwendung

Über die Schnittstelle TIA Portal Openness API können Sie auf die Systemordner für Typen und Masterkopien in einer Bibliothek zugreifen. Dabei können Sie auf Typen, Typversionen, Masterkopien und benutzerdefinierte Ordner im Systemordner zugreifen.

Mit der Methode `Find`, z. B.

`libTypeUserFolder.Folders.Find("SomeUserFolder");`, können Sie jederzeit auf einen benutzerdefinierten Ordner zugreifen.

Programmcode: Auf Systemordner zugreifen

Um auf die Systemordner für Typen in einer Bibliothek zuzugreifen, ändern Sie folgenden Programmcode:

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

Um auf die Systemordner für Masterkopien in einer Bibliothek zuzugreifen, ändern Sie folgenden Programmcode:

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

Programmcode: Über die Methode `Find()` auf benutzerdefinierte Ordner zugreifen

Ändern Sie folgenden Programmcode:

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

Programmcode: Benutzerdefinierte Ordner enumerieren

Um benutzerdefinierte Unterordner in einem Systemordner für Typen zu enumerieren, ändern Sie folgenden Programmcode:

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

Um benutzerdefinierte Unterordner in einem Systemordner für Masterkopien zu enumerieren, ändern Sie folgenden Programmcode:

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

Um benutzerdefinierte Unterordner in einem benutzerdefinierten Ordner für Typen zu enumerieren, ändern Sie folgenden Programmcode:

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

Um benutzerdefinierte Unterordner in einem benutzerdefinierten Ordner für Masterkopien zu enumerieren, ändern Sie folgenden Programmcode:

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

Programmcode: Benutzerdefinierte Ordner erstellen

Um einen benutzerdefinierten Ordner für Typen zu erstellen, ändern Sie folgenden Programmcode:

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

Um einen benutzerdefinierten Ordner für Masterkopien zu erstellen, ändern Sie folgenden Programmcode:

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

Programmcode: Benutzerdefinierte Ordner umbenennen

Um einen benutzerdefinierten Ordner für Typen zu erstellen, ändern Sie folgenden Programmcode:

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

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

Um einen benutzerdefinierten Ordner für Masterkopien zu erstellen, ändern Sie folgenden Programmcode:

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

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

Siehe auch

[Auf Kopiervorlagen zugreifen \(Seite 169\)](#)

7.12.10 Auf Typen zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Sie haben mit Ihrer TIA Portal Openness-Anwendung ein Projekt geöffnet.
Siehe Projekt öffnen (Seite 111)
- Sie haben Zugriff auf die erforderliche Bibliothek.
Siehe Auf globale Bibliotheken zugreifen (Seite 143).
- Sie haben Zugriff auf eine Gruppe für Typen.
Siehe Auf Ordner in einer Bibliothek zugreifen (Seite 155).

Verwendung

Auf die Typen in einer Bibliothek können Sie über die Schnittstelle TIA Portal Openness API zugreifen.

- Sie können die Typen enumerieren.
- Sie können Typen umbenennen.
- Sie können auf die folgenden Attribute der einzelnen Typen zugreifen:

Attribut	Datentyp	Beschreibung
Author	String	Gibt den Namen des Autors zurück.
Comment	MultilingualText	Gibt den Kommentar zurück.
Guid	Guid	Gibt die GUID des Typs zurück. ¹
Name	String	Gibt den Namen des Typs zurück. ²

¹ Mit diesem Attribut können Sie einen einzelnen Typ in einer Bibliothek finden. Die Suche ist rekursiv.

² Über dieses Attribut können Sie einen einzelnen Typ in einem Ordner finden. Unterordner werden nicht in die Suche einbezogen. Die Typennamen sind nicht eindeutig. Es kann mehrere Typen mit demselben Namen in verschiedenen Gruppen geben. Die GUID des Typs ist hingegen eindeutig.

Unterklassen für Bibliothekstypobjekte

Mit der TIA Portal Openness API können Sie über Unterklassen auf Bibliothekstypobjekte zugreifen. Folgende Unterklassen sind vorhanden:

- Siemens.Engineering.Hmi.Faceplate.FaceplateLibraryType
- Siemens.Engineering.Hmi.Faceplate.FaceplateLibraryTypeVersion
- Siemens.Engineering.Hmi.RuntimeScripting.VBScriptLibraryType
- Siemens.Engineering.Hmi.RuntimeScripting.VBScriptLibraryTypeVersion
- Siemens.Engineering.Hmi.RuntimeScripting.CScriptLibraryType
- Siemens.Engineering.Hmi.RuntimeScripting.CScriptLibraryTypeVersion
- Siemens.Engineering.Hmi.Screen.ScreenLibraryType

- Siemens.Engineering.Hmi.Screen.ScreenLibraryTypeVersion
- Siemens.Engineering.Hmi.Screen.StyleLibraryType
- Siemens.Engineering.Hmi.Screen.StyleLibraryTypeVersion
- Siemens.Engineering.Hmi.Screen.StyleSheetLibraryTypeVersion
- Siemens.Engineering.Hmi.Tag.HmiUdtLibraryType
- Siemens.Engineering.Hmi.Tag.HmiUdtLibraryTypeVersion
- Siemens.Engineering.SW.Blocks.CodeBlockLibraryType
- Siemens.Engineering.SW.Blocks.CodeBlockLibraryTypeVersion
- Siemens.Engineering.SW.Types.PlcTypeLibraryType
- Siemens.Engineering.SW.Types.PlcTypeLibraryTypeVersion

Der folgende Code ist ein Beispiel für die Verwendung von Unterklassen für Bibliothekstypen.

```
ProjectLibrary library = project.ProjectLibrary;
VBScriptLibraryType vbScriptType = ...;
VBScriptLibraryType libraryTypeAsVbScript = libraryType as VBScriptLibraryType;
VBScriptLibraryTypeVersion libraryTypeVersionAsVbScript = libraryTypeVersion as
VBScriptLibraryTypeVersion
```

Programmcode

Um alle Typen im Systemordner einer Bibliothek zu enumerieren, ändern Sie folgenden Programmcode:

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

Um alle Typen in einem benutzerdefinierten Ordner einer Bibliothek zu enumerieren, ändern Sie folgenden Programmcode:

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

7.12 Funktionen in Bibliotheken

Um auf die Attribute eines Typs zuzugreifen, ändern Sie folgenden Programmcode:

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

Um einen einzelnen Typ anhand seines Namens oder der GUID zu finden, ändern Sie folgenden Programmcode:

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

Um einen Typ umzubenennen, ändern Sie folgenden Programmcode:

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

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

7.12.11 Auf Typ-Versionen zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Sie haben mit Ihrer TIA Portal Openness-Anwendung ein Projekt geöffnet.
Siehe Projekt öffnen (Seite 111)
- Sie haben Zugriff auf die erforderliche Bibliothek.
Siehe Auf globale Bibliotheken zugreifen (Seite 143).
- Sie haben Zugriff auf eine Gruppe für Typen.
Siehe Auf Ordner in einer Bibliothek zugreifen (Seite 155).

Verwendung

Sie haben über die Schnittstelle TIA Portal Openness API Zugriff auf Typversionen.

- Sie können die Typversionen eines Typs enumerieren.
- Sie können den Typ ermitteln, zu dem eine Typversion gehört.
- Sie können die Instanzen einer Typversion enumerieren.
- Sie können eine neue Instanz einer Typversion erstellen.
- Sie können von einer Instanz zu dem damit verbundenen Versionsobjekt navigieren.
- Sie können auf die folgenden Attribute der einzelnen Typversionen zugreifen:

Attribut	Datentyp	Beschreibung
Author	String	Gibt den Namen des Autors zurück.
Comment	MultilingualText	Gibt den Kommentar zurück.
Guid	Guid	Gibt die GUID der Typversion zurück. ¹
ModifiedDate	DateTime	Gibt Datum und Uhrzeit zurück, zu denen die Typversion auf den Status "Committed" gesetzt wurde.
State	LibraryTypeVersionState	Gibt den Status der Version zurück: <ul style="list-style-type: none"> • InWork: Entspricht dem Status „In Bearbeitung“ oder „Im Test“, abhängig von dem zugeordneten Typ. • Committed: Entspricht dem Status „Freigegeben“.
TypeObject	LibraryType	Gibt den Typ zurück, zu dem diese Typversion gehört.
VersionNumber	Version	Gibt die Versionsnummer als dreistellige Versionskennung zurück, zum Beispiel „1.0.0“. ²

¹ Über dieses Attribut können Sie eine einzelne Typversion in einer Bibliothek finden.

² Über dieses Attribut können Sie eine einzelne Typversion in einer Zusammensetzung "LibraryTypeVersion" finden.

Alle Typversionen eines Typs enumerieren

Ändern Sie folgenden Programmcode:

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

Attribute einer Typversion aufrufen

Ändern Sie folgenden Programmcode:

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

Instanz einer Typversion erstellen

Sie können eine neue Instanz einer Typversion erstellen. Die folgenden Objekte werden unterstützt:

- Bausteine (FB/FC)
- PLC-Anwenderdatentypen
- Bilder
- VB-Skripte

Eine Instanz einer Typversion kann aus der globalen Bibliothek oder der Projektbibliothek erzeugt werden. Wenn Sie eine Instanz einer Typversion aus einer globalen Bibliothek erstellen, wird die Typversion zunächst mit der Projektbibliothek synchronisiert.

Eine wiederherstellbare Ausnahme wird ausgelöst, wenn eine Instanz nicht im Ziel erstellt werden kann. Mögliche Gründe sind:

- Die Bibliothekstypversion ist in Arbeit
- Eine Instanz der Bibliothekstypversion ist bereits im Zielgerät vorhanden

Ändern Sie folgenden Programmcode:

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

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

Ändern Sie folgenden Programmcode:

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

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

Ändern Sie folgenden Programmcode:

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

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

Ändern Sie folgenden Programmcode:

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

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

Verwendungen einer Typversion ermitteln

Die folgenden Verwendungen werden bei Typversionen unterschieden:

- Die Typversion verwendet andere Typversionen aus der Bibliothek.
Beispiel: Ein Anwenderdatentyp wird in einem Programmstein verwendet. Der Programmstein muss Zugriff auf den Anwenderdatentyp haben. Das bedeutet, dass der Programmstein vom Anwenderdatentyp abhängig ist.
Wenn Sie mit der Methode `GetDependencies()` auf das Abhängigkeitsattribut von `CodeBlockLibraryVersion` zugreifen, wird eine Liste von `LibraryTypeVersions` zurückgegeben.
- Der Typ wird von einer anderen Typversion in der Bibliothek verwendet.
Beispiel: Ein Anwenderdatentyp wird in einem Programmstein verwendet. Der Programmstein muss Zugriff auf den Anwenderdatentyp haben. Der Anwenderdatentyp hat den zugehörigen Programmstein. Der Programmstein ist vom Anwenderdatentyp abhängig.
Wenn Sie mit der Methode `GetDependents()` auf das Abhängigkeitsattribut von `PlcTypeLibraryTypeVersion` zugreifen, wird eine Liste von `LibraryTypeVersions` zurückgegeben.

Bei beiden Attributen wird eine Liste zurückgegeben, die Objekte des Typs `LibraryTypeVersion` enthält. Sind keine Verwendungen vorhanden, wird eine leere Liste zurückgegeben.

Hinweis

Wenn Sie diese Attribute bei Typversionen mit dem Status "InWork" verwenden, kann eine Ausnahme ausgelöst werden.

Ändern Sie folgenden Programmcode:

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

Programmcode

Um den Typ zu ermitteln, zu dem eine Typversion gehört, ändern Sie folgenden Programmcode:

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

Um die Masterkopien zu ermitteln, die Instanzen einer Typversion enthalten, ändern Sie folgenden Programmcode:

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

Um eine einzelne Typversion anhand ihrer Versionsnummer zu finden, ändern Sie folgenden Programmcode:

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

7.12.12 Zugriff auf Bausteine in Bibliotheken

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Sie können mit TIA Portal Openness den Versionsstand aus dem Bibliotheksobjekt abrufen, ohne das Objekt zu instanziieren und zu übersetzen.

Die folgende neue Export-Aktion wird im Engineering-Objekt LibraryTypeVersion zum Exportieren des Versionsinhalts bereitgestellt.

Die Aktion erstellt die exportierte XML-Datei auf dem mit exportFileInfo angegebenen Versionsstand.

```
void Export(''FileInfo'' exportFileInfo, ''ExportOptions'' exportOptions)
```

Es wird eine Ausnahme zurückgegeben:

- Wenn die Benutzerausnahme lautet "Die Versionsdaten können nicht exportiert werden, weil sie sich im Arbeitszustand befinden" und die zu exportierende Version den Zustand "Im Test" aufweist.
- Wenn eine allgemeine DataExchange-Benutzerausnahme, z B. "FileAlreadyExists" vorliegt, und die Meldung ausgegeben wird: "Der Export kann nicht durchgeführt werden, weil die Datei 'D:**.xml' bereits vorhanden ist.".

Engineering-Objekt Bibliothekstypversion

Die folgenden Ergänzungen werden am Objekt Library TypeVersion engineering vorgenommen:

1. Die Export-Aktion wird an LibraryTypeVersion bereitgestellt.
2. Ein Navigator zum Navigieren zum durchsuchbaren Versionsinhalt.
 - Navigatorname: ContentObject
 - ReadPublicationLevel : System
 - Zusammenhangsname: Engineering.Library.DefaultVersionContentObject
 - Der Basisnavigator bietet eine Standard-Implementierung, um eine leere durchsuchbare Sammlung als Versionsinhalt bereitzustellen. Der Kunde kann diesen Navigator überschreiben und den Versionsinhalt bereitstellen.
3. "LibraryTypeName" und "LibraryTypeGuid" am Objekt LibraryTypeVersion.

Attribute	ReadPublicationLevel
LibraryTypeName	System
LibraryTypeGuid	System

Hinweis

Dies ist erforderlich, weil ohne diese Information nicht klar wäre, zu welchem Typ die exportierte Version (laut Versionsstand in der exportierten XML-Datei) gehört.

Exportierter Inhalt

Der folgende Inhalt wird in der exportierten Datei angezeigt:

- Exportierter Versionsstand der Bibliothek

Attribute	SimaticMLAccess
Author	ReadWrite
Guid	ReadOnly
Modified Data	ReadOnly
VersionNumber	ReadWrite
LibraryTypeName	ReadOnly
Library TypeGuid	ReadOnly

Exportierte Navigatoren:

- Comment

Hinweis

Attribute mit schreibgeschütztem SimaticMLAccess werden nur exportiert, wenn Sie die Exportoption "ExportOptions.WithReadOnly" wählen.

Programmcode

Ändern Sie folgenden Programmcode, um den Versionsstand des exportierten Inhalts zu exportieren:

```
Guid g = new Guid("35ad9996-d6d7-40c9-989f-0f0c7e21a7b2");
//Block1
var version = m_Project.ProjectLibrary.FindType(g).Versions[0];
version.Export(new FileInfo(@"D:\ExportCodeBlock.xml"), ExportOptions.WithReadOnly);
```

Siehe auch

Verbindung zum TIA Portal aufbauen (Seite 79)

Projekt öffnen (Seite 111)

7.12.13 Auf Instanzen zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Sie haben mit Ihrer TIA Portal Openness-Anwendung ein Projekt geöffnet.
Siehe Projekt öffnen (Seite 111)
- Sie haben Zugriff auf die erforderliche Bibliothek.
Siehe Auf globale Bibliotheken zugreifen (Seite 143).
- Sie haben Zugriff auf eine Gruppe für Typen.
Siehe Auf Ordner in einer Bibliothek zugreifen (Seite 155).

Verwendung

Sie haben über die Schnittstelle TIA Portal Openness API Zugriff auf Instanzen von Typversionen.

Mit der Methode `FindInstances(IInstanceSearchScope searchScope)` können Sie alle Instanzen einer Typversion ermitteln.

Mit dem Parameter `searchScope` können Sie den Bereich des Projekts angeben, der durchsucht werden soll. Die folgenden Klassen implementieren die Schnittstelle `IInstanceSearchScope` und können zum Suchen nach Instanzen verwendet werden:

- PlcSoftware
- HmiTarget

Die Methode gibt eine Liste mit Objekten des Typs `LibraryTypeInstanceStateInfo` zurück. Sind keine Instanzen vorhanden, wird eine leere Liste zurückgegeben.

Hinweis

Instanzen von Bildbausteinen und HMI-Benutzerdatentypen sind stets mit der zugehörigen Typversion verknüpft.

Instanzen aller anderen Objekte wie Programmabusteine oder Bilder können mit einer Typversion verknüpft sein.

Instanzen einer Typversion enumerieren

Ändern Sie folgenden Programmcode:

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

IInstanceSearchScope searchScope = plcSoftware as IInstanceSearchScope;

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

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

Von einer Instanz zu dem damit verbundenen Versionsobjekt navigieren

Verwenden Sie das Attribut LibraryTypeVersion des Diensts LibraryTypeInstanceStateInfo, um von einer Instanz zu dem damit verbundenen Versionsobjekt zu navigieren.

Die folgenden Objekte bieten den Dienst LibraryTypeInstanceStateInfo:

- Bausteine FB
- Bausteine FC
- PLC-Anwenderdatentypen
- Bilder
- VB-Skripte

Wenn ein Instanzobjekt nicht mit einem Versionsobjekt verbunden ist, dann stellt es den Dienst "LibraryTypeInstanceStateInfo" nicht bereit.

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

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

7.12.14 Auf Kopiervorlagen zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Sie haben Zugriff auf die erforderliche Bibliothek.
Siehe Auf globale Bibliotheken zugreifen (Seite 143)
- Sie haben Zugriff auf eine Gruppe für Masterkopien.
Siehe Auf Ordner in einer Bibliothek zugreifen (Seite 155)

Verwendung

Die Schnittstelle TIA Portal Openness API unterstützt den Zugriff auf Masterkopien in einer globalen Bibliothek und der Projektbibliothek:

- Masterkopien erstellen
- Masterkopien in Systemordnern und benutzerdefinierten Ordnern enumerieren
- Masterkopien umbenennen
- Informationen von Masterkopien abfragen
- Informationen von Objekten in einer Masterkopie abfragen

Attribut	Datentyp	Beschreibung
Author	String	Gibt den Namen des Autors zurück.
ContentDescriptions	MasterCopyContentDescriptionComposition	Gibt eine Beschreibung des Inhalts der Masterkopie zurück.
CreationDate	DateTime	Gibt das Erstellungsdatum zurück.
Name	String	Gibt den Namen der Masterkopie zurück.

Programmcode

Um alle Masterkopien im Systemordner einer Bibliothek zu enumerieren, ändern Sie den folgenden Programmcode:

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

7.12 Funktionen in Bibliotheken

Ändern Sie den folgenden Programmcode, um mit der Methode "Find" auf eine einzelne Masterkopie zuzugreifen:

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

```

Ändern Sie den folgenden Programmcode, um Gruppen und Untergruppen von Masterkopien zu enumerieren:

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

Ändern Sie den folgenden Programmcode, um mit der Methode "Find" auf einen MasterCopyUserFolder zuzugreifen:

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

```

Um eine Masterkopie umzubenennen, ändern Sie folgenden Programmcode:

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

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

Informationen von Masterkopien abfragen

Um Informationen einer Masterkopie abzurufen, ändern Sie den folgenden Programmcode:

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

Informationen von Objekten in einer Masterkopie abfragen

Das Objekt MasterCopy enthält den Navigator ContentDescriptions, bei dem es sich um eine Zusammensetzung von MasterCopyContentDescriptions handelt.

Eine Masterkopie kann mehrere Objekte enthalten. Das Objekt MasterCopy enthält ContentDescriptions für jedes direkt in der Masterkopie enthaltene Objekt. Wenn die Masterkopie einen Ordner enthält, der ebenfalls einige Elemente enthält, enthält das Objekt MasterCopy nur eine ContentDescription des Ordners.

```
MasterCopy multiObjectMasterCopy = ...;

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

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

7.12.15 Masterkopie aus einem Projekt in Bibliothek erzeugen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Wenn eine Bibliothek eine Lesen-Schreiben-Bibliothek ist, können Sie eine Masterkopie aus einer **IMasterCopySource**-Quelle am Zielort erzeugen.

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

Eine **EngineeringException**-Ausnahme wird in folgenden Fällen ausgelöst:

- Am Zielort besteht nur Lesezugriff
- Die Erzeugung der Masterkopie aus der Quelle wird vom System abgelehnt

Die folgenden Elemente sind als **IMasterCopySources** definiert:

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

Programmcode

Verwenden Sie folgenden Programmcode:

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

7.12.16 Ein Objekt aus einer Masterkopie erstellen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Die PLC ist nicht online.

Verwendung

Die Schnittstelle TIA Portal Openness API unterstützt die Verwendung von Masterkopien im Projekt. Sie können in der Zusammensetzung des Objekts aus einer Masterkopie in einer Projektbibliothek oder einer globalen Bibliothek mit Hilfe der Methode CreateFrom ein Objekt erstellen.

Der Ausgabetyp entspricht dem Ausgabetyp der jeweiligen Zusammensetzung.

Die Methode CreateFrom unterstützt nur Masterkopien, die einzelne Objekte enthalten. Wenn die Zusammensetzung, wo die Aktion aufgerufen wird, und die Quellmasterkopie inkompatibel sind (z. B. wenn die Quellmasterkopie eine PLC-Variablentabelle enthält und die Zusammensetzung die Zusammensetzung eines PLC-Bausteins ist), wird eine wiederherstellbare Ausnahme ausgelöst.

Die folgenden Zusammensetzungen werden unterstützt:

- Siemens.Engineering.HW.DeviceComposition
- Siemens.Engineering.HW.DeviceItemComposition
- Siemens.Engineering.SW.Blocks.PlcBlockComposition
- Siemens.Engineering.SW.Tags.PlcTagTableComposition
- Siemens.Engineering.SW.Tags.PlcTagComposition
- Siemens.Engineering.SW.Types.PlcTypeComposition
- Siemens.Engineering.SW.TechnologicalObjects.TechnologicalInstanceDBComposition

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

Programmcode: PLC-Baustein aus einer Masterkopie erstellen

Um einen PLC-Baustein aus einer Masterkopie in einer Bibliothek zu erstellen, ändern Sie den folgenden Programmcode:

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

Programmcode: Ein Gerät aus einer Masterkopie erstellen

Um ein Gerät aus einer Masterkopie in einer Bibliothek zu erstellen, ändern Sie den folgenden Programmcode:

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

Programmcode: Ein Geräteelement aus einer Masterkopie erstellen

Um ein Geräteelement aus einer Masterkopie in einer Bibliothek zu erstellen, ändern Sie den folgenden Programmcode:

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

Programmcode: Ein Subnetz aus einer Masterkopie erstellen

Um ein Subnetz aus einer Masterkopie in einer Bibliothek zu erstellen, ändern Sie den folgenden Programmcode:

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

Programmcode: Einen Geräteordner aus einer Masterkopie erstellen

Um einen Geräteordner aus einer Masterkopie in einer Bibliothek zu erstellen, ändern Sie den folgenden Programmcode:

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

Siehe auch

Auf Kopiervorlagen zugreifen (Seite 169)

7.12.17 Masterkopien kopieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Die Schnittstelle TIA Portal Openness API unterstützt das Kopieren von Masterkopien innerhalb einer Bibliothek und von einer Bibliothek zu einer anderen mithilfe der Aktion `CreateFrom`. Die Aktion erstellt ein neues Objekt basierend auf der Quellmasterkopie und platziert es in der Zusammensetzung, wo die Aktion aufgerufen wurde. Die Aktion versucht, die neue Masterkopie mit dem gleichen Namen zu erstellen wie die Quellmasterkopie. Wenn dieser Name nicht verfügbar ist, gibt das System der neuen Masterkopie einen neuen Namen. Danach gibt es die neue Masterkopie aus.

Wenn sich die Zusammensetzung, in der die Aktion "CreateFrom" aufgerufen wird, in einer schreibgeschützten globalen Bibliothek befindet, wird eine wiederherstellbare Ausnahme ausgelöst.

Programmcode

Ändern Sie folgenden Programmcode:

```
ProjectLibrary projectLibrary = ...;

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

Siehe auch

Auf Kopiervorlagen zugreifen (Seite 169)

7.12.18 Ermitteln veralteter Typinstanzen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Sie haben Zugriff auf die erforderliche Bibliothek.
Siehe Auf globale Bibliotheken zugreifen (Seite 143)
- Sie haben Zugriff auf einen Ordner für Typen.
Siehe Auf Ordner in einer Bibliothek zugreifen (Seite 155).

Verwendung

Über die Schnittstelle TIA Portal OpennessAPI können Sie Typversionen ermitteln, die zu den Instanzen im geöffneten Projekt gehören. Dabei liefert die TIA Portal OpennessAPI pro Instanz einen der beiden folgenden Zustände:

- Die Instanz bezieht sich auf eine veraltete Typversion.
- Die Instanz bezieht sich auf die aktuelle Typversion.

Für die Versionsermittlung gelten die folgenden Regeln:

- Basis für die Versionsermittlung sind eine Bibliothek und das Projekt, das Sie über die Schnittstelle TIA Portal OpennessAPI öffnen möchten.
- Im Rahmen der Versionsermittlung werden keine Instanzen aktualisiert.

Signatur

Über die Methode UpdateCheck können Sie die Instanzen einer Typversion ermitteln:
`UpdateCheck(Project project, UpdateCheckMode updateCheckMode)`

Parameter	Funktion
Project	Gibt das Projekt an, in dem die Typversionen von Instanzen ermittelt werden.
UpdateCheckMode	Gibt die Versionen an, die ermittelt werden: <ul style="list-style-type: none"> • ReportOutOfDateOnly: Gibt nur den Status vom Typ „veraltet“ zurück. • ReportOutOfDateAndUpToDate: Gibt den Status der Typen „veraltet“ und „aktuell“ zurück.

Ergebnis

Bei der Versionsermittlung werden die Geräte des Projekts von oben nach unten abgefragt. Jedes Gerät wird darauf geprüft, ob seine Konfigurationsdaten eine Instanz einer Typversion aus der angegebenen Bibliothek enthalten. Die Methode `UpdateCheck` gibt das Ergebnis der Versionsprüfung in hierarchischer Reihenfolge zurück.

Die nachstehende Tabelle zeigt das Ergebnis einer Versionsprüfung mit dem Parameter `UpdateCheck.ReportOutOfDateAndUpToDate`:

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

Programmcode

Es besteht kein Unterschied zwischen Projektbibliotheken und globalen Bibliotheken hinsichtlich der Handhabung der Aktualisierungsprüfung.

7.12 Funktionen in Bibliotheken

Um die Typversionen einer globalen Bibliothek oder Projektbibliothek für Instanzen im Projekt zu ermitteln, ändern Sie den folgenden Programmcode:

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

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

    //Show result
    RecursivelyWriteMessages(result.Messages);

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

Um das Ergebnis der Versionsermittlung auszugeben und die Meldungen einzeln zu durchlaufen, ändern Sie folgenden Programmcode:

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

Um im Ergebnis der Versionsermittlung auf einzelne Meldungsteile zuzugreifen, ändern Sie folgenden Programmcode:

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

7.12.19 Projekt aktualisieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Sie haben mit Ihrer TIA Portal Openness-Anwendung ein Projekt geöffnet.
Siehe Projekt öffnen (Seite 111)
- Sie haben Zugriff auf die erforderliche Bibliothek.
Siehe Auf globale Bibliotheken zugreifen (Seite 143).
- Sie haben Zugriff auf einen Ordner für Typen.
Siehe Auf Ordner in einer Bibliothek zugreifen (Seite 155).

Verwendung

Über die Schnittstelle TIA Portal Openness API können Sie Instanzen eines ausgewählten Typs in einem Typordner eines Projekts aktualisieren.

Bei der Aktualisierung werden die im Projekt verwendeten Instanzen basierend auf der letzten freigegebenen Typversion aktualisiert. Wenn Sie die Instanzen in einer globalen Bibliothek aktualisieren, wird zunächst eine Synchronisierung durchgeführt.

Signatur

Zum Aktualisieren von Instanzen verwenden Sie die Methode `UpdateProject`.

Verwenden Sie für Klassen, die die Schnittstelle `LibraryTypes` implementieren, den folgenden Aufruf:

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

Verwenden Sie für Klassen, die die Schnittstelle `ILibrary` implementieren, den folgenden Aufruf:

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

Jeder Aufruf wird in die Protokolldatei im Projektverzeichnis eingetragen.

Parameter	Funktion
<code>IEnumerable<ILibraryTypeOrFolderSelection> selectedTypesOrFolders</code>	Gibt den Ordner oder zu synchronisierende Typen oder deren Instanzen im Projekt an, die aktualisiert werden sollen.
<code>IUpdateProjectScope updateProjectScope</code> <code>IEnumerable <IUpdateProjectScope> updateProjectScope</code>	Gibt die Objekte im Projekt an, in denen die Verwendungen der Instanzen aktualisiert werden sollen. Die folgenden Objekte werden unterstützt: <ul style="list-style-type: none"> • PlcSoftware • HmiTarget

Programmcode

Um Instanzen von ausgewählten Typen innerhalb eines Typordners zu aktualisieren, ändern Sie folgenden Programmcode:

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

7.12.20 Bibliothek aktualisieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Sie haben mit Ihrer TIA Portal Openness-Anwendung ein Projekt geöffnet.
Siehe Projekt öffnen (Seite 111)
- Sie haben Zugriff auf die erforderliche Bibliothek.
Siehe Auf globale Bibliotheken zugreifen (Seite 143).
- Sie haben Zugriff auf einen Ordner für Typen.
Siehe Auf Ordner in einer Bibliothek zugreifen (Seite 155).

Verwendung

Die Schnittstelle TIA Portal Openness API unterstützt die folgenden Aktualisierungen in der Projektbibliothek:

- Synchronisieren ausgewählter Typen zwischen Bibliotheken.

Bei der Synchronisierung wird die Ordnerstruktur nicht angepasst. Die zu aktualisierenden Typen werden über ihre GUID ermittelt und aktualisiert:

- Wenn ein Typ in einer Bibliothek eine Typversion enthält, die in der zu aktualisierenden Bibliothek fehlt, wird die Typversion kopiert.
- Der Vorgang wird abgebrochen und eine Exception ausgelöst, wenn ein Typ in einer Bibliothek eine Typversion mit unterschiedlicher GUID enthält.

Signatur

Zum Synchronisieren von Typversionen verwenden Sie die Methode `UpdateLibrary`.

Verwenden Sie für Klassen, die die Schnittstelle `LibraryTypes` implementieren, den folgenden Aufruf:

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

Verwenden Sie für Klassen, die die Schnittstelle `ILibrary` implementieren, den folgenden Aufruf:

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

Parameter	Funktion
<code>IEnumerable<ILibraryTypeOrFolderSelection> selectedTypesOrFolders</code>	Gibt den Ordner oder zu synchronisierende Typen oder deren Instanzen im Projekt an, die aktualisiert werden sollen.
<code>ILibrary targetLibrary</code>	Gibt die Bibliothek an, deren Inhalt mit einer Bibliothek synchronisiert wird. Wenn Quellbibliothek und Zielbibliothek identisch sind, wird eine Ausnahme ausgelöst.

Programmcode

Um einen Typ aus einer Projektbibliothek mit einer globalen Bibliothek zu synchronisieren, ändern Sie den folgenden Programmcode:

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

Um ausgewählte Typen in einem Typordner zwischen einer globalen Bibliothek und der Projektbibliothek zu synchronisieren, ändern Sie folgenden Programmcode:

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

7.12.21 Bibliotheksinhalte löschen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Sie haben mit Ihrer TIA Portal Openness-Anwendung ein Projekt geöffnet.
Siehe Projekt öffnen (Seite 111)
- Sie haben Zugriff auf die erforderliche Bibliothek.
Siehe Auf globale Bibliotheken zugreifen (Seite 143).
- Sie haben Zugriff auf einen Ordner für Typen.
Siehe Auf Ordner in einer Bibliothek zugreifen (Seite 155).

Verwendung

Mit der Schnittstelle TIA Portal Openness API können Sie die folgenden Inhalte in der Projektbibliothek löschen:

- Typen
- Typversion
- Benutzerdefinierte Ordner für Typen
- Masterkopien
- Benutzerdefinierte Ordner für Masterkopien

Hinweis

Löschen von Typen und Ordnern mit benutzerdefinierten Typen

Wenn Sie einen Typ oder einen Ordner mit benutzerdefinierten Typen löschen möchten, müssen die "Regeln zum Löschen von Versionen" erfüllt werden. Einen leeren Typordner können Sie jederzeit löschen.

Hinweis

Regeln zum Löschen von Versionen

Sie können nur Versionen mit dem Status „Committed“ löschen. Die folgenden Regeln gelten ebenfalls beim Löschen von Versionen:

- Wenn eine neue Version mit dem Status "InWork" gerade aus einer Version mit dem Status "Committed" erstellt wurde, können Sie die Version mit dem Status "Committed" nur dann löschen, wenn die neue Version verworfen wird oder den Status "Committed" erhält.
 - Wenn ein Typ lediglich eine Version hat, wird auch der Typ gelöscht.
 - Wenn Version A von Version B eines anderen Typs abhängig ist, löschen Sie zunächst Version A und danach Version B.
 - Wenn es Instanzen von Version A gibt, können Sie Version A nur dann löschen, wenn auch die Instanzen gelöscht werden. Wenn eine Instanz auch in einer Masterkopie enthalten ist, wird die Masterkopie ebenfalls gelöscht.
-

Programmcode

Ändern Sie den folgenden Programmcode, um Typen oder Ordner mit benutzerdefinierten Typen zu löschen:

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

Um einen einzelnen Typ oder einen einzelnen Ordner mit benutzerdefinierten Typen zu löschen, ändern Sie folgenden Programmcode:

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

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

7.13 Funktionen für das Aufrufen von Geräten, Netzwerken und Verbindungen

Um eine Version zu löschen, ändern Sie folgenden Programmcode:

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

Um eine Masterkopie oder einen Ordner mit benutzerdefinierten Masterkopien zu löschen, ändern Sie den folgenden Programmcode:

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

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

Siehe auch

Auf Kopiervorlagen zugreifen (Seite 169)

7.13 Funktionen für das Aufrufen von Geräten, Netzwerken und Verbindungen

7.13.1 Editor "Geräte & Netze" öffnen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Sie können über die API-Schnittstelle den Editor „Geräte & Netze“ über eine der beiden folgenden Methoden öffnen:

- `ShowHwEditor(View.Topology oder View.Network oder View.Device)`: Öffnet den Editor „Geräte & Netze“ aus dem Projekt.
- `ShowInEditor(View.Topology oder View.Network oder View.Device)`: Zeigt das angegebene Gerät im Editor „Geräte & Netze“ an.

Über den Parameter `View` definieren Sie die Sicht, die beim Öffnen des Editors angezeigt wird:

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

Programmcode

Um den Editor „Geräte & Netze“ zu öffnen, ändern Sie den folgenden Programmcode:

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

Um den Editor „Geräte & Netze“ für ein Gerät zu öffnen, ändern Sie folgenden Programmcode:

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

Siehe auch

[Import von Projektierungsdaten \(Seite 780\)](#)

7.13.2 PLC-Target und HMI-Target abfragen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Sie können ermitteln, ob eine Softwarebasis in der TIA Portal Openness API als PLC-Target (PlcSoftware) oder HMI-Target verwendet werden kann.

Programmcode: PLC-Target

Um festzustellen, ob ein Geräteelement als PLC-Target verwendet werden kann, verwenden Sie folgenden Programmcode:

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

Programmcode: HMI-Target

Um festzustellen, ob ein Geräteelement als HMI-Target verwendet werden kann, ändern Sie folgenden Programmcode:

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

Siehe auch

[Geräte enumerieren \(Seite 236\)](#)

7.13.3 Auf Attribute eines Adressobjekts zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Für Schreibzugriff ist der PLC offline.

Verwendung

Mit der Schnittstelle TIA Portal Openness API können Sie Attribute des Adressobjekts abrufen oder festlegen.

Ferner können Sie das aktuelle Prozessabbild einem OB zuweisen.

Auf die folgenden Attribute kann zugegriffen werden:

Attributname	Datentyp	Schreibbar	Zugriff	Beschreibung
IsochronousMode	BOOL	r/w	Dynamisches Attribut	Isochronen Modus aktivieren/deaktivieren
ProcessImage	Int32	r/w	Dynamisches Attribut	Partitionsnummer des Prozessabbilds festlegen/abrufen
InterruptObNumber	Int64	r/w	Dynamisches Attribut	Nummer des Alarm-Organisationsbausteins festlegen/abrufen (nur klassischer Controller)
StartAddress	Int32	r/w	Modelliertes Attribut	Neuen Wert für StartAddress festlegen/abrufen

Einschränkungen

- Attribut StartAddress
 - Durch Festlegen von StartAddress kann implizit die StartAddress des gegenteiligen IO-Typs am Namensmodul geändert werden. Durch Ändern der Eingangsadresse wird die Ausgangsadresse geändert.
 - Der Schreibzugriff wird nicht für alle Geräte unterstützt.
 - Gepackte Adressen werden in TIA Portal Openess nicht unterstützt.
 - Durch Ändern der Adresse über TIA Portal Openess werden die zugewiesenen Variablen nicht neu verdrahtet.
- Attribut InterruptObNumber
 - Nur zugänglich in Einstellungen mit S7-300 oder S7-400 Controllern. Bei S7-400 Controllern wird der Schreibzugriff unterstützt.

Programmcode: Attribute eines Adressobjekts abrufen oder festlegen

Um auf den isochronen Modus eines Adressobjekts zuzugreifen, ändern Sie den folgenden Programmcode:

```
Address address= ...;

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

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

Um auf das Attribut ProcessImage eines Adressobjekts zuzugreifen, ändern Sie den folgenden Programmcode:

```
Address address= ...;

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

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

7.13 Funktionen für das Aufrufen von Geräten, Netzwerken und Verbindungen

Um auf das Attribut `InterruptObNumber` eines Adressobjekts zuzugreifen, ändern Sie den folgenden Programmcode:

```
Address address= ...;

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

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

//default value = 40
```

Um auf das Attribut `StartAddress` eines Adressobjekts zuzugreifen, ändern Sie den folgenden Programmcode:

```
Address address= ...;

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

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

Programmcode: Das aktuelle Prozessabbild einem OB zuweisen

Um das aktuelle Prozessabbild einem OB zuweisen, ändern Sie den folgenden Programmcode:

```
OB obX =...
Address address= ...;

// assign PIP 5 to obX

address.SetAttribute("ProcessImage", 5);

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

// remove this PIP-OB assignment

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

7.13.4 Aufrufen eines Modulkanals

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Signalmodule wie analoge Eingangsmodule haben normalerweise mehrere Kanäle in einem einzelnen Modul. Normalerweise bieten Kanäle mehrere Funktionen gleichzeitig, z. B. kann ein analoges Eingangsmodul mit vier Kanälen vier Spannungswerte zur selben Zeit messen.

Zum Aufrufen aller Kanäle eines Moduls wird das Kanalattribut eines Geräteelements verwendet.

Programmcode: Attribute von Kanälen

Um auf Attribute eines Kanals zuzugreifen, ändern Sie folgenden Programmcode:

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

Programmcode: Attribute ermitteln

Um die identifizierenden Attribute für die einzelnen Kanäle abzurufen, ändern Sie folgenden Programmcode:

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

Programmcode: Aufrufen eines einzelnen Kanals

Um die identifizierenden Attribute für den direkten Zugriff auf einen Kanal zu verwenden, ändern Sie folgenden Programmcode:

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

Kanaltypen

Wert	Beschreibung
ChannelType.None	Der Kanaltyp ist ungültig.
ChannelType.Analog	Der Kanaltyp ist analog.
ChannelType.Digital	Der Kanaltyp ist digital.
ChannelType.Technology	Der Kanaltyp ist technologisch.

Kanal-IO-Typen

Wert	Beschreibung
ChannelIOType.None	Der Kanal-IO-Typ ist ungültig.
ChannelIOType.Input	Ein Eingangskanal.
ChannelIOType.Output	Ein Ausgangskanal.
ChannelIOType.Complex	Komplexe IO-Typen, z. B. für technologische Kanäle.

7.14 Funktionen in Netzwerken

7.14.1 Subnetz anlegen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Subnetze können auf zwei verschiedene Arten angelegt werden:

- Erstellen eines Subnetzes, das mit einer Schnittstelle verbunden ist: Die Art der Schnittstelle, an der das Subnetz angelegt wird, bestimmt die Art des Subnetzes
- Erstellen eines Subnetzes, das nicht mit einer Schnittstelle verbunden ist.

Programmcode: Erstellen eines Subnetzes, das mit einem Teilnehmer verbunden ist

Um ein Subnetz zu erstellen, ändern Sie folgenden Programmcode:

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

7.14 Funktionen in Netzwerken

Die folgenden Typkennungen werden verwendet:

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

Programmcode: Erstellen eines Subnetzes, das nicht mit einer Schnittstelle verbunden ist

Um ein Subnetz zu erstellen, ändern Sie folgenden Programmcode:

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

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

Die folgenden Typkennungen können verwendet werden:

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

7.14.2 Auf Subnetze zugreifen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Für mehrere netzwerkbezogene Merkmale, z. B. Zuordnung von Schnittstellen zu einem Subnetz, müssen Sie Subnetze im Projekt aufrufen. Subnetze sind typischerweise direkt auf Projektebene zusammengefasst.

Programmcode: Auf alle Subnetze eines Projekts zugreifen

Um auf alle Subnetze mit Ausnahme von internen Subnetzen eines Projekts zuzugreifen, ändern Sie folgenden Programmcode:

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

Programmcode: Auf ein spezifisches Subnetz zugreifen

Um auf ein bestimmtes Subnetz anhand des Namens zuzugreifen, ändern Sie folgenden Programmcode:

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

Attribute eines Subnetzes

Ein Subnetz hat folgende Attribute:

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

Netzwerktypen

Wert	Beschreibung
NetType.Unknown	Der Typ des Netzwerks ist unbekannt.
NetType.Ethernet	Der Typ des Netzwerks ist Ethernet.
NetType.Profibus	Der Typ des Netzwerks ist Profibus.
NetType.Mpi	Der Typ des Netzwerks ist MPI.
NetType.ProfibusIntegrated	Der Typ des Netzwerks ist integrierter Profibus.
NetType.Asi	Der Typ des Netzwerks ist ASI.
NetType.PcInternal	Der Typ des Netzwerks ist PC intern.
NetType.Ptp	Der Typ des Netzwerks ist PtP.
NetType.Link	Der Typ des Netzwerks ist Link.
NetType.Wan	Der Typ des Netzwerks ist Wide Area Network.

7.14.3 Auf interne Subnetze zugreifen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Wenn ein Geräteelement ein Subnetz bilden kann, verfügt es über die zusätzliche Funktionalität "Subnetzeigentümer". Um auf diese zusätzliche Funktionalität zuzugreifen, muss ein bestimmter Dienst des Geräteelements verwendet werden.

Programmcode: Subnetzeigentümerrolle abrufen

Um die Subnetzeigentümerrolle abzurufen, ändern Sie folgenden Programmcode:

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

Programmcode: Attribute eines Subnetzeigentümers

Um auf die Subnetze eines Subnetzeigentümers zuzugreifen, verwenden Sie folgenden Programmcode:

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

7.14.4 Typkennung von Subnetzen abrufen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Das Attribut `TypeIdentifier` wird zur Identifizierung eines Subnetzes verwendet. `TypeIdentifier` ist eine Zeichenkette, die aus mehreren Teilen besteht: `<TypeIdentifierType>:<SystemIdentifier>`

Mögliche Werte für `TypeIdentifierType` sind:

- System

SystemIdentifier

Subnetztyp	Systemkennung
PROFIBUS	Subnet.Profibus
MPI	Subnet.Mpi
Industrial Ethernet	Subnet.Ethernet
ASI	Subnet.Asi
PtP	Subnet.Ptp
PROFIBUS - integriert	Subnet.ProfibusIntegrated
PC - intern	null

Programmcode

Um die Typkennung für vom Benutzer verwaltbare und getrennt erstellbare Objekte für GSD abzurufen, ändern Sie den folgenden Programmcode:

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

7.14.5 Attribute eines Subnetzes aufrufen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Ein Subnetz bietet bestimmte obligatorische Attribute, die gelesen und/oder geschrieben werden können. Die Attribute sind nur verfügbar, wenn sie in der UI verfügbar sind. Der Schreibvorgang ist im Allgemeinen nur dann zulässig, wenn ein Attribut auch vom Benutzer in der UI geändert werden kann. Dies kann abhängig von der Art des Subnetzes variieren. Der Benutzer kann `DpCycleTime` nur dann festlegen, wenn `IsochronousMode` wahr und `DpCycleMinTimeAutoCalculation` falsch ist.

Attribute von Subnetzen vom Typ ASI

Attribut	Datentyp	Schreibbar	Zugriff	Beschreibung
Name	String	r/w		Name des Subnetzes.
NetType	NetType	r		Typ des Subnetzes
SubnetId	String	r	dynamisch	Eindeutige Kennung des Subnetzes. Die S7-Subnetz-ID setzt sich aus zwei Zahlen zusammen, die durch einen Bindestrich voneinander getrennt werden. Eine Zahl für das Projekt und eine Zahl für das Subnetz, z. B. 4493-1.

Attribute von Subnetzen vom Typ Ethernet

Attribut	Datentyp	Schreibbar	Zugriff	Beschreibung
Name	String	r/w		Name des Subnetzes.
NetType	NetType	r		Typ des Subnetzes
SubnetId	String	r/w	dynamisch	Eindeutige Kennung des Subnetzes. Die S7-Subnetz-ID setzt sich aus zwei Zahlen zusammen, die durch einen Bindestrich voneinander getrennt werden. Eine Zahl für das Projekt und eine Zahl für das Subnetz, z. B. 4493-1.
DefaultSubnet	Bool	r/w	dynamisch	Wahr, wenn das Subnetz ein Standardsubnetz ist. In einem Projekt ist höchstens ein Standardsubnetz vorhanden.

Attribute von Subnetzen vom Typ MPI

Attribut	Datentyp	Schreibbar	Zugriff	Beschreibung
Name	String	r/w		Name des Subnetzes.
NetType	NetType	r		Typ des Subnetzes
SubnetId	String	r/w	dynamisch	Eindeutige Kennung des Subnetzes. Die S7-Subnetz-ID setzt sich aus zwei Zahlen zusammen, die durch einen Bindestrich voneinander getrennt werden. Eine Zahl für das Projekt und eine Zahl für das Subnetz, z. B. 4493-1.
HighestAddress	Int	r/w	dynamisch	Höchste MPI-Adresse am Subnetz.
TransmissionSpeed	BaudRate	r/w	dynamisch	Wahr, wenn das Subnetz ein Standardsubnetz ist. In einem Projekt ist höchstens ein Standardsubnetz vorhanden.

Attribute von Subnetzen vom Typ PC internal

Attribut	Datentyp	Schreibbar	Zugriff	Beschreibung
Name	String	r		Name des Subnetzes.
NetType	NetType	r		Typ des Subnetzes
SubnetId	String	r	dynamisch	Eindeutige Kennung des Subnetzes. Die S7-Subnetz-ID setzt sich aus zwei Zahlen zusammen, die durch einen Bindestrich voneinander getrennt werden. Eine Zahl für das Projekt und eine Zahl für das Subnetz, z. B. 4493-1.

Attribute von Subnetzen vom Typ PROFIBUS

Attribut	Datentyp	Schreibbar	Zugriff	Beschreibung
Name	String	r/w		Name des Subnetzes.
NetType	NetType	r		Typ des Subnetzes
SubnetId	String	r/w	dynamisch	Eindeutige Kennung des Subnetzes. Die S7-Subnetz-ID setzt sich aus zwei Zahlen zusammen, die durch einen Bindestrich voneinander getrennt werden. Eine Zahl für das Projekt und eine Zahl für das Subnetz, z. B. 4493-1.
HighestAddress	Int	r/w	dynamisch	Höchste PROFIBUS-Adresse am Subnetz.
TransmissionSpeed	BaudRate	r/w	dynamisch	Wahr, wenn das Subnetz ein Standardsubnetz ist. In einem Projekt ist höchstens ein Standardsubnetz vorhanden.
BusProfile	Busprofil	r/w	dynamisch	Das PROFIBUS-Profil.
PbCableConfiguration	Bool	r/w	dynamisch	Wahr, um zusätzliche PROFIBUS-Netzwerkinstellungen zu aktivieren
PbRepeaterCount	Int	r/w	dynamisch	Anzahl der Repeater für Kupferleitung
PbCopperCableLength	double	r/w	dynamisch	Die Länge der Kupferleitung
PbOpticalComponentCount	Int	r/w	dynamisch	Anzahl der OLMs und OBTs von LWL-Leitung.
PbOpticalCableLength	double	r/w	dynamisch	Die Länge der LWL-Leitung für das PROFIBUS-Netzwerk in km.
PbOpticalRing	Bool	r/w	dynamisch	Wahr, wenn Busparameter für einen optischen Ring übernommen werden
PbOlmP12	Bool	r/w	dynamisch	Wahr, wenn OLM/P12 für die Berechnung des Busparameters aktiviert ist.
PbOlmG12	Bool	r/w	dynamisch	Wahr, wenn OLM/G12 für die Berechnung des Busparameters aktiviert ist.
PbOlmG12Eec	Bool	r/w	dynamisch	Wahr, wenn OLM/G12-EEC für die Berechnung des Busparameters aktiviert ist.

7.14 Funktionen in Netzwerken

Attribut	Datentyp	Schreibbar	Zugriff	Beschreibung
PbOlmG121300	Bool	r/w	dynamisch	Wahr, wenn OLM/G12-1300 für die Berechnung des Busparameters aktiviert ist.
PbAdditionalNetworkDevices	Bool	r/w	dynamisch	Wahr, wenn zusätzliche Busgeräte, die im Projekt nicht vorhanden sind, bei der Berechnung der Buszeiten berücksichtigt werden.
PbAdditionalDpMaster	Int	r/w	dynamisch	Anzahl der nicht konfigurierten DP-Master.
PbTotalDpMaster	Int	r	dynamisch	Gesamtanzahl der DP-Master
PbAdditionalPassiveDevice	Int	r/w	dynamisch	Anzahl der nicht konfigurierten DP-Slaves oder passiven Geräte.
PbTotalPassiveDevice	Int	r	dynamisch	Gesamtanzahl der DP-Slaves oder passiven Geräte.
PbAdditionalActiveDevice	Int	r/w	dynamisch	Anzahl der nicht konfigurierten aktiven Geräte mit FDL/FMS/S-/Kommunikationslast.
PbTotalActiveDevice	Int	r	dynamisch	Gesamtanzahl der aktiven Geräte mit FDL/FMS/S-/Kommunikationslast.
PbAdditionalCommunicationLoad	CommunicationLoad	r/w	dynamisch	Grobe Quantifizierung der Kommunikationslast
PbDirectDateExchange	Bool	r/w	dynamisch	Optimierung für direkten Datenaustausch.
PbMinimizeTslotForSlaveFailure	Bool	r/w	dynamisch	Minimierung für Zeitzuweisung für Slave-Fehler.
PbOptimizeCableConfiguration	Bool	r/w	dynamisch	Optimierung der Leitungskonfiguration.
PbCyclicDistribution	Bool	r/w	dynamisch	Wahr, wenn zyklische Verteilung der Busparameter aktiviert ist.
PbTslotInit	Int	r/w	dynamisch	Standardwert von Tslot.
PbTslot	Int	r	dynamisch	Warte-auf-Empfangszeit (slot time)
PbMinTsdr	Int	r/w	dynamisch	Minimale Dauer für die Protokollverarbeitung
PbMaxTsdr	Int	r/w	dynamisch	Maximale Dauer für die Protokollverarbeitung
PbTid1	Int	r	dynamisch	Leerlaufzeit 1
PbTid2	Int	r	dynamisch	Leerlaufzeit 2
PbTrdy	Int	r	dynamisch	Bereitschaftszeit
PbTset	Int	r/w	dynamisch	Einrichtungszeit
PbTqui	Int	r/w	dynamisch	Modulator-Ausklingzeit
PbTtr	int64	r/w	dynamisch	Der Ttr-Wert in t_Bit.

Attribut	Datentyp	Schreibbar	Zugriff	Beschreibung
PbTtrTypical	int64	r	dynamisch	Mittlere Antwortzeit am Bus
PbWatchdog	int64	r/w	dynamisch	Ansprechüberwachung
PbGapFactor	Int	r/w	dynamisch	GAP-Aktualisierungsfaktor
PbRetryLimit	Int	r/w	dynamisch	Maximale Anzahl an Wiederholungen
IsochronousMode	Bool	r/w	dynamisch	Wahr, wenn konstante Buszykluszeit aktiviert ist.
PbAdditionalPassivDeviceForIsochronousMode	Int	r/w	dynamisch	Anzahl der zusätzlichen OPs/PGs/TDs usw., die in dieser Netzwerksicht nicht konfiguriert sind.
PbTotalPassivDeviceForIsochronousMode	Int	r	dynamisch	Summe der konfigurierten und nicht konfigurierten Geräte, z. B. OPs/PGs/TDs usw.
DpCycleMinTimeAutoCalculation	Bool	r/w	dynamisch	Wahr, wenn automatische Berechnung und Einstellung der kürzesten DP-Zykluszeit aktiviert sind.
DpCycleTime	double	r/w	dynamisch	Die DP-Zykluszeit.
IsochronousTiToAutoCalculation	Bool	r/w	dynamisch	Wahr, wenn automatische Berechnung und Einstellung der Werte IsochronousTi und IsochronousTo aktiviert sind.
IsochronousTi	double	r/w	dynamisch	Time Ti (Zeit zum Lesen von Prozesswerten)
IsochronousTo	double	r/w	dynamisch	Time To (Zeit zum Ausgeben von Prozesswerten)

Attribute von Subnetzen vom Typ PROFIBUS Integrated

Attribut	Datentyp	Schreibbar	Zugriff	Beschreibung
Name	String	r/w		Name des Subnetzes.
NetType	NetType	r		Typ des Subnetzes
SubnetId	String	r/w	dynamisch	Eindeutige Kennung des Subnetzes. Die S7-Subnetz-ID setzt sich aus zwei Zahlen zusammen, die durch einen Bindestrich voneinander getrennt werden. Eine Zahl für das Projekt und eine Zahl für das Subnetz, z. B. 4493-1.
IsochronousMode	Bool	r	dynamisch	Konstante Buszykluszeit aktiviert.
DpCycleMinTimeAutoCalculation	Bool	r/w	dynamisch	Wahr, wenn automatische Berechnung und Einstellung der kürzesten DP-Zykluszeit aktiviert sind.
DpCycleTime	double	r/w	dynamisch	Die DP-Zykluszeit.
IsochronousTiToAutoCalculation	Bool	r/w	dynamisch	Wahr, wenn automatische Berechnung und Einstellung der Werte IsochronousTi und IsochronousTo aktiviert sind.

7.14 Funktionen in Netzwerken

Attribut	Datentyp	Schreibbar	Zugriff	Beschreibung
IsochronousTi	double	r/w	dynamisch	Time Ti (Zeit zum Lesen von Prozesswerten)
IsochronousTo	double	r/w	dynamisch	Time To (Zeit zum Ausgeben von Prozesswerten)

Programmcode

Um die Attribute eines Subnetzes abzurufen oder festzulegen, ändern Sie folgenden Programmcode:

```
Subnet subnet = ...;

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

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

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

Baud-Raten

Wert	Beschreibung
BaudRate.None	Die Baud-Rate ist unbekannt.
BaudRate.Baud9600	9,6 kBaud
BaudRate.Baud19200	19,2 kBaud
BaudRate.Baud45450	45,45 kBaud
BaudRate.Baud93700	93,75 kBaud
BaudRate.Baud187500	187,5 kBaud
BaudRate.Baud500000	500 kBaud
BaudRate.Baud1500000	1,5 MBaud
BaudRate.Baud3000000	3 MBaud
BaudRate.Baud6000000	6 MBaud
BaudRate.Baud12000000	12 MBaud

Busprofile

Wert	Beschreibung
BusProfile.None	Das Busprofil ist unbekannt.
BusProfile.DP	Der Typ des Netzwerks ist DP.
BusProfile.Standard	Der Typ des Netzwerks ist Standard.

Wert	Beschreibung
BusProfile.Universal	Der Typ des Netzwerks ist Universal.
BusProfile.UserDefined	Der Typ des Netzwerks ist benutzerdefiniert.

Kommunikationslast

Wert	Beschreibung
CommunicationLoad.None	Keine gültige Kommunikationslast.
CommunicationLoad.Low	Typisch für DP, keine größere Datenkommunikation außer DP.
CommunicationLoad.Medium	Typisch für Mischbetrieb von DP und anderen Kommunikationsdiensten (z. B. S7-Kommunikation), wenn DP hohe Zeitanforderungen hat und bei mittlerem azyklischen Kommunikationsaufkommen.
CommunicationLoad.High	Für Mischbetrieb von DP und anderen Kommunikationsdiensten (z. B. S7-Kommunikation), wenn DP geringe Zeitanforderungen hat und bei hohem azyklischen Kommunikationsaufkommen.

7.14.6 Globales Subnetz löschen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode

Um ein globales Subnetz innerhalb eines Projekts zu löschen, ändern Sie folgenden Programmcode:

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

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

7.14.7 Alle Beteiligten eines Subnetzes enumerieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Enumeration aller Beteiligten eines Subnetzes.

Programmcode

Um DP-Mastersysteme aus dem Subnetz zu enumerieren, ändern Sie folgenden Programmcode:

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

7.14.8 IO-Systeme eines Subnetzes enumerieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Die Enumeration von IoSystem stellt alle IO-Systeme, die sich auf dem Subnetz befinden, bereit. Das Mastersystem und das IO-System werden beide von der Klasse IoSystem dargestellt.

Programmcode

Um DP-Mastersysteme aus dem Subnet zu enumerieren, ändern Sie folgenden Programmcode:

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

7.14.9 Auf Teilnehmer zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Die Rollenschnittstelle aggregiert Teilnehmer: Um auf Attribute zuzugreifen, die mit der Adress- und Subnetzuordnung einer Schnittstelle zusammenhängen.

Der Name eines Teilnehmers wird in den Attributen einer Schnittstelle im TIA Portal angezeigt. Die Teilnehmerkennung ist eine einzigartige Kennung für jeden auf einer Schnittstelle erfassten Teilnehmer. Der Wert kann nur über TIA Portal Openness angezeigt werden.

Programmcode

Um auf alle Teilnehmer einer Schnittstelle zuzugreifen, ändern Sie folgenden Programmcode:

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

Die meisten Schnittstellen haben nur einen einzigen Teilnehmer, daher wird normalerweise der erste Teilnehmer verwendet.

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

Teilnehmer stellen ihre Namen und Typen und Teilnehmer-IDs als Attribute zur Verfügung:

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

7.14.10 Attribute eines Teilnehmers aufrufen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Ein Geräteelement bietet bestimmte obligatorische Attribute, die gelesen und/oder geschrieben werden können. Die Attribute sind nur verfügbar, wenn sie in der UI verfügbar sind. Der Schreibvorgang ist im Allgemeinen nur dann zulässig, wenn ein Attribut auch vom Benutzer in der UI geändert werden kann. Dies kann abhängig von der Art des Geräteelements variieren. Der Benutzer kann RouterAddress nur dann festlegen, wenn RouterUsed wahr ist. Wenn der Benutzer die SubnetMask am IO-Controller ändert, dann wird auch die Subnetzmaske auf allen IO-Devices auf denselben Wert geändert.

Attribute eines Teilnehmers vom Typ ASI

Attribut	Datentyp	Schreibbar	Zugriff	Beschreibung
Name	String	r		Name des Teilnehmers.
NodeID	String	r		ID des Teilnehmers.
NodeType	NetType	r		Ein Teilnehmer ruft seinen Typen vom Subnetz ab.
Address	String	r/w	dynamisch	Weitere Attribute für AS-i-Slaves.

Attribute eines Teilnehmers vom Typ Ethernet

Attribut	Datentyp	Schreibbar	Zugriff	Beschreibung
Name	String	r		Name des Teilnehmers.
NodeID	String	r		ID des Teilnehmers.

Attribute	Datentyp	Schreibbar	Zugriff	Beschreibung
NodeType	NetType	r/w manch mal r/o		Ein Teilnehmer ruft seinen Typen vom Subnetz ab.
UselsoProtocol	Bool	r/w	dynamisch	True, wenn ISO-Protokoll verwendet werden muss
MacAddress	String	r/w	dynamisch	z. B. 01-80-C2-00-00-00
UselpProtocol	Bool	r/w	dynamisch	Dieser Wert kann auch dann gelesen werden, wenn er an der entsprechenden TIA UI-Steuerung nicht sichtbar ist.
IpProtocolSelection	Enum	r/w	dynamisch	
Address	String	r/w	dynamisch	nur IPv4 und nicht IPv6 wird unterstützt
SubnetMask	String	r/w	dynamisch	
UseRouter	Bool	r/w	dynamisch	
RouterAddress	String	r/w	dynamisch	
DhcpClientId	String	r/w	dynamisch	
PnDeviceNameSetDirectly	Bool	r/w	dynamisch	PROFINET-Gerätename wird direkt am Gerät festgelegt. Nicht für jedes Gerät verfügbar.
PnDeviceNameAutoGeneration	Bool	r/w	dynamisch	PROFINET-Gerätename wird automatisch erstellt.
PnDeviceName	String	r/w	dynamisch	Eindeutiger Name im Subnetz.
PnDeviceNameConverted	String	r	dynamisch	Gerätename für systeminterne Nutzung umgewandelt.

Attribute eines Teilnehmers vom Typ MPI

Attribut	Datentyp	Schreibbar	Zugriff	Beschreibung
Name	String	r		Name des Teilnehmers.
NodeID	String	r		ID des Teilnehmers.
NodeType	NetType	r		Ein Teilnehmer ruft seinen Typen vom Subnetz ab.
Address	String	r/w	dynamisch	

Attribute eines Teilnehmers vom Typ PC internal

Attribut	Datentyp	Schreibbar	Zugriff	Beschreibung
Name	String	r		Name des Teilnehmers.
NodeID	String	r		ID des Teilnehmers.
NodeType	NetType	r		Ein Teilnehmer ruft seinen Typen vom Subnetz ab.

Attribute eines Teilnehmers vom Typ PROFIBUS

Attribut	Datentyp	Schreibbar	Zugriff	Beschreibung
Name	String	r		Name des Teilnehmers.
NodeID	String	r		ID des Teilnehmers.
NodeType	NetType	r		Ein Teilnehmer ruft seinen Typen vom Subnetz ab.
Address	String	r/w	dynamisch	Netzwerkadresse des Knotens. Der Typ der Adresse ist abhängig vom Knotentyp (z. B. IP-Adresse für PROFINET-Knoten, PROFIBUS-Adresse für PROFIBUS-Knoten)

Attribute eines Teilnehmers vom Typ PROFIBUS integrated

Attribut	Datentyp	Schreibbar	Zugriff	Beschreibung
Name	String	r		Name des Teilnehmers.
NodeID	String	r		ID des Teilnehmers.
NodeType	NetType	r		Ein Teilnehmer ruft seinen Typen vom Subnetz ab.
Address	String	r	dynamisch	

Programmcode: Attribute eines Teilnehmers

Um die Attribute eines Teilnehmers abzurufen oder festzulegen, ändern Sie folgenden Programmcode:

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

Programmcode: Dynamische Attribute

Um dynamische Attribute eines Teilnehmers abzurufen oder festzulegen, ändern Sie folgenden Programmcode:

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

Protokollauswahl

Wert	Beschreibung
IpProtocolSelection.None	Fehlerwert
IpProtocolSelection.Project	IP Suite innerhalb des Projekts konfiguriert.
IpProtocolSelection.Dhcp	IP Suite mit Hilfe von DHCP-Protokoll verwaltet. ID des DHCP-Client notwendig.
IpProtocolSelection.UserProgram	IP Suite mit Hilfe von FB (Funktionsbaustein) festgelegt.
IpProtocolSelection.OtherPath	IP Suite mit Hilfe von anderen Methoden festgelegt, z. B. PST Tool.
IpProtocolSelection.VialoController	IP Suite mit Hilfe von IO-Controller in Runtime festgelegt.

Netzwerktyp

Wert	Beschreibung
NetType.Asi	Netzwerktyp ist ASI.
NetType.Ethernet	Netzwerktyp ist Ethernet.
NetType.Link	Netzwerktyp ist Link.
NetType.Mpi	Netzwerktyp ist MPI.
NetType.PcInternal	Netzwerktyp ist PC internal.
NetType.Profibus	Netzwerktyp ist PROFIBUS.
NetType.ProfibusIntegrated	Netzwerktyp ist PROFIBUS integrated.
NetType.Ptp	Netzwerktyp ist PTP.
NetType.Wan	Netzwerktyp ist Wide Area Network (WAN).
NetType.Unknown	Netzwerktyp ist unbekannt.

7.14.11 Teilnehmer mit einem Subnetz verbinden

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode

Um einen Teilnehmer (Gerät, Schnittstelle) einem Netzwerk zuzuweisen, ändern Sie folgenden Programmcode:

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

7.14.12 Teilnehmer von einem Subnetz trennen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode

Um einen Teilnehmer (Gerät, Schnittstelle) von einem Netzwerk zu trennen, ändern Sie folgenden Programmcode:

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

7.14.13 IO-System erstellen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Ein IO-System wird durch Aufrufen der Aktion IoController.CreateIoSystem("name") auf einem Objekt vom Typ IoController erstellt. Wenn der name Null oder String.Empty ist, dann wird der Standardname verwendet. Der IO-Controller wird erfasst durch Zugreifen auf das Objekt des Attributs IoControllers an der NetworkInterface. Der Navigator des IoControllers gibt ein IoController-Objekt zurück.

Voraussetzungen für das Erstellen eines IO-Systems:

- Die Schnittstelle des IO-Controllers ist mit einem Subnetz verbunden.
- Der IO-Controller besitzt kein IO-System.

Programmcode

Um ein IO-System zu erstellen, ändern Sie folgenden Programmcode:

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

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

7.14.14 Attribute eines IO-Systems aufrufen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Das Mastersystem und das IO-System werden beide von der Klasse IoSystem dargestellt.

Programmcode: Attribute eines IO-Systems

Um die Attribute des IoSystem abzurufen, ändern Sie folgenden Programmcode:

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

Programmcode: Subnetz eines IO-Systems

Um zu dem Subnetz zu navigieren, das dem IO-System zugeordnet ist, ändern Sie folgenden Programmcode:

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

7.14.15 IO-Connector mit einem IO-System verbinden

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Verwenden Sie die Aktion ConnectToIoSystem(IoSystem ioSystem) des IoConnector, um einen Profinet- oder DP-IoConnector mit einem vorhandenen IO-System zu verbinden.

Verwenden Sie die Aktion GetIoController, um zum dezentralen IoController zu navigieren. Weitere Informationen dazu, wie Sie zum lokalen IoConnector und dem IO-System navigieren, finden Sie unter Mastersystem oder IO-System einer Schnittstelle abrufen (Seite 211).

Voraussetzungen:

- Der IoConnector ist noch nicht mit einem IO-System verbunden.
- Die IoConnector-Schnittstelle ist mit dem gleichen Subnetz verbunden wie die Schnittstelle des gewünschten IoController.

Programmcode

Ändern Sie folgenden Programmcode:

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

7.14.16 Mastersystem oder IO-System einer Schnittstelle abrufen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Der Dienst NetworkInterface liefert dem Navigator IoControllers, wovon jeder IoController wiederum dem Navigator IoSystem liefert. Das Mastersystem und das IO-System werden beide von der Klasse IoSystem dargestellt. Das IO-Device und der Slave werden beide IO-Device genannt.

- Der Navigator des IoControllers gibt IoController-Objekte zurück, wenn die Netzwerkschnittstelle ein IO-System haben kann. Im Moment wird nur ein IO-Controller zurückgegeben.
- Der Navigator des IoConnectors gibt IoConnector-Objekte zurück, wenn die Netzwerkschnittstelle als IO-Gerät mit einem IO-System verbunden werden kann. Im Moment wird nur ein IO-Connector zurückgegeben.

Programmcode: IO-System des IoController abrufen

Um das IO-System des IoController abzurufen, ändern Sie folgenden Programmcode:

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

Programmcode: IO-System des IoConnector abrufen

Um das IO-System des IoConnector abzurufen, ändern Sie folgenden Programmcode:

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

7.14.17 IO-Controller abrufen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode

Derzeit sind nur Konfigurationen mit einem IoController möglich. Ein IoController verfügt nicht über modellierte Attribute oder Aktionen.

Programmcode

Um den IO Controller abzurufen, ändern Sie folgenden Programmcode:

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

7.14.18 IO-Connector abrufen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Ein IoConnector liefert die modellierten Attribute oder Aktionen.

Die folgenden Attribute und Aktionen sind am IoConnector verfügbar:

Aktionen

Aktion	Signatur	Beschreibung
ConnectToIoSystem	void ConnectToIoSystem(Siemens.Engineering.HW.IoSystem ioSystem)	Zum Verbinden eines IoConnector mit einem vorhandenen DP-Mastersystem
DisconnectFromIoSystem	void DisconnectFromIoSystem()	Zum Trennen eines IoConnector von einem vorhandenen IO-System
GetIoController	Siemens.Engineering.HW.IoController GetIoController()	Gibt den IO-Controller für diesen Connector zurück

Links

Link	Typ	Zugriff
ConnectedToIoSystem	Siemens.Engineering.HW.IoSystem	schreibgeschützt

Programmcode

Um den IO Connector abzurufen, ändern Sie folgenden Programmcode:

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

7.14.19 IO-Connector von einem IO-System oder einem DP-Mastersystem trennen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Verwenden Sie die Aktion DisconnectFromIoSystem() des IoConnector, um einen IoConnector von einem vorhandenen IO-System oder einem vorhandenen DP-Mastersystem zu trennen.

Weitere Informationen dazu, wie Sie zum lokalen IoConnector und dem IO-System navigieren, finden Sie unter Mastersystem oder IO-System einer Schnittstelle abrufen (Seite 211).

Programmcode

Ändern Sie folgenden Programmcode:

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

ioConnector.DisconnectFromIoSystem();
```

7.14.20 Attribute eines DP-Mastersystems aufrufen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Ein DP-Mastersystem bietet bestimmte Attribute, die gelesen und/oder geschrieben werden können. Die Attribute sind nur verfügbar, wenn sie in der UI verfügbar sind. Der Schreibvorgang ist im Allgemeinen nur dann zulässig, wenn ein Attribut auch vom Benutzer in der UI geändert werden kann. Dies kann abhängig vom DP-Master und den DP-Slaves variieren, die diesem DP-Mastersystem zugeordnet werden.

Attribute eines DP-Mastersystems

Attribut	Datentyp	Schreibbar	Zugriff	Beschreibung
Name	String	r/w		
Number	Int	r/w		Das Attribut Number übernimmt Werte, die nicht über die UI festgelegt werden können. In einem solchen Fall schlägt die Übersetzung fehl.

Programmcode: Attribute abrufen

Um die Attribute abzurufen, ändern Sie folgenden Programmcode:

```
IoSystem dpMastersystem = ...;

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

Programmcode: Attribute festlegen

Um die Attribute einzustellen, ändern Sie folgenden Programmcode:

```
IoSystem dpMastersystem = ...;

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

7.14.21 Attribute eines PROFINET IO-Systems aufrufen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Ein IO-System bietet bestimmte Attribute, die gelesen und/oder geschrieben werden können. Die Attribute sind nur verfügbar, wenn sie in der UI verfügbar sind. Der Schreibvorgang ist im Allgemeinen nur dann zulässig, wenn ein Attribut auch vom Benutzer in der UI geändert werden kann. Dies kann abhängig vom IO-Controller und den IO-Devices variieren, die diesem IO-System zugeordnet werden.

Attribute eines PROFINET IO-Systems

Attribut	Datentyp	Schreibbar	Zugriff	Beschreibung
MultipleUseIoSystem	Bool	r/w	dynamisch	
Name	String	r/w		
Number	Int	r/w		Das Attribut Number übernimmt Werte, die nicht über die UI festgelegt werden können. In einem solchen Fall schlägt die Übersetzung fehl.
UseIoSystemNameAsDeviceNameExtension	Bool	r/w	dynamisch	Wenn MultipleUseIoSystem auf TRUE gesetzt ist, dann wird UseIoSystemNameAsDeviceNameExtension auf FALSE gesetzt und der Schreibzugriff ist nicht möglich.
MaxNumberIWlanLinksPerSegment	Int	r/w	dynamisch	

Programmcode: Attribute abrufen

Um die Attribute abzurufen, ändern Sie folgenden Programmcode:

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

Programmcode: Attribute festlegen

Um die Attribute einzustellen, ändern Sie folgenden Programmcode:

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

Programmcode: Attribute bei dynamischem Zugriff abrufen

Um die Werte von dynamischen Attributen abzurufen, ändern Sie folgenden Programmcode:

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

Programmcode: Attribute bei dynamischem Zugriff festlegen

Um die Werte der dynamischen Attribute festzulegen, ändern Sie folgenden Programmcode:

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

7.14.22 DP-Mastersystem löschen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode: PROFINET IO-System löschen

Um ein PROFINET IO-System zu löschen, ändern Sie den folgenden Programmcode:

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

7.14.23 Profinet IO-System löschen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode

Um ein Profinet IO-System zu löschen, ändern Sie den folgenden Programmcode:

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

7.14.24 DP-Mastersystem erstellen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Ein DP-Mastersystem wird durch Aufrufen der Aktion CreateIoSystem(string nameOfIoSystem) auf einem Objekt vom Typ IoController erstellt. Der IO-Controller wird erfasst durch Zugreifen auf das Objekt des Attributs IoControllers an der NetworkInterface.

Voraussetzungen für das Erstellen eines DP-Mastersystems:

- Die Schnittstelle des IO-Controllers ist mit einem Subnetz verbunden.
- Der IO-Controller besitzt kein IO-System.

Programmcode

Um ein DP-Mastersystem zu erstellen, ändern Sie folgenden Programmcode:

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

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

7.14.25 Port-Verschaltungsinformationen des Port-Geräteelements aufrufen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

NetworkPort bietet die Verknüpfung ConnectedPorts, die eine Enumeration von Ports ist, um auf alle verschalteten Partnerports eines Ports zuzugreifen.

Es können nur Ports miteinander verschaltet werden, die auch in der TIA UI verschaltet werden können. Beispielsweise können zwei Ports derselben Ethernet-Schnittstelle nicht miteinander verschaltet werden. Eine wiederherstellbare Ausnahme wird ausgelöst,

- wenn bereits eine Verschaltung mit demselben Partnerport besteht
- wenn Sie versuchen, zwei Ports zu verschalten, die nicht miteinander verschaltet werden können
- wenn Sie versuchen, eine zweite Verschaltung zu einem Port anzulegen, der keine alternativen Partner unterstützt

Programmcode: Portverschaltung abrufen

Um die Port-Verschaltungsinformationen eines Port-Geräteelements abzurufen, ändern Sie den folgenden Programmcode:

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

Programmcode: Portverschaltungen erstellen

Ändern Sie folgenden Programmcode:

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

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

Programmcode: Portverschaltung löschen

Ändern Sie folgenden Programmcode:

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

7.14.26 Attribute der Port-Verschaltung**Attribute für Port-Verschaltungen**

TIA Portal Openness unterstützt die folgenden Attribute für Port-Verschaltungen. Wenn die Attribute in der Benutzeroberfläche verfügbar sind, sind die entsprechenden Attribute auch über TIA Portal Openness zugänglich. Wenn der Anwender Zugriff hat und die Attribute in der Benutzeroberfläche ändern kann, können die Attribute auch über TIA Portal Openness geändert werden.

Attributname	Datentyp	Schreibbar	Zugriff	Beschreibung
MediumAttachment-Type	MediumAttachment-Type	Schreibgeschützt	Dynamisches Attribut	
CableName	CableName	Lesen-Schreiben	Dynamisches Attribut	
AlternativePartnerPorts	Bool	Lesen-Schreiben	Dynamisches Attribut	Nur verfügbar, wenn die Funktionalität des Werkzeugwechslers unterstützt wird.
SignalDelaySelection	SignalDelaySelection	Lesen-Schreiben	Dynamisches Attribut	
CableLength	CableLength	Lesen-Schreiben	Dynamisches Attribut	
SignalDelayTime	Double	Lesen-Schreiben	Dynamisches Attribut	

Enum-Werte von Port-Verschaltungsattributen

Die Enum MediumAttachmentType hat folgende Werte.

Wert	Beschreibung
MediumAttachmentType.None	Anbindungstyp kann nicht ermittelt werden.
MediumAttachmentType.Copper	Anbindungstyp ist Kupfer.
MediumAttachmentType.FiberOptic	Anbindungstyp ist Glasfaser.

Die Enum CableName hat folgenden Wert

Wert	Beschreibung
CableName.None	Kein Kabelname angegeben
CableName.FO_Standard_Cable_9	FO-Standardkabel GP (9 µm)
CableName.Flexible_FO_Cable_9	Flexibles FO-Kabel (9 µm)
CableName.FO_Standard_Cable_GP_50	FO-Standardkabel GP (50 µm)
CableName.FO_Trailing_Cable_GP	FO-Schleppkabel / GP
CableName.FO_Ground_Cable	FO-Erdungskabel
CableName.FO_Standard_Cable_62_5	FO-Standardkabel (62,5 µm)
CableName.Flexible_FO_Cable_62_5	Flexibles FO-Kabel (62,5 µm)
CableName.POF_Standard_Cable_GP	POF-Standardkabel GP
CableName.POF_Trailing_Cable	POF-Schleppkabel
CableName.PCF_Standard_Cable_GP	PCF-Schleppkabel / GP
CableName.GI_POF_Standard_Cable	GI-POF-Standardkabel
CableName.GI_POF_Trailing_Cable	GI-POF-Schleppkabel
CableName.GI_PCF_Standard_Cable	GI-PCF-Standardkabel
CableName.GI_PCF_Trailing_Cable	GI-PCF-Schleppkabel
CableName.GI_POF_Standard_Cable	GI-POF-Standardkabel
CableName.GI_POF_Trailing_Cable	GI-POF-Schleppkabel
CableName.GI_PCF_Standard_Cable	GI-PCF-Standardkabel
CableName.GI_PCF_Trailing_Cable	GI-PCF-Schleppkabel

Die Enum SignalDelaySelection hat folgende Werte.

Wert	Beschreibung
SignalDelaySelection.None	
SignalDelaySelection.CableLength	CableLength dient zur Festlegung der Signalverzögerung.
SignalDelaySelection.SignalDelayTime	CableDelayTime dient zur Festlegung der Signalverzögerung.

Die Enum CableLength hat folgende Werte

Wert	Beschreibung
CableLength.None	Kabellänge ist nicht angegeben.
CableLength.Length20m	Kabellänge ist 20 m.
CableLength.Length50m	Kabellänge ist 50 m.
CableLength.Length100m	Kabellänge ist 100 m.
CableLength.Length1000m	Kabellänge ist 1000 m.
CableLength.Length3000m	Kabellänge ist 3000 m.

Attribute von Port-Optionen

Die Attribute von Port-Optionen werden nachfolgend dargestellt.

Attributname	Datentyp	Schreibbar	Zugriff
PortActivation	Bool	Lesen-Schreiben	Dynamisches Attribut
TransmissionRateAnd-Duplex	TransmissionRateAnd-Duplex	Lesen-Schreiben	Dynamisches Attribut
PortMonitoring	Bool	Lesen-Schreiben	Dynamisches Attribut
TransmissionRateAuto-Negotiation	Bool	Lesen-Schreiben	Dynamisches Attribut
EndOfDetectionOfAc-cessibleDevices	Bool	Lesen-Schreiben	Dynamisches Attribut
EndOfTopologyDisco-very	Bool	Lesen-Schreiben	Dynamisches Attribut
EndOfSyncDomain	Bool	Lesen-Schreiben	Dynamisches Attribut

Die Enum TransmissionRateAndDuplex hat folgende Werte.

Wert	Beschreibung
TransmissionRateAndDuplex.None	
TransmissionRateAndDuplex.Automatic	Automatisch
TransmissionRateAndDuplex.AUI10Mbps	10 Mbps AUI
TransmissionRateAndDuplex.TP10MbpsHalfDu-plex	TP 10 Mbps Halbduplex
TransmissionRateAndDuplex.TP10MbpsFullDu-plex	TP 10 Mbps Vollduplex
TransmissionRateAndDuplex.AsyncFi-ber10MbpsHalfDuplex	Asynchrone Glasfaser 10 Mbit/s Halbduplexmo-dus
TransmissionRateAndDuplex.AsyncFi-ber10MbpsFullDuplex	Asynchrone Glasfaser 10 Mbit/s Vollduplexmodus

Wert	Beschreibung
TransmissionRateAndDuplex.TP100MbpsHalfDuplex	TP 100 Mbps Halbduplex
TransmissionRateAndDuplex.TP100MbpsFullDuplex	TP 100 Mbps Vollduplex
TransmissionRateAndDuplex.FO100MbpsFullDuplex	FO 100 Mbps Vollduplex
TransmissionRateAndDuplex.X1000MbpsFullDuplex	X1000 Mbps Vollduplex
TransmissionRateAndDuplex.FO1000MbpsFullDuplexLD	FO 1000 Mbps Vollduplex LD
TransmissionRateAndDuplex.FO1000MbpsFullDuplex	FO 1000 Mbps Vollduplex
TransmissionRateAndDuplex.TP1000MbpsFullDuplex	TP 1000 Mbps Vollduplex
TransmissionRateAndDuplex.FO10000MbpsFullDuplex	FO 10000 Mbps Vollduplex
TransmissionRateAndDuplex.FO100MbpsFullDuplexLD	FO 100 Mbps Vollduplex LD
TransmissionRateAndDuplex.POFPCF100MbpsFullDuplexLD	POF/PCF 100 Mbps Vollduplex

7.14.27 Attribute eines Ports aufrufen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Ein Geräteelement, das zugleich ein Port ist, bietet gegenüber einem einfachen Geräteelement zusätzliche Funktionalität.

- Es ist möglich, auf die verknüpften Partnerports des Ports zuzugreifen.
- Es ist möglich, auf die Schnittstelle des Ports zuzugreifen.

Um auf diese zusätzliche Funktionalität zuzugreifen, muss die NetworkPort-Funktion, ein bestimmter Dienst des Geräteelements, verwendet werden.

Programmcode: Auf einen Port zugreifen

Um auf Attribute eines Kanals zuzugreifen, ändern Sie folgenden Programmcode:

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

Attribute eines Ports

Ein Port hat folgende Attribute:

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

7.14.28 DP-Mastersysteme eines Subnetzes enumerieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Die Enumeration von IoSystem stellt alle DP-Mastersysteme, die sich auf dem Subnetz befinden, bereit. Das Mastersystem und das IO-System werden beide von der Klasse IoSystem dargestellt.

Programmcode

Um DP-Mastersysteme aus dem Subnetz zu enumerieren, ändern Sie folgenden Programmcode:

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

7.14.29 Zugeordnete IO-Connectors enumerieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Das Mastersystem und das IO-System werden beide von der Klasse IoSystem dargestellt.

Wird verwendet für:

- Enumeration zugeordneter IO-Connectors des DP-Mastersystems
- Enumeration zugeordneter IO-Connectors des Profinet-IO-Systems

Programmcode

Um zugeordnete IO Connectors des DP-Mastersystems zu enumerieren, ändern Sie folgenden Programmcode:

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

7.14.30 DP-IO-Connector mit einem DP-Mastersystem verbinden

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Verwenden Sie die Aktion ConnectToIoSystem(IoSystem ioSystem) des IoConnector, um einen IoConnector mit einem vorhandenen DP-Mastersystem zu verbinden.

Verwenden Sie die Aktion GetIoController, um zum dezentralen IoController zu navigieren. Weitere Informationen dazu, wie Sie zum lokalen IoConnector und dem IO-System navigieren, finden Sie unter Mastersystem oder IO-System einer Schnittstelle abrufen (Seite 211).

Voraussetzungen:

- Der IoConnector ist noch nicht mit einem IO-System verbunden.
- Die IoConnector-Schnittstelle ist mit dem gleichen Subnetz verbunden wie die Schnittstelle des gewünschten IoController.

Programmcode

Ändern Sie folgenden Programmcode:

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

7.14.31 Zugriff auf AS-i-Profil und Parameterattribute für virtuelle Slaves

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Das TIA Portal Openness unterstützt die folgenden zusätzlichen Parameter des AS-i-Profil für die virtuellen Slaves des CTT5 AS-i Slave unter Verwendung der StructuredData-Namen:

Name	Beschreibung
AsiProfileVirtualSlave1	Enthält die AS-i Profilparameter für den virtuellen Slave 1
AsiProfileVirtualSlave2	Enthält die AS-i Profilparameter für den virtuellen Slave 2
AsiProfileVirtualSlave3	Enthält die AS-i Profilparameter für den virtuellen Slave 3

Die AS-i-Slave-Parameter für die virtuellen Slaves sind im Folgenden aufgeführt:

Name	Beschreibung
AsiParamterVirtualSlave1	Enthält die AS-i-Parameter für den virtuellen Slave 1
AsiParamterVirtualSlave2	Enthält die AS-i-Parameter für den virtuellen Slave 2
AsiParamterVirtualSlave3	Enthält die AS-i-Parameter für den virtuellen Slave 3

Programmcode

Ändern Sie folgenden Programmcode, um die zusätzlichen Attribute von AS-i-Slaves abzurufen und festzulegen:

```
DeviceItem slaveModule= ...;
var structuredDataNamesProfile = new[]
{
    "AsiProfileVirtualSlave1", "AsiProfileVirtualSlave2", "AsiProfileVirtualSlave3"
};
var structuredDataNamesAsiParam = new[]
{
    "AsiParamVirtualSlave1", "AsiParamVirtualSlave2", "AsiParamVirtualSlave3"
};
var attributeNamesProfile = new[] {"AsiProfileID", "AsiProfileIO", "AsiProfileID2",
    "AsiProfileID1"};
string attributeNameAsiParam = "AsiSlaveParameter";
foreach (var structuredDataName in structuredDataNamesProfile)
{
    foreach (var attributeName in attributeNamesProfile)
    {
        StructuredData structuredData =
            (StructuredData)slaveModule.GetAttribute(structuredDataName);
        //get
        UInt32 attributeValue = (UInt32)structuredData.GetAttribute(attributeName);
        //set
        slaveHead.SetAttribute(attributeName, (UInt32)5);
    }
}
foreach (var structuredDataName in structuredDataNamesAsiParam)
{
    StructuredData structuredData =
        (StructuredData)slaveModule.GetAttribute(structuredDataName);
    //get
    UInt32 attributeValue = (UInt32)structuredData.GetAttribute(attributeNameAsiParam);
    //set
    slaveHead.SetAttribute(attributeName, (UInt32)5);
}
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.15 Funktionen auf Geräten

7.15.1 Obligatorische Attribute von Geräten

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Jedes Gerät oder Geräteelement besitzt bestimmte obligatorische Attribute, die gelesen und/oder geschrieben werden können. Diese Attribute sind immer die gleichen wie in der Benutzeroberfläche des TIA Portals.

In Openness werden die folgenden Attribute unterstützt:

Attributname	Datentyp	Schreibbar	Zugriff	Kommentar
Author	String	read/write	dynamisch	
Comment	String	read/write	dynamisch	manchmal schreibgeschützter Zugriff
CommentML	MultilingualTextItem	read/write	dynamisch	manchmal schreibgeschützter Zugriff
IsGsd	Bool	read		WAHR, wenn die Gerätebeschreibung über GSD/GSDML installiert wird
Name	String	read/write		manchmal schreibgeschützter Zugriff
TypeIdentifier	String	read		
TypeName	String	read	dynamisch	

Programmcode: Obligatorische Attribute eines Geräts

Ändern Sie den folgenden Programmcode, um die obligatorischen Attribute eines Geräts abzurufen:

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

Programmcode: Obligatorische Attribute bei dynamischem Zugriff

Ändern Sie den folgenden Programmcode, um die Attribute bei dynamischem Zugriff aufzurufen:

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

7.15.2 Typkennung von Geräten und Geräteelementen abrufen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Das Attribut `TypeIdentifier` wird zur Identifizierung eines Hardwareobjekts verwendet, das über TIA Portal Openness API erstellt werden kann. `TypeIdentifier` ist eine Zeichenkette, die aus mehreren Teilen besteht: `<TypeIdentifierType>:<Identifier>`

Mögliche Werte für `TypeIdentifierType` sind:

- OrderNumber
- GSD
- System

OrderNumber

OrderNumber ist die allgemeine TypeIdentifier für alle im Hardware-Katalog vorhandenen Module.

Format der Typkennung	Beispiel	Besonderheiten
<OrderNumber>	OrderNumber:3RK1 200-0CE00-0AA2	
<OrderNumber>/<FirmwareVersion>	OrderNumber:6ES7 510-1DJ01-0AB0/V2.0	Die Firmware-Version ist optional, wenn diese gar nicht oder nur eine Version im System vorhanden ist. Bitte beachten
<OrderNumber>//<AdditionalTypeIdentifier>	OrderNumber:6AV2 124-2DC01-0AX0//Landscape	Die zusätzliche Typkennung kann notwendig sein, wenn OrderNumber und FirmwareVersion keinen eindeutigen Treffer im System haben.

Hinweis

Es gibt einige wenige Module im Hardware-Katalog, die Platzhalterzeichen in der Bestellnummer enthalten, welche eine bestimmte Gruppe realer Hardware repräsentiert, z. B. die verschiedenen Längen der S7-300 Baugruppenträger. In diesem Fall können sowohl die OrderNumber als auch die Platzhalter-OrderNumber verwendet werden, um eine Instanz des Hardwareobjekts zu erstellen. Platzhalter können jedoch nicht beliebig an irgendeiner Stelle verwendet werden.

GSD

Dies ist die Kennung, die für Module verwendet wird, die über GSD oder GSDML dem TIA Portal hinzugefügt werden.

Format der Typkennung	Beispiel	Besonderheiten
<GsdName>/<GsdType>	GSD:SIEM8139.GSD/DAP	GsdName ist der Name des GSD oder GSDML in Großbuchstaben.
<GsdName>/<GsdType>/<GsdId>	GSD:SIEM8139.GSD/M/4	<p>GsdType ist eines der folgenden:</p> <ul style="list-style-type: none"> • D: Gerät (device) • R: Baugruppenträger (rack) • DAP: Kopfmodul • M: Modul • SM: Submodul <p>GsdId ist die Typkennung.</p>

System

Dies ist die Kennung für Objekte, die nicht mittels OrderNumber oder GSD bestimmt werden können.

Format der Typkennung	Beispiel	Besonderheiten
<SystemTypeIdentifier>	System:Device.S7300	SystemTypeIdentifier ist die primäre Kennung eines Objekts.
<SystemTypeIdentifier>/<AdditionalTypeIdentifier>	GSD:SIEM8139.GSD/M4	AdditionalTypeIdentifier kann notwendig sein, wenn die SystemTypeIdentifier nicht eindeutig ist. Die Präfixe für bestimmte Objekttypen sind: <ul style="list-style-type: none"> • Connection. • Subnet. • Device. • Rack.

Programmcode

Um die Typkennung für vom Benutzer verwaltbare und getrennt erstellbare Objekte für GSD abzurufen, ändern Sie den folgenden Programmcode:

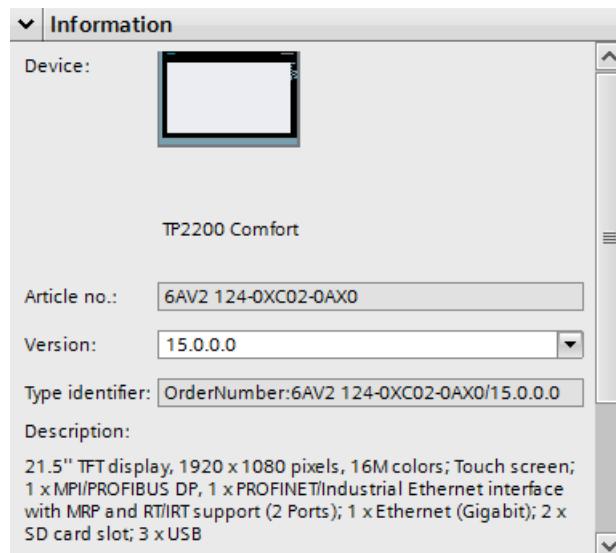
```
HardwareObject hardwareObject = ....;
string typeIdentifier = hardwareObject.TypeIdentifier;
```

Typkennungen im TIA Portal anzeigen

Wenn Sie eine Typkennung kennen müssen, ermitteln Sie sie im TIA Portal wie folgt:

1. Aktivieren Sie unter "Optionen > Einstellungen > Hardware-Konfiguration > Anzeige der Typkennung" die Einstellung "Anzeige der Typkennung für Geräte und Module".
2. Öffnen Sie den Editor "Geräte & Netze".
3. Wählen Sie ein Gerät im Katalog aus.

Die Typkennung wird unter "Information" angezeigt.



7.15.3 App-ID in Gerät und Geräteelementen einstellen

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen

Einleitung

Sie können mit TIA Portal Openness die Anwendungs-ID im Gerät und in den Geräteelementen eines TIA Portal-Projekts einstellen, damit diese Anwendungs-IDs in der aktuellen Sitzung des TIA Portal zur späteren Verwendung gespeichert werden.

Sie stellen in TIA Portal Openness die Anwendungs-ID ein, indem Sie der API das Paar aus "Anwendungsschlüssel" und "Anwendungswert" bereitstellen.

Es gelten folgende Randbedingungen für die Anwendungs-ID aus Geräte- und Geräteelementobjekt:

- Ein Objekt eines Typs, der diese Funktion unterstützt, darf maximal 64 verfügbare Anwendungs-IDs haben.
- Die Schlüssel- und/oder Wertlänge der Anwendungs-ID darf maximal 128 Zeichen umfassen.
- Der Schlüssel der Anwendungs-ID eines bestimmten Objekts ist eindeutig.
- Es darf nur eine Anwendungs-ID mit einem "Anwendungsschlüssel" eingestellt werden.
- Wenn ein anderer Anwendungswert mit dem gleichen "Anwendungsschlüssel" eingestellt wird, wird der zuletzt eingestellte Anwendungswert gespeichert.

Programmcode

Ändern Sie folgenden Programmcode, um die Anwendungs-ID für Geräte- und Geräteelementobjekt einzustellen:

```
Device device = ...;  
// Ask for Service CustomIdentityProvider on the device/device item  
var customIdentityProviderService = device.GetService<CustomIdentityProvider>();  
//Set the Application ID (Key-Value) pair  
customIdentityProviderService.Set("Application_Key", "Application_Value");
```

7.15.4 App-ID in Gerät und Geräteelementen abrufen

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Einleitung

Sie können mit TIA Portal Openness die Anwendungs-ID im Gerät und in den Geräteelementen eines TIA Portal-Projekts abrufen, wenn für das Gerät oder die Geräteelemente bereits eine Anwendungs-ID eingestellt ist.

Anwendungs-IDs werden als Paar aus "Anwendungsschlüssel" und "Anwendungswert" als Bestandteil des TIA Portal gespeichert. Sie können mit TIA Portal Openness den Wert der Anwendungs-ID durch Eingabe des "Anwendungsschlüssels" abrufen.

Programmcode

Ändern Sie folgenden Programmcode, um den Wert der Anwendungs-ID für ein Objekt abzurufen, das bereits mit dem Anwendungsschlüssel eingestellt wurde:

```
Device device = ...;
// Ask for Service CustomIdentityProvider on the device/device item
var customIdentityProviderService = device.GetService<CustomIdentityProvider>();
customIdentityProviderService.Set("Application_Key", "Application_Value");
//Get the Application ID (Value) for the given Application Key
var applicationValue = customIdentityProviderService.Get("Application_Key");
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.15.5 App-ID in Gerät und Geräteelementen entfernen

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe [Verbindung zum TIA Portal herstellen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
Siehe [Projekt öffnen \(Seite 111\)](#)

Prinzip

Sie können mit TIA Portal Openness die Anwendungs-ID (benutzerdefinierte Identität) im Gerät und in den Geräteelementen eines TIA Portal-Projekts entfernen, damit das Gerät und die Geräteelemente mit der relevanten Anwendungs-ID aktualisiert werden können.

Eine `CustomIdentityNotFoundException` wird zurückgegeben, wenn die Anwendungs-ID, die als Argument übergeben wurde, nicht vorhanden ist.

Programmcode

Ändern Sie folgenden Programmcode, um die Anwendungs-ID für Gerät und Geräteelemente zu entfernen:

```
Device device = ...;
// Ask for Service CustomIdentityProvider on the device/device item
var customIdentityProviderService = device.GetService<CustomIdentityProvider>();
//Remove the CustomIdentity corresponding to Application ID
try
{
customIdentityProviderService.Remove("Application_Key");
}
catch(CustomIdentityNotFoundException ex)
{
// When CustomIdentity is not found for a the given key "Application_Key".
}
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.15.6 Erstellen eines Geräts

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Ein Gerät kann auf zwei Arten erstellt werden, in einem Projekt oder in einer Gerätegruppe:

- Erstellen eines Geräts über eine Geräteelement-Typkennung wie im TIA-Hardwarekatalog
Device CreateWithItem(DeviceItemId, DeviceItemName, DeviceName)
- Nur Gerät erstellen
Device Create(DeviceTypeId, DeviceName)

Name	Typ	Beschreibung
DeviceItemTypeId	String	Typkennung des Geräteelements
DeviceTypeId	String	Typkennung des Geräts

7.15 Funktionen auf Geräten

Name	Typ	Beschreibung
DeviceItemName	String	Name des erstellten Geräteelements
DeviceName	String	Name des erstellten Geräts

Siehe: Typkennung (Seite 229)

Programmcode: Gerät mit Typkennung erstellen

Um ein Gerät über eine Typkennung zu erstellen, ändern Sie den folgenden Code:

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

Programmcode: Nur Gerät erstellen

Um nur das Gerät zu erstellen, ändern Sie den folgenden Code:

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

7.15.7 Geräte enumerieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung: Geräte enumerieren

Die TIA Portal Openness API ordnet Geräte ähnlich der Projektnavigation im TIA Portal an. PNV.

- Geräte, die dem Projekt direkt untergeordnet sind, werden in der Zusammensetzung "Devices" des Projekts gesammelt angeordnet.
- Geräte, die sich in Geräteordnern befinden, werden in der Zusammensetzung "Devices" des Ordners gesammelt angeordnet.

Hinweis

AUTOHOTSPOT beachten.

Um die Geräte eines Projekts zu enumerieren, verwenden Sie eine der folgenden Möglichkeiten:

- Alle Geräte auf erster Ebene enumerieren
- Alle Geräte in Gruppen oder Untergruppen enumerieren
- Alle Geräte eines keine Gerätegruppen enthaltenden Projekts enumerieren
- Alle Geräte ungruppiertes Gerätesystemgruppen enumerieren

Beispiele für enumerierbare Geräte:

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

Programmcode: Geräte auf erster Ebene enumerieren

Um Geräte auf erster Ebene zu enumerieren, ändern Sie den folgenden Programmcode:

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

Ändern Sie den folgenden Programmcode, um auf ein einzelnes Gerät zuzugreifen:

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

Programmcode: Geräte in Gruppen oder Untergruppen enumerieren

Um auf Geräte in einer Gruppe zuzugreifen, müssen Sie zuerst zu der Gruppe und danach zu dem Gerät navigieren.

Ändern Sie folgenden Programmcode:

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

Programmcode: Spezifische Geräte finden

Zum Finden eines bestimmten Geräts unter seinem Namen ändern Sie den folgenden Programmcode:

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

Zum Finden eines bestimmten Geräts über die Methode "Find" ändern Sie den folgenden Programmcode:

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

Programmcode: Geräte eines keine Gerätegruppen enthaltenden Projekts enumerieren

Ändern Sie folgenden Programmcode:

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

Programmcode: Alle Geräte in einem Ordner enumerieren

Ändern Sie folgenden Programmcode:

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

Programmcode: Geräte ungruppiertes Gerätesystemgruppen enumerieren

Zum Strukturieren der Projekte wurden dezentrale Geräte in die Gruppe UngroupedDevices eingefügt. Um auf diese Gruppe zuzugreifen, navigieren Sie zuerst zur Gruppe und danach zum Gerät.

Ändern Sie folgenden Programmcode:

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

7.15.8 Auf Geräte zugreifen**Voraussetzung**

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Jedes GSD- oder GSDML-basierte Gerät verfügt über Attribute. Einige davon dienen zur Ermittlung des genauen Typs des Geräts.

Name	Datentyp	Schreibbar	Zugriff	Beschreibung
Author	String	read/write	dynamisch	
Comment	String	read/write	dynamisch	
GsdName	String	read	dynamisch	Name der GSD- oder GSDML-Datei.
GsdType	String	read	dynamisch	Typ des Hardwareobjekts. Der Wert des Geräts ist stets "D".
GsdlId	String	read	dynamisch	Spezifische Kennung des Hardwareobjekts. Für Geräte immer leer.
IsGsd	Bool	read		WAHR für GSD-Gerät oder GSDML-Gerät
Name	String	read/write		
TypeIdentifier	String	read		

Programmcode: Identifikationsattribute abrufen

Um die Attribute abzurufen, ändern Sie folgenden Programmcode:

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

Programmcode: Attribute

Um die Attribute abzurufen, ändern Sie folgenden Programmcode:

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

Programmcode: Attribute bei dynamischem Zugriff

Um die Attribute abzurufen, ändern Sie folgenden Programmcode:

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

Besonderheiten von GSD-Geräten

Ein Gerät, das zugleich ein GSD-Gerät ist, bietet zusätzliche Funktionalität. Um die Funktion `GsdDevice` abzurufen, verwenden Sie die Methode `GetService`.

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

Programmcode: Attribute eines GSD-Geräts

Um die Attribute abzurufen, ändern Sie folgenden Programmcode:

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

7.15.9 Löschen eines Geräts

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode

Ändern Sie den folgenden Programmcode, um ein Gerät zu löschen:

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

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

7.16 Funktionen auf Geräteelementen

7.16.1 Obligatorische Attribute von Geräteelementen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Jedes Gerät oder Geräteelement besitzt bestimmte obligatorische Attribute, die gelesen und/oder geschrieben werden können. Diese Attribute sind immer die gleichen wie in der Benutzeroberfläche des TIA Portals.

In TIA Portal Openness werden die folgenden Attribute unterstützt:

Attributname	Datentyp	Schreib-bar	Zugriff	Kommentar
Author	String	read/write	dynamisch	
Classification	DeviceItemClassification	read		
Comment	String	read/write	dynamisch	manchmal schreibgeschützter Zugriff
CommentML	MultilingualTextItem	read/write	dynamisch	manchmal schreibgeschützter Zugriff
FirmwareVersion	String	read	dynamisch	
InterfaceOperatingMode	InterfaceOperatingModes	read/write	dynamisch	Für Geräteelemente mit der Funktion NetworkInterface
InterfaceType	NetType	read/write	dynamisch	Für Geräteelemente mit der Funktion NetworkInterface

Attributname	Datentyp	Schreibbar	Zugriff	Kommentar
IsBuiltIn	Bool	read		FALSCH für Objekte, die durch den Anwender erstellt werden können
IsGsd	Bool	read		WAHR, wenn die Gerätebeschreibung über GSD/GSDML installiert wird
IsPlugged	Bool	read		WAHR für gesteckte Geräte
Label	String	read	dynamisch	Für Geräteelemente mit der Funktion NetworkPort oder NetworkInterface. Wenn die Schnittstelle oder der Port kein "Label" hat, dann ist Label gleich String.Empty.
LocationIdentifier	String	read/write	dynamisch	
Name	String	read/write		manchmal schreibgeschützter Zugriff
OrderNumber	String	read/write	dynamisch	manchmal schreibgeschützter Zugriff
PlantDesignation	String	read/write	dynamisch	
PositionNumber	int	read		
TypeIdentifier	String	read		
TypeName	String	read	dynamisch	Der sprachenunabhängige Typname. Optional für Geräteelemente, die vom Anwender als automatisch erstellte Elemente oder feste Submodule nicht verwaltbar sind.
InstallationDate	DateTime	read/write	dynamisch	
AdditionalInformation	String	read/write	dynamisch	

Klassifikation des Geräteelements

Wert	Beschreibung
Siemens.Engineering.HW.DeviceItemClassifications.None	Keine Klassifikation.
Siemens.Engineering.HW.DeviceItemClassifications.CPU	Das Geräteelement ist eine CPU
Siemens.Engineering.HW.DeviceItemClassifications.HM	Das Geräteelement ist ein Kopfmodul.

Programmcode: Obligatorische Attribute eines Gerätelements

Ändern Sie den folgenden Programmcode, um die obligatorischen Attribute eines Geräteelements abzurufen:

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

Programmcode: Obligatorische Attribute bei dynamischem Zugriff

Ändern Sie den folgenden Programmcode, um die Attribute bei dynamischem Zugriff aufzurufen:

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

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

7.16.2 Ein Geräteelement erstellen und stecken

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Die Aktion `PlugNew(string typeIdentifier, string name, int positionNumber)` von `HardwareObject` wird verwendet, um

- ein neues Geräteelement zu erstellen und in ein vorhandenes Hardwareobjekt zu stecken
- ein neues Untergeräteelement zu erstellen, z. B. ein Submodul, und dieses in ein Geräteelement zu stecken

Wenn die Aktion erfolgreich war, gibt sie das erstellte Geräteelementobjekt aus. Andernfalls wird eine wiederherstellbare Ausnahme ausgelöst.

Mit der Aktion `CanPlugNew(string typeIdentifier, string name, int positionNumber)` können Sie ermitteln, ob Erstellen und Stecken möglich sind. Wenn eine Ausführung nicht möglich ist, gibt die Aktion `false` aus.

Wenn die Methode den Wert `True` zurückgibt, kann die Aktion aus den folgenden unvorhersehbaren Gründen dennoch fehlschlagen.

- eine Positionsnummer wird bereits von einem anderen Geräteelement verwendet
- das aktuelle Geräteelement kann nicht in die Position gesteckt werden, obwohl sie frei ist
- der Behälter stellt die Positionsnummer nicht bereit
- der Name des Geräteelements wird bereits von einem vorhandenen Geräteelement im selben Behälter verwendet
- das Geräteelement kann nicht in den Behälter gesteckt werden
- das Gerät ist online

Die folgende Tabelle zeigt die benötigten Methodenparameter:

Name	Typ	Beschreibung
<code>typeIdentifier</code>	<code>String</code>	Typkennung des erstellten Geräteelements
<code>name</code>	<code>String</code>	Name des erstellten Geräteelements
<code>positionNumber</code>	<code>int</code>	Positionsnummer des erstellten Geräteelements

Programmcode

Um ein Geräteelement in ein vorhandenes Hardwareobjekt zu stecken, ändern Sie den folgenden Programmcode:

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

Auf Modulinformationen zugreifen

Der TIA Portal Openess-Anwender kann über das Objekt ModuleInformationProvider auf Informationen zu den steckbaren Modulen zugreifen. Der Anwender hat Zugriff auf

- die Behältertypen, in die ein bestimmtes Modul zu stecken ist (z. B. Gerät und Baugruppenträger) – über die Methode FindContainerTypes.
- die verfügbaren Versionen für ein bestimmtes teilweise angegebenes Modul – über die Methode FindModuleTypes.

Programmcode: Auf das Objekt ModuleInformationProvider zugreifen

```
Project project = ...;
HardwareUtilityComposition extensions = project.HwUtilities;
var result = extensions.Find("ModuleInformationProvider") as ModuleInformationProvider;
```

Programmcode: Mit der Methode FindContinerTypes auf Behältertypen zugreifen

Die Methode FindContinerTypes gibt die Behältertypen eines angegebenen Moduls zurück. Das Modul wird über den Parameter typeIdentifier angegeben. Die resultierende Liste enthält die TypIdentifiers sämtlicher Behältertypen des angeforderten Typs. Üblicherweise umfasst dies ein Gerät und einen Baugruppenträger, und die Behälter werden in ihrer Reihenfolge in der Hierarchie im Projekt mit Beginn am Gerät angegebenen.

Name des Parameters	Typ	Beschreibung
typeIdentifier	String	Typkennung eines Gerätelelements

ACHTUNG

Diese Methode funktioniert nur bei die in der Netzsicht sichtbaren Modulen.

Diese Methode funktioniert nur bei Modulen, nicht bei Submodulen.

```
string typeIdentifier = ...;
string[] containerTypes = moduleInformationProvider.FindContainerTypes(typeIdentifier);
```

Programmcode: Mit der Methode FindModuleTypes auf Versionen zugreifen

Die Methode FindModuleTypes gibt alle möglichen Versionen eines Hardwareobjekts unter Verwendung der teilweisen Typkennung des Geräteelements zurück. Diese Methode gibt eine Liste von Strings aus. Jeder String entspricht dem vollständigen TypIdentifier einer möglichen Übereinstimmung für den teilweisen TypIdentifier.

Eine gültige teilweise Typkennung darf nur vollständige Teile enthalten, wobei jeder Teil in der Typkennung durch das Zeichen "/" getrennt ist. Platzhalter oder unvollständige Teile werden nicht unterstützt. Außerdem müssen die folgenden Einschränkungen im Hinblick auf die Mindestanzahl von angegebenen Teilen beachten werden:

- Bestellnummer mindestens ein Teil. Beispiel: OrderNumber: 6ES7 317-2EK14-0AB0
- GSD: mindestens zwei Teile. Beispiel: GSD: SI05816A.GSD/M
- System: mindestens ein Teil. Beispiel: System: Rack.ET200SP

Name des Parameters	Typ	Beschreibung
partialTypeIdentifier	String	teilweise Typkennung eines Geräteelements

```
string partialTypeIdentifier = ...;
string[] moduleTypes = moduleInformationProvider.FindModuleTypes(partialTypeIdentifier);
```

Programmcode: Mit der Methode GetPlugLocations auf die Steckkarte zugreifen

Die Methode GetPlugLocations gibt die Informationen über Steckplätze wie Steckort, Positionsnummer (Bezeichnung eines Steckplatzes) und verfügbare Steckplätze für das Hardwareobjekt zurück.

Die Klasse PlugLocation hat die folgenden Eigenschaften.

Name der Eigenschaft	Typ	Beschreibung
PositionNumber	Int	Die Positionsnummer des freien Steckplatzes
Label	String	Die Bezeichnung des freien Steckplatzes

- Ist für eine bestimmte Positionsnummer kein "label" vorhanden, wird die String-Darstellung der Positionsnummer verwendet.
- PlugLocation-Objekte werden nur für freie Steckplätze bereitgestellt.

```
HardwareObject hardwareObject = ...;
IList<PlugLocation> result = hardwareObject.GetPlugLocations();
foreach (PlugLocation item in result)
{
    Console.WriteLine("{0} - {1}", item.PositionNumber, item.Label);
}
```

7.16.3 Geräteelemente in einen anderen Steckplatz verschieben

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Mit der Aktion PlugMove(DeviceItem deviceItem, int positionNumber) von HardwareObject kann ein vorhandenes Geräteelement verschoben und in ein vorhandenes Hardwareobjekt gesteckt werden. Die Methode PlugMove fügt die Geräteelemente ein, wo das Modul die Bedienoberfläche nicht stecken konnte. In diesen Fällen wird die Aktion PlugMove mit Übersetzungsfehlern beendet.

Die Aktion CanPlugMove(DeviceItem deviceItem, int positionNumber) dient zum Ermitteln der Bewegungsmöglichkeit. Wenn die Bewegung nicht möglich ist, gibt CanPlugMove den Wert False zurück. Wenn die Methode den Wert True zurückgibt, kann die Aktion aus den folgenden unvorhersehbaren Gründen dennoch fehlschlagen.

- eine Positionsnummer wird bereits von einem anderen Geräteelement verwendet
- das aktuelle Geräteelement kann nicht in die Position gesteckt werden, obwohl sie frei ist
- der Behälter stellt die Positionsnummer nicht bereit
- der Name des Geräteelements wird bereits von einem vorhandenen Geräteelement im selben Behälter verwendet
- das Geräteelement kann nicht in den Behälter gesteckt werden
- das Geräteelement kann vom Benutzer nicht gesteckt werden
- das Geräteelement kann vom Benutzer nicht entfernt werden
- das Gerät ist online

Programmcode

Ändern Sie folgenden Programmcode:

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

7.16.4 Geräteelement kopieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Verwenden Sie die Aktion PlugCopy(DevcItem devicItem, int positionNumber) von HardwareObject, um ein Gerät innerhalb eines Projekts zu kopieren und in vorhandene Hardware zu stecken. In seltenen Fällen kann die Methode PlugCopy dann funktionieren, wenn eine Baugruppe nicht in die UI gesteckt werden kann. In diesem Fall treten nach dem Kopieren Übersetzungsfehler auf. Wenn PlugCopy erfolgreich war, wird die Kopie des Geräteelementobjekts zurückgegeben. Andernfalls wird eine wiederherstellbare Ausnahme ausgelöst.

Mögliche Ursachen für eine fehlgeschlagene Aktion:

- eine Positionsnummer wird bereits von einem anderen Geräteelement verwendet
- das aktuelle Geräteelement kann nicht in die Position gesteckt werden, obwohl sie frei ist
- der Behälter stellt die Positionsnummer nicht bereit
- der Name des Geräteelements wird bereits von einem vorhandenen Geräteelement im selben Behälter verwendet
- das Geräteelement kann nicht in den Behälter gesteckt werden
- das Geräteelement kann nicht in die UI gesteckt werden
- ...

Mit der Aktion CanPlugCopy(DevcItem devicItem, int positionNumber) können Sie bestimmen, ob der Kopievorgang möglich ist. Wenn der Kopievorgang nicht ausgeführt werden kann, gibt CanPlugCopy den Wert False zurück. Wenn die Methode jedoch den Wert True zurückgibt, kann die Aktion aus unvorhersehbaren Gründen dennoch fehlschlagen.

Name des Parameters	Typ	Beschreibung
deviceItem	DevcItem	Zu kopierendes Geräteelement
positionNumber	Int	Positionsnummer zum Kopieren des Geräteelements

Programmcode

Ändern Sie folgenden Programmcode:

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

7.16.5 Geräteelement löschen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode

Um ein Geräteelement zu löschen, ändern Sie folgenden Programmcode:

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

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

7.16.6 Geräteelemente enumerieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Um ein Geräteelement abzurufen, verwenden Sie das `HardwareObject`. Die Elemente eines `Hardwareobjekts` sind die in das `Hardwareobjekt` gesteckten Teile, die der Anwender im TIA Portal sehen kann:

- ein Baugruppenträger in einem Gerät
- ein Modul in einem Baugruppenträger
- Ein Submodul in einem Modul
- Ein Submodul in einem Submodul

Hinweis

Weiterführende Informationen zu diesem Thema finden Sie im Kapitel AUTOHOTSPOT.

Programmcode: Geräteelement eines Geräts enumerieren

Ändern Sie den folgenden Programmcode, um Geräteelemente eines Hardware-Objekts zu enumerieren:

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

Programmcode: Mit Zusammensetzungshierarchie enumerieren

Ändern Sie den folgenden Programmcode, wenn Sie die Geräteelemente eines Geräts mit Hilfe der Zusammensetzungshierarchie enumerieren möchten:

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

Programmcode: Geräteelemente mit Zuordnung enumerieren

Ändern Sie den folgenden Programmcode, um die Geräteelemente mit einer Zuordnung zu enumerieren:

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

7.16.7 Geräteelemente aufrufen**Voraussetzung**

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung: Geräteelemente aufrufen

Zum Aufrufen von Objekten vom Typ "DeviceItem" verwenden Sie die folgenden Attribute:

- Name (string): Name des Geräteelements
- Behälter (HardwareObject): Behälter, in den das Geräteelement gesteckt wird

Name	Datentyp	Schreibbar	Zugriff	Beschreibung
Author	String	read/write	dynamisch	
Comment	String	read/write	dynamisch	
FirmwareVersion	String	read	dynamisch	Nur für Kopfmodule
GsdName	String	read	dynamisch	Name der GSD-Datei.
GsdType	String	read	dynamisch	Typ des Hardwareobjekts. Der Wert des Geräts ist stets "D".
GsdlId	String	read	dynamisch	Spezifische Kennung des Hardwareobjekts. Für Geräte immer leer.
IsBuiltIn	Bool	read		
IsGsd	Bool	read		WAHR für GSD-Gerät oder GSDML-Gerät
IsPlugged	Bool	read		
IsProfibus	Bool	read		
IsProfinet	Bool	read		
Name	String	read/write		

Name	Datentyp	Schreibbar	Zugriff	Beschreibung
OrderNumber	String	read	dynamisch	Nur für Kopfmodule
PositionNumber	Bool	read		
TypeIdentifier	String	read		

Programmcode: Geräteelement aufrufen

Ändern Sie zum Aufrufen eines Geräteelements den folgenden Programmcode:

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

Programmcode: Auf Geräteelement von einem Geräteelement zugreifen

Ändern Sie den folgenden Programmcode, um auf ein Geräteelement eines Geräteelements zuzugreifen:

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

Programmcode: Zum Behälter eines Geräteelements navigieren

Ändern Sie den folgenden Programmcode über das Attribut "Container" von DeviceItem, um zurück zum Behälter eines Geräteelements zu navigieren:

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

Programmcode: Identifikationsattribute abrufen

Um die Attribute abzurufen, ändern Sie folgenden Programmcode:

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

Programmcode: Attribute abrufen

Um die Attribute abzurufen, ändern Sie folgenden Programmcode:

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

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

Programmcode: Attribute bei dynamischem Zugriff abrufen

Um die Attribute abzurufen, ändern Sie folgenden Programmcode:

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

var attributeNames = new[] {
    "TypeName", "Author", "Comment", ...
};

foreach (var attributeName in attributeNames) {
    object attributeValue =
((IEngineeringObject)gsdDeviceItem).GetAttribute(attributeName);
}
```

Programmcode: Attribute festlegen

Um die Attribute festzulegen, ändern Sie folgenden Programmcode:

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

Programmcode: prm-Daten eines Kopfmoduls abrufen

Um die prm-Daten abzurufen, ändern Sie folgenden Programmcode:

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

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

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

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

Programmcode: prm-Daten eines Kopfmoduls einstellen

Um die prm-Daten abzurufen, ändern Sie folgenden Programmcode:

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

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

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

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

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

7.16.8 Auf das Geräteelement als Schnittstelle zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Wenn ein Geräteelement eine Schnittstelle ist, bietet es gegenüber einem einfachen Geräteelement zusätzliche Funktionalität. Über diese Schnittstelle kann der Anwender auf die Teilnehmer und die Betriebsart der Schnittstelle zugreifen. Aufgrund dieser Funktionalität kann das Geräteelement durch Zugriff auf die Funktion NetworkInterface als IoDevice (Slave) oder als IoController (Master) verwendet werden (ein spezifischer Dienst des Geräteelements).

Auf die Eigenschaften der Schnittstelle wird über die Enumeration InterfaceOperatingModes zugegriffen.

Wert	Beschreibung
InterfaceOperatingModes.None	Standardeinstellung
InterfaceOperatingModes.IoDevice	Betriebsart der Schnittstelle "IoDevice" (Slave).
InterfaceOperatingModes.IoController	Betriebsart der Schnittstelle "IoController" (Master).
InterfaceOperatingModes.IoDevice or InterfaceOperatingModes.IoController	Schnittstellenbetrieb: beide der oben genannten.

Programmcode: Auf die Netzwerkschnittstellenfunktion zugreifen

Um die Netzwerkschnittstellenfunktion abzurufen, ändern Sie folgenden Programmcode:

```
NetworkInterface itf =
((IEngineeringServiceProvider)deviceItem).GetService<NetworkInterface>();
if (itf != null)
{
    ... // work with the interface
}

//Accessing nodes and operating mode
NodeComposition nodes = itf.Nodes;
InterfaceOperatingModes mode = itf.InterfaceOperatingMode;

//Accessing the type of interface
NetType itfType = itf.InterfaceType;

//Modifying the operating mode and interface type
itf.InterfaceOperatingMode = InterfaceOperatingModes.IoDevice;
itf.InterfaceType = NetType.Profibus

//Accessing the ports linked to an interface.
NetworkPortAssociation nodes = itf.Ports;
```

7.16.9 Auf Attribute einer I/O-Geräteschnittstelle zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Für Schreibzugriff ist der PLC offline.

Verwendung

Mit der Schnittstelle TIA Portal Openness API können Sie Attribute für IRT und isochronen Modus an der I/O-Geräteschnittstelle abrufen oder festlegen.

Zugriff auf die Schnittstelle eines I/O-Controllers

Über die folgenden Auf die folgenden Attribute kann auf die Schnittstelle eines I/O-Controllers zugegriffen werden. Der Controller muss der Synchronisierungsmaster sein:

Attributname	Datentyp	Schreibbar	Zugriff	Beschreibung
PnSendClock	Int64	r/w	Dynamisches Attribut	Takt in Nanosekunden senden

Zugriff auf die Schnittstelle eines I/O-Systems

Über die folgenden Attribute kann auf die Schnittstelle eines I/O-Systems zugegriffen werden. Die Ti/To-Werte können von allen Modulen und Submodulen, die zum I/O-System gehören, verwendet werden.

Attributname	Datentyp	Schreibbar	Zugriff
IsochronousTiToAutoCalculation	BOOL	r/w	Dynamisches Attribut
IsochronousTi	DOUBLE	r/w	Dynamisches Attribut
IsochronousTo	DOUBLE	r/w	Dynamisches Attribut

Zugriff auf die Schnittstelle eines I/O-Geräts

Über die folgenden Attribute kann auf die Schnittstelle eines I/O-Geräts zugegriffen werden. Die Ti/To-Werte können von allen Modulen und Submodulen, die zum I/O-System gehören, verwendet werden.

Attributname	Datentyp	Schreibbar	Zugriff
IsochronousMode	BOOL	r/w	Dynamisches Attribut
IsochronousTiToCalculationMode	IsochronousTiToCalculationMode	r/w	Dynamisches Attribut
IsochronousTi	DOUBLE	r/w	Dynamisches Attribut
IsochronousTo	DOUBLE	r/w	Dynamisches Attribut

Die folgenden ENUM-Werte werden für das Attribut IsochronousTiToCalculationMode bereitgestellt:

Wert	Beschreibung
IsochronousTiToCalculationMode.None	
IsochronousTiToCalculationMode.FromOB	Ti/To-Werte des OB (am IO-System konfiguriert) werden verwendet.
IsochronousTiToCalculationMode.FromSubnet	Dieser Wert wird von PROFINET-Schnittstellen nicht verwendet.
IsochronousTiToCalculationMode.AutomaticMinimum	Ti/To-Werte werden für das IO-Device automatisch berechnet.
IsochronousTiToCalculationMode.Manual	Der Anwender kann für dieses IO-Device Ti/To-Werte manuell eingeben.

Programmcode: Attribute einer I/O-Geräteschnittstelle abrufen oder festlegen

Ändern Sie den folgenden Programmcode, um auf den Sendetaktwert zuzugreifen:

```
DeviceItem pnInterface = ...;
// read attribute
long attributeValue = (long)pnInterface.GetAttribute("PnSendClock");
// write attribute
long sendClock = 2000000;
pnInterface.SetAttribute("PnSendClock", sendClock);
```

Ändern Sie den folgenden Programmcode, um auf die Ti/To-Werte eines OB zuzugreifen:

```
IoSystem ioSystem = ...;
bool titoAutoCalculation = (bool)ioSystem.GetAttribute("IsochronousTiToAutoCalculation");
ioSystem.SetAttribute("IsochronousTiToAutoCalculation", true);
```

Ändern Sie den folgenden Programmcode, um auf die isochrone Einstellung einer I/O-Geräteschnittstelle zuzugreifen:

```
DeviceItem pnInterface = ...;
bool isochronousMode = (bool)pnInterface.GetAttribute("IsochronousMode");
pnInterface.SetAttribute("IsochronousMode", true);
```

7.16.10 Auf Attribute des IoController zugreifen

Voraussetzungen

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Für Schreibzugriff ist der PLC offline.

Anwendung

Mit der Schnittstelle TIA Portal Openness API können Sie Attribute für den IoController abrufen oder festlegen. Die folgenden Attribute sind nur unter PROFINET IoController (unter einer PROFINET-Schnittstelle) verfügbar. Wenn der Anwender ein Attribut in der Benutzeroberfläche ändern kann, kann der Anwender das entsprechende Attribut auch über TIA Portal Openness ändern.

Attributname	Datentyp	Typ	Zugriff	Beschreibung
SyncRole	SyncRole	Lesen-Schreiben	Dynamisches Attribut	
PnDeviceNumber	Int	Schreibgeschützt	Dynamisches Attribut	In der TIA Portal UI befindet sich diese Eigenschaft unter dem Knoten Ethernet (Abschnitt PROFINET)

Die Eigenschaft Synchronization role ist in der PROFINET-Schnittstelle der TIA Portal UI verfügbar. Die Enum SyncRole hat die folgenden Werte:

Enumerationswert	Numerischer Wert
SyncRole.NotSynchronized	0
SyncRole.SyncMaster	1
SyncRole.SyncSlave	2
SyncRole.RedundantSyncMaster	4

Programmcode: Attribute des IoController festlegen

```
IoController ioController= ...;
SyncRole syncRole = (SyncRole)((IEngineeringObject)ioController).GetAttribute("SyncRole");
((IEngineeringObject)ioController).SetAttribute("SyncRole", SyncRole.SyncMaster);
```

7.16.11 Auf Attribute des IoConnector zugreifen

Voraussetzungen

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Für Schreibzugriff ist der PLC offline.

Anwendung

Mit der Schnittstelle TIA Portal Openness API können Sie Attribute für den IoConnector abrufen oder festlegen. Die folgenden Attribute sind nur unter PROFINET IoController (unter einer PROFINET-Schnittstelle) verfügbar. Wenn der Anwender ein Attribut in der Benutzeroberfläche ändern kann, kann der Anwender das entsprechende Attribut auch über TIA Portal Openness ändern.

Es gibt vier Typen von Attributen wie Attribute für die Aktualisierungszeit, Attribute für die Überwachungszeit, Attribute für die Synchronisation und Attribute für die Gerätenummer.

Attribute für die Aktualisierungszeit

Die Attribute für die Aktualisierungszeit werden nachstehend aufgeführt.

Attributname	Datentyp	Typ	Zugriff	Beschreibung
PnUpdateTimeAuto-Calculation	Bool	Lesen-Schreiben	Dynamisches Attribut	Wenn dieses Attribut wahr ist, wird die Aktualisierungszeit automatisch berechnet.
PnUpdateTime	Int64	Lesen-Schreiben	Dynamisches Attribut	Die Aktualisierungszeit wird in Nanosekunden gemessen.
PnUpdateTimeAdapti-on	Bool	Lesen-Schreiben	Dynamisches Attribut	

Attribute für die Überwachungszeit

Die Attribute für die Überwachungszeit werden nachstehend aufgeführt.

Attributname	Datentyp	Typ	Zugriff	Beschreibung
PnWatchdogFactor	Int32	Lesen-Schreiben	Dynamisches Attribut	
PnWatchdogTime	Int64	Schreibgeschützt	Dynamisches Attribut	Die Überwachungszeit wird in Nanosekunden gemessen.

Attribute für die Synchronisation

Die Attribute für die Synchronisation werden nachstehend aufgeführt.

Attributname	Datentyp	Typ	Zugriff	Beschreibung
RtClass	RtClass	Lesen-Schreiben	Dynamisches Attribut	
SyncRole	SyncRole	Schreibgeschützt	Dynamisches Attribut	

Die Enum RtClass hat folgende Werte.

Enumerationswert	Numerischer Wert
RtClass.None	0
RtClass.RT	1
RtClass.IRT	2

Die Enum SyncRole hat folgende Werte.

Enumerationswert	Numerischer Wert
SyncRole.NotSynchronized	0
SyncRole.SyncMaster	1
SyncRole.SyncSlave	2
SyncRole.RedundantSyncMaster	4

Attribute für die Gerätenummer

Die Attribute für die Gerätenummer werden nachstehend aufgeführt.

Attributname	Datentyp	Typ	Zugriff	Beschreibung
PnDeviceNumber	Int	Lesen-Schreiben	Dynamisches Attribut	Gibt die Gerätenummer an.

Programmcode: Attribute des IoConnector abrufen und festlegen

```
IoConnector connector = ...  
  
var attributeNames = new[] {  
    "PnUpdateTimeAutoCalculation", "PnUpdateTime", "PnUpdateTimeAdaption", "PnWatchdogFactor",  
    "PnWatchdogTime", "RtClass", "SyncRole"  
};  
  
foreach (var attributeName in attributeNames)  
{  
    object attributeValue = ((IEngineeringObject)connector).GetAttribute(attributeName);  
}  
  
connector.SetAttribute("PnUpdateTimeAutoCalculation", true);
```

Siehe auch

[Projekt öffnen \(Seite 111\)](#)

7.16.12 Auf einen Adresscontroller zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Ein Geräteelement, das zugleich ein Adresscontroller ist, bietet zusätzliche Funktionalität. Um auf die registrierten Adressen des Adresscontrollers zuzugreifen, muss die Rolle AddressController verwendet werden.

Programmcode: Adresscontroller abrufen

Um die Adresscontrollerrolle abzurufen, ändern Sie folgenden Programmcode:

```
AddressController addressController =
((IEngineeringServiceProvider)deviceItem).GetService<AddressController>();
if (addressController != null)
{
    ... // work with the address controller
}
```

Attribute eines Adresscontrollers

Ein Adresscontroller hat folgende Attribute:

- RegisteredAddresses

Um die Attribute eines Adresscontrollers abzurufen, ändern Sie folgenden Programmcode:

```
AddressController addressController = ...;
foreach (Address registeredAddress in addressController.RegisteredAddresses)
{
    ...
}
```

7.16.13 Auf Adressen zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Adressobjekte werden über den Zusammensetzungslink `Addresses` eines Geräteelements aufgerufen. Das Attribut `Addresses` gibt eine Sammlung von `AddressComposition` zurück, die enumeriert werden kann.

Programmcode: Adresse eines Geräteelements abrufen

Ändern Sie den folgenden Programmcode, um die Adresse eines Geräteelements abzurufen:

```
AddressComposition addresses = deviceItem.Addresses;
foreach(Address address in addresses)
{
    // work with the address
}
```

Programmcode: Adresse eines IO-Controllers abrufen

Ändern Sie den folgenden Programmcode, um die Adresse eines IO-Controllers abzurufen:

```
AddressComposition addresses = ioController.Addresses;
foreach(Address address in addresses)
{
    // work with the address
}
```

Attribute

Adresse unterstützt die folgenden Attribute:

Attributname	Datentyp	Schreibbar	Zugriff	Kommentar
AddressControllers	AddressControllerAssociation	read		
Context	enum: AddressContext	read	dynamisch	nur für Diagnoseadressen und für spezifische Geräteelemente
IoType	enum: AddressIoType	read		
StartAdress	Int32	read/write		
Length	Int32	read		

Wert	Beschreibung
AddressIoType.Diagnosis	Der Adress-IO-Typ ist Diagnose.
AddressIoType.Input	Der Adress-IO-Typ ist Eingang.
AddressIoType.Output	Der Adress-IO-Typ ist Ausgang.
AddressIoType.Substitute	Der Adress-IO-Typ ist Ersatz.
AddressIoType.None	Der Adress-IO-Typ ist nicht angegeben.

Wert	Beschreibung
AddressContext.None	Der Adresskontext ist nicht gültig.
AddressContext.Device	Ein Geräte-Adressenkontext
AddressContext.Head	Ein Kopf-Adressenkontext

Programmcode: Attribute lesen

Um die Attribute abzurufen, ändern Sie folgenden Programmcode:

```
AddressControllerAssociation addressControllers = address.AddressControllers;  
Int32 startAddress = address.StartAddress;  
AddressIoType addressType = address.IoType;  
Int32 addressLength = address.Length;
```

Programmcode: Attribute schreiben

Um die Attribute zu schreiben, ändern Sie folgenden Programmcode:

```
Address addressControllers = ...;  
  
address.StartAddress = intValueStartAddress;
```

Programmcode: Attribute bei dynamischem Zugriff

Um die Attribute abzurufen, ändern Sie folgenden Programmcode:

```
Address address= ...;  
  
object attributeValue = ((IEngineeringObject)address).GetAttribute("Context");
```

7.16.14 Auf "Hardwarekennung" zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Hardwarekennungsobjekte werden von den folgenden Objekten abgerufen:

- Device
- DeviceItem
- IoSystem

Die Hardwarekennung wird durch die Klasse `HwIdentifier` dargestellt und über das Attribut `HwIdentifiers` abgerufen.

Programmcode: Hardwarekennung abrufen

Um HwIdentifier verfügbar zu machen, ändern Sie folgenden Programmcode:

```
var hwObject = ...
foreach(HwIdentifier hardwareIdentifier in hwObject.HwIdentifiers)
{
    // Work with the HwIdentifier
}
```

Attribute einer Hardwarekennung

```
HwIdentifierControllerAssociation controllers = hwIdentifier.HwIdentifierControllers;
Int64 Identifier = hwIdentifier.Identifier;
```

7.16.15 Auf Hardwarekennungscontroller zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Wenn ein Geräteelement gleichzeitig ein Hardwarekennungscontroller ist, kann auf die registrierten Hardwarekennungen zugegriffen werden. Um auf diese HwIdentifierController zuzugreifen, muss ein bestimmter Dienst des Geräteelements verwendet werden.

Programmcode: Hardwarekennungscontroller abrufen

Um den HwIdentifierController abzurufen, ändern Sie folgenden Programmcode:

```
HwIdentifierController hwIdentifierController =
((IEngineeringServiceProvider)deviceItem).GetService<HwIdentifierController>();
if (hwIdentifierController != null)
{
    ... // work with the hardware identifier controller
}
```

Programmcode: Attribute eines Hardwarekennungscontrollers

Ein Adresscontroller hat folgende Attribute:

- RegisteredHwIdentifiers: Die Hardwarekennungscontroller, auf denen die Hardwarekennung registriert ist.

Um die Attribute eines Adresscontrollers abzurufen, ändern Sie folgenden Programmcode:

```
HwIdentifierController hwIdentifierController = ...;
HwIdentifierAssociation controllers = hwIdentifierController.RegisteredHwIdentifiers;
```

7.16.16 Auf Kanäle von Geräteelementen zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Ein Kanal wird durch die Klasse Channel dargestellt. Kanäle werden von einem Geräteelement über das Attribut Channels der Klasse DeviceItem abgerufen. Das Attribut Channels gibt eine Implementierung von ChannelComposition zurück, die enumeriert werden kann. Wenn das Geräteelement keine Kanäle hat, gibt das Attribut Channels eine leere Sammlung zurück.

Obligatorische Attribute

Ein Kanal unterstützt die folgenden obligatorischen Attribute:

Attributname	Datentyp	Schreibbar	Zugriff	Kommentar
IoType	ChannelIoType	read		
Type	ChannelType	read		
Number	Int32	read		
ChannelAddress	Int32	read	dynamisch	Adresse des Kanals in Bits
ChannelWidth	UInt32	read	dynamisch	Breite des Kanals in Bits

Programmcode: Kanäle von einem Geräteelement abrufen

Ändern Sie den folgenden Programmcode, um die Kanäle eines Geräteelements abzurufen:

```
ChannelComposition channels = deviceItem.Channels
foreach(Channel channel in channels)
{
    // work with the channel
}
```

Programmcode: Obligatorische Attribute eines Kanals

Ändern Sie den folgenden Programmcode, um die Kanäle eines Geräteelements abzurufen:

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

Programmcode: Werte der Attribute bei dynamischem Zugriff abrufen

Um die Werte von dynamischen Attributen abzurufen, ändern Sie folgenden Programmcode:

```
Channel channel = ...;
Int32 channelAddress = (Int32)((IEngineeringObject)channel).GetAttribute("ChannelAddress");
UInt32 channelWidth = (UInt32)((IEngineeringObject)channel).GetAttribute("ChannelWidth");
```

Programmcode: Wert eines dynamischen Attributs festlegen

Um den Wert eines schreibbaren dynamischen Attributs festzulegen, ändern Sie folgenden Programmcode:

```
Channel channel = ...;
((IEngineeringObject)channel).SetAttribute("AnAttribute", 1234);
```

7.16.17 PSC-Datei erstellen und exportieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Projekt öffnen
Siehe Projekt öffnen (Seite 111)

Anwendung

Über die TIA Portal Openness API können Sie eine PSC-Datei erstellen und ein Gerät in die Datei laden. Um eine PSC-Datei zu erstellen und eine Gerätekonfiguration in die Datei zu laden, können Sie die Methode Export eines Dienstes CardReaderPscProvider verwenden. Der Dienst ist im Namensraum Siemens.Engineering.HW.Utilities vorhanden.

Programmcode

Ändern Sie folgenden Programmcode, um eine PSC-Datei zu erstellen und zu exportieren:

```
// preconditions
Device ipcDevice = myProject.Devices.Find("IPC427D");
FileInfo exportFile = new FileInfo(@"C:\Users\Ertan\Documents\Automation\PC system
configuration74.psc");
// get the card reader provider from hardware utilities
HardwareUtilityComposition utilities = myProject.HwUtilities;
HardwareUtility utility = utilities.Find("CardReaderPscProvider");
CardReaderPscProvider crp = (CardReaderPscProvider)utility;
// do the export
crp.Export(ipcDevice, exportFile);
```

Das Erstellen und Öffnen einer PSC-Datei mit separaten Befehlen wird nicht unterstützt. Wenn Sie eine bereits vorhandene PSC-Datei als Parameter angeben, wird die Datei nicht überschrieben und eine Ausnahme wird zurückgegeben. Ist die PSC-Datei noch nicht vorhanden, wird sie erstellt und der Ladevorgang durchgeführt.

Sie müssen sicherstellen, dass das Projekt problemlos übersetzt werden kann. Ansonsten wird eine Ausnahme zurückgegeben.

Hinweis

Aus Sicherheitsgründen wird der Exportvorgang für F-aktivierte Geräte in V15.1 nicht unterstützt (beim Export wird eine Ausnahme zurückgegeben).

7.16.18 Verwalten von Verbindungen bei Erweiterungsbaugruppenträgern

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Sie können mithilfe von TIA Portal Openess Verbindungen für Erweiterungsbaugruppenträger abrufen, hinzufügen und entfernen. Dadurch können Sie auch die Unterstützung von TIA Portal Openess für die Realisierung von Verbindungen für Erweiterungsbaugruppenträger beim CAx-Export/Import nutzen.

Programmcode

```
ImConnection imConnection = portDeviceItem.GetService<ImConnection>();  
imConnection.Connect(partnerport);  
imConnection.Disconnect();  
imConnection.GetPartnerPort();  
var imConnectionOwner = imConnection.OwnedBy;
```

Siehe auch

- Verbindung zum TIA Portal aufbauen (Seite 79)
- Projekt öffnen (Seite 111)

7.17 Funktionen auf Daten eines HMI-Gerätes

7.17.1 Bilder

7.17.1.1 Benutzerdefinierte Bilderordner erzeugen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode

Um einen benutzerdefinierten Bilderordner zu erstellen, ändern Sie folgenden Programmcode:

```
//Creates a screen folder
private static void CreateScreenFolder(HmiTarget hmitarget)
{
    ScreenUserFolder myCreatedFolder =
    hmitarget.ScreenFolder.Folders.Create("myScreenFolder");
}
```

7.17.1.2 Bild aus einem Ordner löschen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Hinweis

Sie können ein Permanentfenster nicht löschen. Ein Permanentfenster ist ein Systembild, das immer vorhanden ist.

Programmcode

Um ein Bild aus einem bestimmten Ordner zu löschen, ändern Sie folgenden Programmcode:

```
public static void DeleteScreenFromFolder(HmiTarget hmiTarget)
{
    ScreenUserFolder screenUserFolder =
    hmiTarget.ScreenFolder.Folders.Find("myScreenFolder");
    ScreenComposition screens = screenUserFolder.Screens;
    Screen screen = screens.Find("myScreenName");
    if (screen != null)
    {
        screen.Delete();
    }
}
```

7.17.1.3 Bildvorlage aus einem Ordner löschen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Im Projekt ist ein HMI-Gerät vorhanden.

Programmcode

Um eine Bildvorlage aus einem bestimmten Ordner zu löschen, ändern Sie folgenden Programmcode:

```
private static void DeleteScreenTemplateFromFolder(HmiTarget hmiTarget)
{
    string templateName = "MyScreenTemplate";
    ScreenTemplateUserFolder folder =
hmiTarget.ScreenTemplateFolder.Folders.Find("myScreenTemplateFolder");
    ScreenTemplateComposition templates = folder.ScreenTemplates;
    ScreenTemplate template = templates.Find(templateName);
    if (template != null)
    {
        template.Delete();
    }
}
```

7.17.1.4 Alle Bilder aus einem Ordner löschen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Hinweis

Sie können ein Permanentfenster nicht löschen. Ein Permanentfenster ist ein Systembild, das immer vorhanden ist.

Programmcode

Um alle Bilder aus einem bestimmten Ordner zu löschen, ändern Sie folgenden Programmcode:

```
private static void DeleteAllScreensFromFolder(HmiTarget hmitarget)
//Deletes all screens from a user folder or a system folder
{
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Find("myScreenFolder");
    //or ScreenSystemFolder folder = hmitarget.ScreenFolder;
    ScreenComposition screens = folder.Screens;
    List<Screen> list = new List<Screen>();
    foreach(Screen screen in screens)
    {
        list.Add(screen);
    }
    foreach (Screen screen in list)
    {
        screen.Delete();
    }
}
```

7.17.2 Zyklen

7.17.2.1 Zyklus löschen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Im Projekt ist ein HMI-Gerät vorhanden.

Verwendung

Standardzyklen können Sie nicht löschen.

Sie können anhand der Zusammensetzung im Objektmodell (composition count) des jeweiligen Zyklus ermitteln, ob tatsächlich Zyklen gelöscht wurden. Auf diese Zyklen ist kein Zugriff mehr möglich.

Programmcode

Um einen Zyklus aus einem HMI-Gerät zu löschen, ändern Sie folgenden Programmcode:

```
public static void DeleteCycle(HmiTarget hmiTarget)
{
    CycleComposition cycles = hmiTarget.Cycles;
    Cycle cycle = cycles.Find("myCycle");
    cycle.Delete();
}
```

7.17.3 Textlisten

7.17.3.1 Textliste löschen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Im Projekt ist ein HMI-Gerät vorhanden.

Programmcode

Um eine ausgewählte Textliste und alle zugehörigen Listeneinträge aus einem HMI-Gerät zu löschen, ändern Sie folgenden Programmcode:

```
public static void DeleteTextList(HmiTarget hmiTarget)
{
    TextListComposition textLists = hmiTarget.TextLists;
    TextList textList = textLists.Find("myTextList");
    textList.Delete();
}
```

7.17.4 Grafiklisten

7.17.4.1 Grafikliste löschen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Im Projekt ist ein HMI-Gerät vorhanden.

Programmcode

Um eine ausgewählte Grafikliste und alle zugehörigen Listeneinträge aus einem HMI-Gerät zu löschen, ändern Sie folgenden Programmcode:

```
private static void DeleteGraphicList(HmiTarget hmiTarget)
{
    GraphicListComposition graphicLists = hmiTarget.GraphicLists;
    GraphicList graphicList = graphicLists.Find("myGraphicList");
    graphicList.Delete();
}
```

7.17.5 Verbindungen

7.17.5.1 Verbindung löschen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Im Projekt ist ein HMI-Gerät vorhanden.

Programmcode

Um eine ausgewählte Kommunikationsverbindung aus einem HMI-Gerät zu löschen, ändern Sie folgenden Programmcode:

```
private static void DeleteConnection(HmiTarget hmiTarget)
{
    ConnectionComposition connections = hmiTarget.Connections;
    Connection connection = connections.Find("HMI_connection_1");
    connection.Delete();
}
```

7.17.6 Variablenliste

7.17.6.1 Benutzerdefinierte Ordner für HMI-Variablen erzeugen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode

Um einen benutzerdefinierten Ordner für HMI-Variablen zu erstellen, ändern Sie folgenden Programmcode:

```
private static void CreateUserfolderForHMITags(HmiTarget hmitarget)
// Creates an HMI tag user folder
{
    TagSystemFolder folder = hmitarget.TagFolder;
    TagUserFolder myCreatedFolder = folder.Folders.Create("MySubFolder");
}
```

7.17.6.2 Variablen einer HMI-Variablenliste enumerieren

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode

Um alle Variablen einer HMI-Variablenliste zu enumerieren, ändern Sie folgenden Programmcode:

```
private static void EnumerateTagsInTagtable(HmiTarget hmitarget)
// //Enumerates all tags of a tag table
{
    TagTable table = hmitarget.TagFolder.TagTables.Find("MyTagtable");
    // Alternatively, you can access the default tag table:
    // TagTable defaulttable = hmitarget.TagFolder.DefaultTagTable;

    TagComposition tagComposition = table.Tags;
    foreach (Tag tag in tagComposition)
    {
        // Add your code here
    }
}
```

7.17.6.3 Einzelne Variable aus einer HMI-Variablenliste löschen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode

Um eine bestimmte Variable aus einer HMI-Variablenliste zu löschen, ändern Sie folgenden Programmcode:

```
private static void DeleteATag(HmiTarget hmiTarget)
{
    string tagName = "MyTag";
    TagTable defaultTagTable = hmiTarget.TagFolder.DefaultTagTable;
    TagComposition Variablen = defaultTagTable.Tags;
    Tag tag = tags.Find(tagName);
    tag.Delete();
}
```

7.17.6.4 Variablenliste aus einem Ordner löschen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Im Projekt ist ein HMI-Gerät vorhanden.

Verwendung

Sie können die Standardvariablenliste nicht löschen

Programmcode

Ändern Sie folgenden Programmcode:

```
// Delete a tag table from a specific folder
private static void DeleteTagTable(HmiTarget hmiTarget)
{
    string tableName = "myTagTable";
    TagSystemFolder tagSystemFolder = hmiTarget.TagFolder;
    TagTableComposition tagTables = tagSystemFolder.TagTables;
    TagTable tagTable = tagTables.Find(tableName);
    tagTable.Delete();
}
```

7.17.7 VB-Skripte

7.17.7.1 Benutzerdefinierte Ordner für Skripte erzeugen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode

Um einen benutzerdefinierten Unterordner für Skripte in einem Systemordner oder einem anderen benutzerdefinierten Ordner zu erstellen, ändern Sie folgenden Programmcode:

```
private static void CreateFolderInScriptfolder(HmiTarget hmitarget)
//Creates a script user subfolderVBScriptSystemFolder
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptUserFolderComposition vbScriptFolders = vbScriptFolder.Folders;
    VBScriptUserFolder vbScriptSubFolder = vbScriptFolder.Folders.Create("mySubfolder");
}
```

7.17.7.2 VB-Skript aus einem Ordner löschen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Im Projekt ist ein HMI-Gerät vorhanden.

Programmcode

Um ein VB-Skript aus einem bestimmten Ordner zu löschen, ändern Sie folgenden Programmcode:

```
//Deletes a vbscript from a script folderVBScriptSystemFolder
private static void DeleteVBScriptFromScriptFolder(HmiTarget hmitarget)
{
    VBScriptUserFolder vbscriptfolder =
    hmitarget.VBScriptFolder.Folders.Find("MyScriptFolder");
    var vbScripts = vbscriptfolder.VBScripts;
    if (null != vbScripts)
    {
        var vbScript = vbScripts.Find("MyScript");
        vbScript.Delete();
    }
}
```

7.17.8 Benutzerdefinierten Ordner eines Bediengeräts löschen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode

Ändern Sie den folgenden Programmcode, um einen benutzerdefinierten Ordner eines Bediengeräts zu löschen:

```
HmiTarget hmiTarget = ...;
ScreenUserFolder screenUserGroup = hmiTarget.ScreenFolder.Folders.Find("MyUserFolder");
screenUserGroup.Delete();
```

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

7.18.1 Objektliste

Einleitung

In den folgenden Tabellen werden die verfügbaren Objekte aufgeführt und es wird angegeben, ob diese Objekte von TIA Portal Openness unterstützt werden.

Objekte

Für WinCC Unified-Geräte können Sie die folgenden Projektdaten steuern:

Objekt	WinCC Scada Runtime Unified
Analogmeldungen	Ja
Bitmeldungen	Ja
Meldeklassen	Ja
Datenarchive	Ja
Meldearchive	Ja
Archivvariablen	Ja
Variable	Ja
Verbindungen	Ja
Laufzeiteinstellungen von Geräten	Ja

Objekt	WinCC Scada Runtime Unified
Bilder und Bildobjekte	Ja
Anlagensicht	Ja
Anlagenobjekt	Ja
Anlagensichtknoten	Ja
Schnittstelle Anlagenobjekt	Ja
Mitglied Schnittstelle Anlagenobjekt	Ja
Archivvariable Anlagenobjekt	Ja

7.18.2 HMI-Unified-Software-Objekt

Einleitung

Es ist erforderlich, dass ein HMI-Softwareobjekt auch auf andere Elemente wie Variablen, Verbindungen, Meldungen und Bilder zugreifen kann.

Voraussetzungen

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Öffnen eines Projekts (Seite 111)

Programmcode

Ändern Sie folgenden Programmcode so, dass der Zugriff auf ein HMI-Softwareobjekt möglich wird:

```
private HmiSoftware GetHmiSoftware()
{
    HmiSoftware hmiSoftware = null;
    Project project = null;
    IList<TiaPortalProcess> tiaProcessList = TiaPortal.GetProcesses();
    //If no TIA Application instance is running, then following using statement will start it or
    //if already running then will attach to it.
    TiaPortal tiaApp = tiaProcessList.Count > 0? tiaProcessList[0].Attach(): new
    TiaPortal(TiaPortalMode.WithUserInterface);
    //If there is device projects are present in given TIA instance, then take first one to work
    upon.
    if (tiaApp.Projects.Count > 0)
        project = tiaApp.Projects[0];
    else
    {
        //if there is no device project in given TIA instance, open the existing project.
        FileInfo file = new FileInfo(@"d:\Automation\Project1\Project1.apx");
        if (!file.Exists)
            throw new FileNotFoundException("Project1.apx not found");
        project = tiaApp.Projects?.Open(file);
    }
    //After getting project, get the object representing UA software HMI device.
    var devices = project.Devices;
    if (devices != null)
    {
        var device = devices[0];
        var deviceItems = device.DeviceItems;
        if (deviceItems != null)
        {
            foreach (DeviceItem deviceItem in deviceItems)
            {
                SoftwareContainer softwareContainer = deviceItem.GetService<SoftwareContainer>();
                hmiSoftware = softwareContainer?.Software as HmiSoftware;
            }
        }
    }
}
return hmiSoftware;
}
```

Hinweis

Im obigen Programmcode bezieht sich die Erweiterung .apx auf die installierte Version des TIA Portals.

7.18.3 Fehler abfragen

Einleitung

Mit TIA Portal Openness können Sie Fehlerinformationen über Eigenschaften für folgende WinCC-Objekte abfragen:

- Analogmeldung
- Bitmeldung
- Meldeklasse
- Datenarchiv
- Meldearchiv
- Variable
- Verbindung
- Archivvariable
- Laufzeiteinstellungen von Geräten
- Bilder und Bildobjekte
- Anlagensicht
- Anlagenobjekt
- Anlagensichtknoten
- Schnittstelle Anlagenobjekt
- Mitglied Schnittstelle Anlagenobjekt
- Archivvariable Anlagenobjekt

Hinweis

Die Funktion ist nicht auf Systemvariablen und Variablenklassen anwendbar, denn diese haben keine Eigenschaft, die im Fehlerzustand sein kann.

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Zugriff auf HMI-Unified-Software-Objekt
Siehe HMI-Unified-Software-Objekt (Seite 281)

Programmcode

Ändern Sie folgenden Programmcode, um Fehler zu Variablen, Meldungen, Laufzeiteinstellungen und Bildern abzufragen:

```
private void ValidateMultipleHmiObjects()
{
    HmiSoftware hmiSoftware = GetHmiSoftware();
    IList<IValidator> objectsToValidate = new List<IValidator>();
    //Put tags in validation list.
    HmiTagComposition hmiTags = hmiSoftware.HmiTags;
    foreach (HmiTag hmiTag in hmiTags)
    {
        objectsToValidate.Add(hmiTag);
    }
    //Put alarms in validation list.
    DiscreteAlarmComposition discreteAlarms = hmiSoftware.DiscreteAlarms;
    foreach (DiscreteAlarm discreteAlarm in discreteAlarms)
    {
        objectsToValidate.Add(discreteAlarm);
    }
    //Put RuntimeSettings in validation list.
    objectsToValidate.Add(hmiSoftware.RuntimeSettings);
    foreach (IValidator validator in objectsToValidate)
    {
        IList<HmiValidationResult> validationResult = validator.Validate();
        if (validationResult != null && validationResult.Count > 0)
        {
            foreach (HmiValidationResult propertyErrors in validationResult)
            {
                //browse error for different property
                string propertyName = propertyErrors.PropertyName;
                foreach (string propertyError in propertyErrors.Errors)
                {
                    //work with property errors
                }
            }
        }
    }
    //Put screens in validation list.
    foreach (var screen in hmiSoftware.Screens)
    {
        objectsToValidate.Add(screen);
    }
    foreach (IValidator validator in objectsToValidate)
    {
        IList<HmiValidationResult> errors = validator.Validate();
        if (errors != null && errors.Count > 0)
        {
            foreach (var errornotification in errors)
            {
                //browse error for different property
                var propName = errornotification.PropertyName;
                foreach (var errormessage in errornotification.Errors)
                {
                    Console.WriteLine(errormessage);
                }
            }
        }
    }
}
```

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

```
//Put screenitems in validation list.  
foreach (var screenitem in ScreenObject.ScreenItems)  
{  
    objectsToValidate.Add(screenitem);  
}  
foreach (IValidator validator in objectsToValidate)  
{  
    IList<HmiValidationResult> errors = validator.Validate();  
    if (errors != null && errors.Count > 0)  
    {  
        foreach (var errornotification in errors)  
        {  
            //browse error for different property  
            var propName = errornotification.PropertyName;  
            foreach (var errormessage in errornotification.Errors)  
            {  
                Console.WriteLine(errormessage);  
            }  
        }  
    }  
}
```

Siehe auch

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

7.18.4 Meldungen

7.18.4.1 Mit Analogmeldungen arbeiten

Einleitung

Mit Analogmeldungen können Sie folgende Aktionen durchführen, während Sie TIA Portal Openness verwenden:

- Analogmeldungen erstellen
- Analogmeldungen enumerieren
- Analogmeldungen löschen
- Auf Eigenschaften von Analogmeldungen zugreifen

Eigenschaften

Die folgende Eigenschaften werden in Analogmeldungen unterstützt:

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
EventText	MultilingualText	Gibt Ereignis/Melde-text der Meldung an	R/W
Id	uint32	Gibt Meldenummer zum Identifizieren der Meldung an. Hierbei handelt es sich um einen eindeutigen Wert für jede Analogmel-dung	R/W
AlarmClass	String	Gibt Meldeklaasse an	R/W
Name	String	Gibt den Meldenamen an	R/W
Priority	byte	Gibt Meldepriorität an	R/W
Origin	String	Gibt Meldeursprung an	R/W
Area	String	Gibt Meldebereich an	R/W
RaisedStateTag	String	Gibt meldungsauslös-sende Variable an	R/W
ConditionValue	object Supported type: Double, int16, , uint16, int32, uint32, int64, uint64, byte	Bedingungswert, der als konstanter Wert angegeben werden kann	R/W
Condition	HmiAlarmCondition	Gibt Begrenzungsmo-dus an	R/W
InfoText	MultilingualText	Gibt Infotext an	R/W
AlarmParameterTags	Object SupportedType: IList<string>	Bei einem Array von zehn Parametern die Variable, die den Mel-deparameter angibt. Jeder Parameter ver-wendet den Variablen-namen als String. Die-se Parameter können im Ereignistext verwen-det werden.	R/W
EventText1	MultilingualText	Ereignistext von HmiA-nalogAlarm	R/W
EventText2	MultilingualText	Ereignistext von HmiA-nalogAlarm	R/W
EventText3	MultilingualText	Ereignistext von HmiA-nalogAlarm	R/W
EventText4	MultilingualText	Ereignistext von HmiA-nalogAlarm	R/W
EventText5	MultilingualText	Ereignistext von HmiA-nalogAlarm	R/W
EventText6	MultilingualText	Ereignistext von HmiA-nalogAlarm	R/W

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
EventText7	MultilingualText	Ereignistext von HmiA-nalogAlarm	R/W
EventText8	MultilingualText	Ereignistext von HmiA-nalogAlarm	R/W
EventText9	MultilingualText	Ereignistext von HmiA-nalogAlarm	R/W

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Zugriff auf HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)

Programmcode

Bei der Arbeit mit Meldungen können Sie den folgenden Beispiel-Programmcode ändern und verwenden.

Analogmeldungen erstellen

Ändern Sie folgenden Programmcode, um eine Analogmeldung zu erstellen:

```
private void AnalogAlarmCreate()
{
//Create Analog Alarm
HmiSoftware hmiSoftware = GetHmiSoftware();
AnalogAlarmComposition analogAlarms = hmiSoftware.AnalogAlarms;
AnalogAlarm analogAlarm = analogAlarms.Create("Alarm_1");
}
```

Analogmeldungen löschen

Ändern Sie folgenden Programmcode, um eine Analogmeldung zu löschen:

```
private void AnalogAlarmDelete()
{
//Delete Analog Alarm
hmiSoftware = GetHmiSoftware();
AnalogAlarmComposition analogAlarms = hmiSoftware.AnalogAlarms;
AnalogAlarm analogAlarm = analogAlarms.Find("Alarm_1");
if (analogAlarm != null)
{
analogAlarm.Delete();
}
}
```

Analogmeldungen enumerieren

Ändern Sie den folgenden Programmcode, um Analogmeldungen zu enumerieren:

```
private void AnalogAlarmBrowse()
{
//Browse Analog Alarms
hmiSoftware = GetHmiSoftware();
AnalogAlarmComposition analogAlarms = hmiSoftware.AnalogAlarms;
foreach (AnalogAlarm analogAlarm in analogAlarms)
{
//work with analog alarms.
}
//Other way to get alarm by using Find function on Alarm Composition. AnalogAlarm
analogAlarmObj = analogAlarms.Find("Alarm_1");
}
```

Auf Eigenschaften von Analogmeldungen zugreifen

Ändern Sie folgenden Programmcode, um die Eigenschaften von Analogmeldungen festzulegen und abzurufen:

```
private void AnalogAlarmPropertiesAcess()
{
hmiSoftware = GetHmiSoftware();
AnalogAlarmComposition analogAlarms = hmiSoftware.AnalogAlarms;
AnalogAlarm analogAlarm = analogAlarms.Find("Alarm_1");
int id = analogAlarm.Id
analogAlarm.Id = 10;
int priority = analogAlarm.Priority; analogAlarm.Priority = 7;
analogAlarm.Area = "Area_1";
string area = analogAlarm.Area;
analogAlarm.Name = "NewAlarm";
string name = analogAlarm.Name;
}
```

7.18.4.2 Mit einzelnen Bitmeldungen arbeiten

Einleitung

Mit Bitmeldungen können Sie folgende Aktionen durchführen, während Sie TIA Portal Openess verwenden:

- Bitmeldungen erstellen
- Bitmeldungen löschen
- Bitmeldungen enumerieren
- Auf Eigenschaften von Bitmeldungen zugreifen

Eigenschaften

Die folgenden Eigenschaften werden in Bitmeldungen unterstützt:

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
EventText	MultilingualText	Gibt Ereignis/Meldetext der Meldung an	R/W
Id	uint32	Gibt Meldenummer zum Identifizieren der Meldung an. Hierbei handelt es sich um einen eindeutigen Wert für jede Bitmeldung	R/W
AlarmClass	String	Gibt Meldeklasse an	R/W
Name	String	Gibt den Meldenamen an	R/W
Priority	byte	Gibt Meldepriorität an	R/W
Origin	String	Gibt Meldeursprung an	R/W
Area	String	Gibt Meldebereich an	R/W
RaisedStateTag	String	Gibt meldungsauslösende Variable an	R/W
RaisedStateTagBitNumber	uint32	Auslösebit an Auslösevariab- le	R/W
TriggerMode	HmiDiscreteAlarmTiggerMo-de	Gibt Auslösemodus für Bit-meldung an	R/W
InfoText	MultilingualText	Gibt Infotext an	R/W
AlarmParameterTags	Object	Bei einem Array von zehn Parametern die Variable, die den Meldeparameter angibt. Jeder Parameter verwendet den Variablenamen als String. Diese Parameter können im Meldetext verwendet werden.	R/W
EventText1	MultilingualText	Ereignistext von HmiDiscre-teAlarm	R/W
EventText2	MultilingualText	Ereignistext von HmiDiscre-teAlarm	R/W

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
EventText3	MultilingualText	Ereignistext von HmiDiscreteAlarm	R/W
EventText4	MultilingualText	Ereignistext von HmiDiscreteAlarm	R/W
EventText5	MultilingualText	Ereignistext von HmiDiscreteAlarm	R/W
EventText6	MultilingualText	Ereignistext von HmiDiscreteAlarm	R/W
EventText7	MultilingualText	Ereignistext von HmiDiscreteAlarm	R/W
EventText8	MultilingualText	Ereignistext von HmiDiscreteAlarm	R/W
EventText9	MultilingualText	Ereignistext von HmiDiscreteAlarm	R/W

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Zugriff auf HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)

Programmcode

Bei der Arbeit mit Meldungen können Sie den folgenden Beispiel-Programmcode ändern und verwenden.

Bitmeldungen erstellen

Ändern Sie den folgenden Programmcode, um eine Bitmeldung zu erstellen:

```
HmiSoftware hmiSoftware = ...;
DiscreteAlarmComposition discreteAlarms = hmiSoftware.DiscreteAlarms;
DiscreteAlarm discreteAlarm = discreteAlarms.Create("Alarm_1");
```

Bitmeldungen löschen

Ändern Sie den folgenden Programmcode, um eine Bitmeldung zu löschen:

```
public void Delete()  
//User wants to delete a discrete alarm with the alarm name Alarm_1  
{  
HmiSoftware hmiSoftware = ....;  
DiscreteAlarmComposition discreteAlarms = hmiSoftware.DiscreteAlarms;  
DiscreteAlarm discreteAlarm = discreteAlarms.Find("Alarm_1");  
discreteAlarm.Delete();  
}
```

Bitmeldungen enumerieren

Ändern Sie den folgenden Programmcode, um Bitmeldungen zu enumerieren:

```
private void DiscreteAlarmBrowse()  
{  
//Browse Discrete Alarms  
HmiSoftware hmiSoftware = GetHmiSoftware();  
DiscreteAlarmComposition discreteAlarms = hmiSoftware.DiscreteAlarms;  
foreach (DiscreteAlarm discreteAlarm in discreteAlarms)  
{  
//work with discrete alarms.  
}  
//Other way to get alarm by using Find function on Alarm Composition.  
DiscreteAlarm discreteAlarmObj = discreteAlarms.Find("Alarm_1");  
}
```

Auf Eigenschaften von Bitmeldungen zugreifen

Ändern Sie den folgenden Programmcode, um auf die Eigenschaften von Bitmeldungen zuzugreifen:

```
HmiSoftware hmiSoftware = ....;  
DiscreteAlarm discreteAlarm = hmiSoftware.DiscreteAlarms[0];  
  
int id = discreteAlarm.Id;  
discreteAlarm.Id = 10;  
  
int priority = discreteAlarm.Priority;  
discreteAlarm.Priority = 7;  
  
discreteAlarm.area = "Aread1";  
string area = discreteAlarm.Area;  
  
discreteAlarm.Name = "NewAlarm";  
string name = discreteAlarm.Name;
```

7.18.4.3 Mit Meldeklassen arbeiten

Einleitung

Mit Meldeklassen können Sie folgende Aktionen durchführen, während Sie TIA Portal Openness verwenden:

- Meldeklassen erstellen
- Meldeklassen löschen
- Meldeklassen enumerieren
- Auf Eigenschaften von Meldeklassen zugreifen

Eigenschaften

Die folgenden Eigenschaften werden in Meldeklassen unterstützt:

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
Name	String	Name der Meldeklasse	Abhängig von Art der Meldeklasse (System/Benutzer)
Priority	byte	Priorität der Meldeklasse	R/W
IsSystem	Bool	Gibt an, ob die Meldeklasse vom System vorgegeben wird	R
StateMachine	HmiAlarmStateMachine	Gibt Maschinenzustand für Meldeklasse an	Abhängig von Art der Meldeklasse (System/Benutzer)
RaisedState	HmiRaisedState	Meldestatus kommend oder ausgelöst	R
RaisedState.BackColor	Color	Gibt Hintergrundfarbe an	R/W
RaisedState.TextColor	Color	Gibt Hintergrundfarbe an	R/W
RaisedState.Flashing	Bool	Gibt Blinken an	R/W
ClearedState	HmiClearedState	Gibt Meldestatus kommend, gehend oder gelöscht an	R
ClearedState.BackColor	Color	Gibt Hintergrundfarbe an	R/W
ClearedState.TextColor	Color	Gibt Textfarbe an	R/W
ClearedState.Flashing	Bool	Gibt Blinken an	R/W
AcknowledgedState	HmiAcknowledgedState	Gibt Meldestatus als kommend quittieren oder quittiert an	R
AcknowledgedState.BackColor	Color	Gibt Hintergrundfarbe an	R/W
AcknowledgedState.TextColor	Color	Gibt Textfarbe an	R/W
AcknowledgedState.Flashing	Bool	Gibt Blinken an	R/W
AcknowledgedClearedState	HmisAcknowledgedClearedState	Gibt Meldestatus als kommend, gehend quittieren oder quittiert/gelöscht an	
AcknowledgedClearedState.BackColor	Color	Gibt Hintergrundfarbe an	R/W

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
AcknowledgedClearedState.TextColor	Color	Gibt Textfarbe an	R/W
AcknowledgedClearedState.Flashing	Bool	Gibt Blinken an	R/W
Id	uint32	Gibt ID zum Kennzeichnen der Meldeklasse an. Hierbei handelt es sich um einen eindeutigen Wert für jede Meldeklasse	R
Log	System.String	Archiv der Meldeklasse	R/W
CommonAlarmClass	System.String	Allgemeine Meldeklasse	R

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Zugriff auf HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)

Programmcode

Bei der Arbeit mit Meldeklassen können Sie den folgenden Beispiel-Programmcode ändern und verwenden.

Meldeklassen erstellen

Ändern Sie den folgenden Programmcode, um eine Meldeklasse zu erstellen:

```
private void AlarmClassCreate()
{
//Create Alarm Class
HmiSoftware hmiSoftware = GetHmiSoftware();
AlarmClassComposition alarmClasses = hmiSoftware.AlarmClasses;
AlarmClass alarmClass = alarmClasses.Create("AlarmClass_1");
}
```

Meldeklassen löschen

Ändern Sie den folgenden Programmcode, um eine Meldeklasse zu löschen:

```
public void Delete()
//User wants to delete an alarm class with the alarm name AlarmClass_1
{
HmiSoftware hmiSoftware = ...;
AlarmClassComposition alarmClasses = hmiSoftware.AlarmClasses;
AlarmClass alarmClass = alarmClasses.Find("AlarmClass_1");
alarmClass.Delete();
}
```

Meldeklassen enumerieren

Ändern Sie den folgenden Programmcode, um Meldeklassen zu enumerieren:

```
//User wants to enumerate all discrete alarms of device with following code
{
HmiSoftware hmiSoftware = ...;
AlarmClassComposition alarmClasses = hmiSoftware.AlarmClasses;
foreach (AlarmClass alarmClass in alarmClasses)
}
{
...
}
```

Auf Eigenschaften von Meldeklassen zugreifen

Ändern Sie den folgenden Programmcode, um auf die Eigenschaften von Meldeklassen zuzugreifen:

```
HmiSoftware hmiSoftware = ...;
AlarmClassComposition alarmClasses = hmiSoftware.AlarmClasses;
AlarmClass alarmClass = hmiSoftware.AlarmClasses.Find("MostCritical");

int priority = alarmClass.Priority;
alarmClass.Priority = 10;

alarmClass.Acknowledgement = StateMachine.AlarmWithDualModeAcknowledgement;
alarmClass.IncomingOutgoingAcknowledgeStatus.BackColor = Color.Red;
alarmClass.IncomingOutgoingAcknowledgeStatus.TextColor = Color.Black;
alarmClass.IncomingOutgoingAcknowledgeStatus.Flashing = true;
```

7.18.5 Logs

7.18.5.1 Arbeiten mit Datenarchiven

Einleitung

Mit Datenarchiven können Sie folgende Aktionen durchführen, während Sie TIA Portal Openess verwenden:

- Datenarchiv erstellen
- Datenarchiv löschen
- Datenarchiv enumerieren
- Auf Eigenschaften von Datenarchiven zugreifen

Eigenschaften

Die folgenden Eigenschaften werden in einem Datenarchiv unterstützt:

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
Name	String	Gibt den Namen des Datenarchivs an	R/W
Settings	LogSettings	Komplexe Eigenschaft. Hat Eigenschaften (StorageFolder, LogMaxSize und LogTimePeriod) als Mitglieder, die die Einstellung bezogen auf das Archiv angeben.	R
	StorageFolder	Pfad für Archivierung	R/W
	LogMaxSize	Definiert die maximale Größe des Archivs auf dem Speichermedium in Megabyte	R/W
	LogTimePeriod	Definiert den vom Archiv abgedeckten maximalen Zeitraum	R/W
	StorageDevice	Definiert das Speichergerät	R/W
Segment	LogSegment	Komplexe Eigenschaft. Hat Eigenschaften (SegmentMaxSize, StartTime und SegmentTimePeriod) als Mitglieder, die die Einstellung bezogen auf das Segment angeben.	
	SegmentMax-Size	Definiert die maximale Größe des Archivsegments in Megabyte	R/W
	SegmentStart-Time	Definiert den genauen Zeitpunkt für den Start des Segments	R/W
	SegmentTime-Period	Definiert den vom Archivsegment abgedeckten maximalen Zeitraum	R/W

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
Backup	LogBackup	Komplexe Eigenschaft. Hat Eigenschaften (PrimaryPath und BackupMode) als Mitglieder, die die Einstellung bezogen auf die Archivsicherung angeben.	R
	HmiBackupMode	Definiert den Sicherungsmodus	R/W
	String	Pfad für Sicherung	R/W

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Zugriff auf HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)

Programmcode

Bei der Arbeit mit Datenarchiven können Sie den folgenden Beispiel-Programmcode ändern und verwenden.

Datenarchiv erstellen

Ändern Sie den folgenden Programmcode, um ein Datenarchiv zu erstellen:

```
private void DatalogCreate()
{
//Create Datalog
HmiSoftware hmiSoftware = GetHmiSoftware();
DataLogComposition dataLogs = hmiSoftware.DataLogs;
DataLog dataLog = dataLogs.Create("Datalog_1");
}
```

Datenarchiv löschen

Ändern Sie den folgenden Programmcode, um ein Datenarchiv zu löschen:

```
private void DatalogDelete()
{
//user wants to delete datalog with name "Datalog_1"
HmiSoftware hmiSoftware = GetHmiSoftware();
DataLogComposition dataLogs = hmiSoftware.DataLogs;
DataLog dataLog = dataLogs.Find("Datalog_1");
dataLog.Delete();
}
```

Datenarchiv enumerieren

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Ändern Sie den folgenden Programmcode, um Datenarchive zu enumerieren:

```
private void DatalogBrowse()
{
//Browse Datalog.
HmiSoftware hmiSoftware = GetHmiSoftware();
DataLogComposition dataLogs = hmiSoftware.DataLogs;
foreach (DataLog dataLog in dataLogs)
{
//work with data log.
}
//Other way to get data log by using Find function on datalog Composition.
DataLog dataLogObj = dataLogs.Find("Datalog_1");
}
```

Auf Eigenschaften von Datenarchiven zugreifen

Sie können den folgenden Programmcode ändern und verwenden, um auf die Eigenschaften von Datenarchiven zuzugreifen:

```
private void DatalogProperties()
{
//Set/Get Data Log properties.
HmiSoftware hmiSoftware = GetHmiSoftware();
DataLogComposition dataLogs = hmiSoftware.DataLogs;
DataLog dataLog = dataLogs.Find("Datalog_1");
string name = dataLog.Name;
dataLog.Settings.StorageFolder = @"C:\Logs\Log";
dataLog.Settings.LogMaxSize = Int32.MaxValue;
dataLog.Settings.LogMaxSize = 2000;
LogDuration duration = dataLog.Settings.LogTimePeriod;
double durationInDouble = duration.GetDoubleLogDuration();
string durationInString = duration.GetStringLogDuration();
duration.Days = 7;
duration.Hours = 23;
duration.Minutes = 50;
duration.Seconds = 0;
duration.Ticks = 1;
//Method to set log duration.
duration.SetLogDuration(10, 12, 4, 5, 0);
dataLog.Backup.BackupMode = BackupMode.NoBackup;
dataLog.Backup.BackupMode = BackupMode.PrimaryPath;
dataLog.Backup.PrimaryPath = @"C:\Logs\Backup";
dataLog.Segment.SegmentMaxSize = 500;
dataLog.Segment.SegmentStartTime = DateTime.Now;
dataLog.Segment.SegmentStartTime = DateTime.Now.Date;
}
```

7.18.5.2 Arbeiten mit Meldearchiven

Einleitung

Mit Meldearchiven können Sie folgende Aktionen durchführen, während Sie TIA Portal Openness verwenden:

- Meldearchiv erstellen
- Meldearchiv löschen
- Meldearchiv enumerieren
- Auf Eigenschaften von Meldearchiven zugreifen

Eigenschaften

Die folgenden Eigenschaften werden in einem Meldearchiv unterstützt:

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
Name	String	Gibt den Namen des Meldearchivs an	R/W
Settings	LogSettings	Komplexe Eigenschaft. Hat Eigenschaften (StorageFolder, LogMaxSize und LogTimePeriod) als Mitglieder, die die Einstellung bezogen auf das Archiv angeben.	R
	StorageFolder	Pfad für Archivierung	R/W
	LogMaxSize	Definiert die maximale Größe des Archivs auf dem Speichermedium in Megabyte	R/W
	LogTimePeriod	Definiert den vom Archiv abgedeckten maximalen Zeitraum	R/W
Segment	LogSegment	Komplexe Eigenschaft. Hat Eigenschaften (SegmentMaxSize, StartTime und SegmentTimePeriod) als Mitglieder, die die Einstellung bezogen auf das Segment angeben.	
	SegmentMaxSize	Definiert die maximale Größe des Archivsegments in Megabyte	R/W
	SegmentStartTime	Definiert den genauen Zeitpunkt für den Start des Segments	R/W
	SegmentTimePeriod	Definiert den vom Archivsegment abgedeckten maximalen Zeitraum	R/W
Backup	LogBackup	Komplexe Eigenschaft. Hat Eigenschaften (PrimaryPath und BackupMode) als Mitglieder, die die Einstellung bezogen auf die Archivsicherung angeben.	R
	BackupMode	Definiert den Sicherungsmodus	R/W
	PrimaryPath	Pfad für Sicherung	R/W

Voraussetzung

- Die Anwendung TIA Portal Openess ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Zugriff auf HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)

Programmcode

Bei der Arbeit mit Meldearchiven können Sie den folgenden Beispiel-Programmcode ändern und verwenden.

Meldearchive erstellen

Ändern Sie den folgenden Programmcode, um ein Meldearchiv zu erstellen:

```
private void AlarmLogCreate()
{
//Create Alarmlog
HmiSoftware hmiSoftware = GetHmiSoftware();
AlarmLogComposition alarmLogs = hmiSoftware.AlarmLogs;
AlarmLog alarmLog = alarmLogs.Create("Alarmlog_1");
}
```

Meldearchive löschen

Ändern Sie den folgenden Programmcode, um ein Meldearchiv zu löschen:

```
private void AlarmLogDelete()
{
//User wants to delete alarm log with name "Alarmlog_1"
HmiSoftware hmiSoftware = GetHmiSoftware();
AlarmLogComposition alarmLogs = hmiSoftware.AlarmLogs;
AlarmLog alarmLog = alarmLogs.Find("Alarmlog_1");
alarmLog.Delete();
}
```

Meldearchive enumerieren

Ändern Sie den folgenden Programmcode, um Meldearchive zu enumerieren:

```
private void AlarmLogBrowse()
{
//Browse alarm log.
HmiSoftware hmiSoftware = GetHmiSoftware();
AlarmLogComposition alarmLogs = hmiSoftware.AlarmLogs;
foreach (AlarmLog alarmLog in alarmLogs)
{
//work with alarm log.
}
//Other way to get alarmlog by using Find function on alarmlog Composition.
AlarmLog alarmLogObj = alarmLogs.Find("Alarmlog_1");
}
```

Auf Eigenschaften von Meldearchiven zugreifen

Sie können den folgenden Programmcode ändern und verwenden, um auf die Eigenschaften von Meldearchiven zuzugreifen.

```
private void AlarmLogProperties()
{
//Set/Get alarm log properties.
HmiSoftware hmiSoftware = GetHmiSoftware();
AlarmLogComposition alarmLogs = hmiSoftware.AlarmLogs;
AlarmLog alarmLog = alarmLogs.Find("Alarmlog_1");
string name = alarmLog.Name;
alarmLog.Settings.StorageFolder = @"C:\Logs\Log";
alarmLog.Settings.LogMaxSize = Int32.MaxValue;
alarmLog.Settings.LogMaxSize = 2000;
LogDuration duration = alarmLog.Settings.LogTimePeriod;
double durationInDouble = duration.GetDoubleLogDuration();
string durationInString = duration.GetStringLogDuration();
duration.Days = 7;
duration.Hours = 23;
duration.Minutes = 50;
duration.Seconds = 0;
duration.Ticks = 1;
//Log duration can be set by using function also.
duration.SetLogDuration(10, 12, 4, 5, 0);
alarmLog.Backup.BackupMode = BackupMode.NoBackup;
alarmLog.Backup.BackupMode = BackupMode.PrimaryPath;
alarmLog.Backup.PrimaryPath = @"C:\Logs\Backup";
alarmLog.Segment.SegmentMaxSize = 500;
alarmLog.Segment.SegmentStartTime = DateTime.Now;
alarmLog.Segment.SegmentStartTime = DateTime.Now.Date;
}
```

Siehe auch

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

7.18.5.3 Arbeiten mit Archivvariablen

Einleitung

Mit Archivvariablen können Sie folgende Aktionen durchführen, während Sie TIA Portal Openess verwenden:

- Archivvariable erstellen
- Archivvariable löschen
- Archivvariable enumerieren
- Auf Eigenschaften von Archivvariablen zugreifen

Eigenschaften

Die folgenden Eigenschaften werden in einer Archivvariable unterstützt:

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
AggregationDelay	System.TimeSpan	Wert für Aggregationsverzögerung	R/W
AggregationMode	HmiAggregationMode	Wert für Aggregationsmodus	R/W
Cycle	System.String	Archivierungszyklus der Archivvariable	R/W
CycleFactor	System.UInt32	Wert für Archivierungszyklusfaktor	R/W
DataLog	System.String	Datenarchiv der Archivvariable	R/W
HighLimit	System.Object	Oberer Grenzwert der Archivvariable	R/W
LimitScope	HmiLimitScope	Umfangsbegrenzung der Archivvariable	R/W
LoggingMode	HmiLoggingMode	Protokollierungsmodus der Archivvariable	R/W
LowLimit	System.Object	Unterer Grenzwert der Archivvariable	R/W
Name	System.string	Name der Archivvariable	R/W
SmoothingDeltaValue	System.Double	Delta der Archivvariable	R/W
SmoothingMaxTime	System.TimeSpan	Höchstzeit der Archivvariable	R/W
SmoothingMinTime	System.TimeSpan	Mindestzeit der Archivvariable	R/W
SmoothingMode	HmiSmoothingMode	Glättungsmodus der Archivvariable	R/W
Source	System.string	Quelle der Archivvariable	R/W
TriggerMode	HmiTriggerMode	Auslösemodus der Archivvariable	R/W

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
TriggerTag	System.String	Wert der Auslösevariable	R/W
TriggerTagBitNumber	System.UInt32	Wert von TriggerTagBitNumber	R/W

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Zugriff auf HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)

Programmcode

Bei der Arbeit mit Archivvariablen können Sie den folgenden Beispiel-Programmcode ändern und verwenden.

Archivvariablen erstellen

Zum Erstellen einer Archivvariablen können Sie den folgenden Programmcode ändern und verwenden:

```
private void LoggingTagCreate()
{
    //Create Logging Tag for given tag.
    HmiSoftware hmiSoftware = GetHmiSoftware();
    HmiTagComposition hmiTags = hmiSoftware.HmiTags;
    //Tag_1 exists in TIA project.
    HmiTag hmiTag = hmiTags.Find("Tag_1");
    LoggingTagComposition loggingTags = hmiTag.LoggingTags;
    LoggingTag loggingTag = loggingTags.Create("LoggingTag_1");
}
```

Archivvariablen löschen

Ändern Sie den folgenden Programmcode, um eine Archivvariable zu löschen:

```
private void LoggingTagDelete()
{
    //user wants to delete Logging tag with name "LoggingTag_1" for tag "Tag_1";
    HmiSoftware hmiSoftware = GetHmiSoftware();
    HmiTagComposition hmiTags = hmiSoftware.HmiTags;
    //Tag_1 exists in TIA project.
    HmiTag hmiTag = hmiTags.Find("Tag_1");
    LoggingTagComposition loggingTags = hmiTag.LoggingTags;
    //LoggingTag_1 exists in TIA project.
    LoggingTag loggingTag = loggingTags.Find("LoggingTag_1");
    loggingTag.Delete();
}
```

Archivvariablen enumerieren

Ändern Sie den folgenden Programmcode, um Archivvariablen zu enumerieren:

```
private void LoggingTagBrowse()
{
    HmiSoftware hmiSoftware = GetHmiSoftware();
    HmiTagComposition hmiTags = hmiSoftware.HmiTags;
    //Tag_1 exists in TIA project.
    HmiTag hmiTag = hmiTags.Find("Tag_1");
    LoggingTagComposition loggingTags = hmiTag.LoggingTags;
    foreach (LoggingTag loggingTag in loggingTags)
    {
        //...
    }
    //LoggingTag_1 exists in TIA project.
    LoggingTag loggingTagObj = loggingTags.Find("LoggingTag_1");
}
```

Auf Eigenschaften von Archivvariablen zugreifen

Sie können den folgenden Programmcode ändern und verwenden, um auf die Eigenschaften von Archivvariablen zuzugreifen:

```
private void LoggingTagProperties()
{
    //Set/Get LoggingTag properties.
    HmiSoftware hmiSoftware = GetHmiSoftware();
    HmiTagComposition hmiTags = hmiSoftware.HmiTags;
    //Tag_1 exists in TIA project.
    HmiTag hmiTag = hmiTags.Find("Tag_1");
    LoggingTagComposition loggingTags = hmiTag.LoggingTags;
    //LoggingTag_1 exists in TIA project.
    LoggingTag loggingTag = loggingTags.Find("LoggingTag_1");

    string name = loggingTag.Name;

    SmoothingMode smoothingMode = loggingTag.SmoothingMode;
    loggingTag.SmoothingMode = SmoothingMode.SwingingDoor;

    LimitScope limitScope = loggingTag.LimitScope;
    loggingTag.LimitScope = LimitScope.WithinLimits;

    string lowLimit = loggingTag.LowLimit;
    loggingTag.LowLimit = "-32768";

    string highLimit = loggingTag.HighLimit;
    loggingTag.HighLimit = "-3";

    TimeSpan smotthingMinTime = loggingTag.SmoothingMinTime;
    TimeSpan tsMin = TimeSpan.Parse("500");
    loggingTag.SmoothingMinTime = tsMin;

    TimeSpan smotthingMaxTime = loggingTag.SmoothingMaxTime;
    TimeSpan tsMax = TimeSpan.Parse("1000");
    loggingTag.SmoothingMaxTime = tsMax;

    double smoothingDeltaValue = loggingTag.SmoothingDeltaValue;
    loggingTag.SmoothingDeltaValue = 5;

    string dataLog = loggingTag.DataLog;
    loggingTag.DataLog = "Datalog_1";
}
```

Siehe auch

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

7.18.6 Variablen und Variablenklassen

7.18.6.1 Mit Variablenklassen arbeiten

Einleitung

Mit Variablenklassen können Sie folgende Aktionen durchführen, während Sie TIA Portal Openess verwenden:

- Variablenklassen erstellen
- Variablenklassen löschen
- Variablenklassen enumerieren
- Auf Eigenschaften von Variablenklassen zugreifen

Eigenschaften

Die folgenden Eigenschaften werden in einer Variablenklasse unterstützt:

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
Name	String	Gibt den Meldennamen an	R/W
Tags	HmiTagComposition	Liste der Variablen	R

Voraussetzung

- Die Anwendung TIA Portal Openess ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).

Programmcode

Bei der Arbeit mit Variablenklassen können Sie den folgenden Beispiel-Programmcode ändern und verwenden.

Variablenklassen erstellen

Ändern Sie den folgenden Programmcode, um eine Variablenklasse zu erstellen:

```
private void TagTableCreate()
{
//Create Tag Table
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagTableComposition tagTables = hmiSoftware.HmiTagTables;
HmiTagTable hmiTagTable = tagTables.Create("TagTable_1");
}
```

Variablenklassen löschen

Ändern Sie den folgenden Programmcode, um eine Variablenliste zu löschen:

```
private void TagTableDelete()
{
//User wants to delete tag table with name "TagTable_1"
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagTableComposition tagTables = hmiSoftware.HmiTagTables;
HmiTagTable tagTable = tagTables.Find("TagTable_1");
tagTable.Delete();
}
```

Variablenlisten enumerieren

Ändern Sie den folgenden Programmcode, um Variablenlisten zu enumerieren:

```
private void TagTableBrowse()
{
//Browse Tag Tables
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagTableComposition tagTables = hmiSoftware.HmiTagTables;
foreach (HmiTagTable tagTable in tagTables)
{
//Work with tag table
}
//Other way to get tag table by using Find function on tag table Composition.
HmiTagTable tagTableObj = tagTables.Find("TagTable_1");
}
```

Auf Eigenschaften von Variablenlisten zugreifen

Ändern Sie den folgenden Programmcode, um auf die Eigenschaften einer Variablenliste zuzugreifen:

```
private void TagTableProperties()
{
//Set/Get Tag Table properties.
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagTableComposition tagTables = hmiSoftware.HmiTagTables;
HmiTagTable tagTable = tagTables.Find("TagTable_1");
tagTable.Name = "NewTagTable";
string name = tagTable.Name;
}
```

7.18.6.2 Arbeiten mit HMI-Variablen

Einleitung

Mit HMI-Variablen können Sie folgende Aktionen durchführen, während Sie TIA Portal Openness verwenden:

- HMI-Variablen erstellen
- HMI-Variablen löschen
- HMI-Variablen enumerieren
- Auf HMI-Variable zugreifen
- Auf Eigenschaften von HMI-Variablen zugreifen
- Verwendung des Datentyps UDT für Variablen
- Eigenschaften von Member-Variablen von UDT

Eigenschaften

Die folgenden Eigenschaften werden in einer HMI-Variable unterstützt:

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
AccessMode	HmiAccessMode	Zugriffsmodus der HMI-Variablen	R/W
AcquisitionCycle	String	Attribut für Erfassungszyklus	R/W
AcquisitionMode	HmiTagAcquisitionMode	Erfassungsmodus der HMI-Variable	R/W
Comment	MultilingualText	Kommentar zu Variable abrufen/festlegen	R/W
Connection	String	HMI-Verbindung	R/W
DataType	String	Datentyp der Variablen	R/W
MaxLength	UInt32	HMI-Variable DataType-Length	R/W
DisplayName	MultilingualText	Anzeige für Namen der Variable abrufen/festlegen	R/W
Address	String	HMI-Variable Addressattribut	R/W
InitialMinValue	LowerRange	Unterer Grenzwert	R
Name	String	Name der HMI-Variable	R/W
Persistent	Boolean	Persistenzattribut	R/W
PlcName	String	PLC-Namensattribut	R
PlcTag	String	PLC-Variablenattribut	R/W
InitialValue	Object	Startwertattribut	R/W
SubstituteValue	HmiSubstituteValue	Ersatzwert	R
Updateld	UInt32	Aktualisierungs-ID-Attribut	R/W
Initial.MaxValue	UpperRange	Oberer Grenzwert	R
TagName	String	Name der Variablen, zu der die Variable gehört	R

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
HmiDataType	String	HMI-Datentyp der Variablen	
LinearScaling	Boolean	Lineare Skalierung	R/W
HmiStartValue	Object		R/W
HmiEndValue	Object		R/W
PlcStartValue	Object		R/W
PlcEndValue	Object		R/W

Die folgenden Eigenschaften sind im unteren/oberen Bereich verfügbar:

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
Value	Object	Oberer/unterer Wert	R/W
ValueType	HmiLimitValueType	Werttyp abfragen und festlegen	R/W

Die folgenden Eigenschaften sind in einem Ersatzwert verfügbar:

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
Value	Object	Werttyp abfragen und festlegen	R/W
SubstituteValueUsage	Hmi SubstituteValueUsage	Beschreibt, wann der Ersatzwert verwendet wird	R/W

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Zugriff auf HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)

Programmcode

Bei der Arbeit mit HMI-Variablen können Sie den folgenden Beispiel-Programmcode ändern und verwenden.

HMI-Variablen erstellen

Ändern Sie den folgenden Programmcode, um eine HMI-Variable zu erstellen:

```
private void CreateTag()
{
    HmiSoftware hmiSoftware = GetHmiSoftware();

    //Four ways to create tag.
    //1. Create tag by accessing HmiTags property at HmiSoftware level, by using Create method
    //with two parameters.
    HmiTagComposition hmiTags1 = hmiSoftware.HmiTags;
    //TagTable1 exists in TIA project.
    HmiTag hmiTag1 = hmiTags1.Create("Tag1", "TagTable1");

    //2. Create tag by accessing HmiTags property at HmiSoftware level, by using Create method
    //with single parameter.
    HmiTag hmiTag2 = hmiTags1.Create("Tag2");
    //Tags will be created in default tag table.

    //3. Creation of tag by accessing HmiTags property at Tag Table, by using Create method with
    //two parameters.
    //Tag creation will fail and it will result in recoverable exception,
    HmiTagComposition hmiTags2 = hmiSoftware.HmiTagTables.Find("Default tag table").HmiTags;
    HmiTag hmiTag3 = hmiTags2.Create("Tag_3", "TableTableName");

    //4. Create tag by accessing HmiTags property at Tag Table, by using Create method with
    //single parameter.
    HmiTag hmiTag4 = hmiTags2.Create("Tag_4");
}
```

HMI-Variablen löschen

Ändern Sie den folgenden Programmcode, um eine HMI-Variable zu löschen:

```
private void DeleteTag()
{
    HmiSoftware hmiSoftware = GetHmiSoftware();
    //1. Delete tag at HmiSoftware level.
    HmiTagComposition hmiTags1 = hmiSoftware.HmiTags;
    HmiTag hmiTag1 = hmiTags1.Find("Tag1");
    hmiTag1.Delete();
    //2. Delete tag at tag table level.
    HmiTagComposition hmiTags2 = hmiSoftware.HmiTagTables.Find("Default tag table").HmiTags;
    HmiTag hmiTag2 = hmiTags1.Find("Tag2");
    hmiTag2.Delete();
}
```

HMI-Variablen enumerieren

Ändern Sie den folgenden Programmcode, um HMI-Variablen zu enumerieren:

```
private void BrowseTag()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
//1. User can navigate all tag of device with following code.
HmiTagComposition hmiTags = hmiSoftware.HmiTags;
foreach (HmiTag hmiTag in hmiTags)
{
//...
}
//2. User can navigate all tags of given tag table with following code.
HmiTagComposition hmiTags2 = hmiSoftware.HmiTagTables.Find("Default tag table").HmiTags;
foreach (HmiTag hmiTag2 in hmiTags2)
{
//...
}
```

Auf HMI-Variable zugreifen

Ändern Sie folgenden Programmcode, um auf eine einzelne HMI-Variable über den Namen zuzugreifen:

```
private void AccessTag()
{
HmiSoftware hmiSoftware = GetHmiSoftware();
//User can search tag present in UA device with following code.
HmiTag hmiTag3 = hmiSoftware.HmiTags.Find("Tag1");
}
```

Ändern Sie folgenden Programmcode, um auf eine HMI-Variable über den Tabellennamen zuzugreifen:

```
{
HmiSoftware hmiSoftware = GetHmiSoftware();
// User can search tag present in given tag table with following code.
HmiTag hmiTag4 = hmiSoftware.HmiTagTables.Find("Default tag
table").HmiTags.Find("Tag1");
}
```

Auf Eigenschaften von HMI-Variablen zugreifen

Ändern Sie den folgenden Programmcode, um auf die Eigenschaften von HMI-Variablen zuzugreifen:

```
private void TagProperties()
{
    HmiSoftware hmiSoftware = GetHmiSoftware();
    //Tag_1 exists in TIA device project.
    HmiTag hmiTag = hmiSoftware.HmiTags.Find("Tag_1");
    //Enumeration type properties
    AccessMode accessMode = hmiTag.AccessMode;
    UpdateScope scope = hmiTag.UpdateScope;
    //Substitute value depends on data type of tag.
    hmiTag.SubstituteValue.Value = "144";
    hmiTag.SubstituteValue.OnCommunicationError = false;
    hmiTag.Name = "Tag_2";
    //Assign valid cycle name.
    hmiTag.AcquisitionCycle = "T1s";
    hmiTag.DataType = "int";
    hmiTag.PlcTag = "PlcTag_1";
    //Start value depends on data type of tag.
    hmiTag.StartValue = "1";
    hmiTag.LowerLimit.ValueType = LimitValueType.Constant;
    hmiTag.LowerLimit.Value = "HmiTag_1";
    //Comment property (multilingual text)
    string culture = "en-US";
    //get the language based on given culture. Culture is standard and it should have correct
    value.
    Language language = GetDeviceProject().LanguageSettings.Languages.Find(new
    System.Globalization.CultureInfo(culture));
    //get comment as multilingual text.
    MultilingualText multiLingualComment = hmiTag.Comment;
    //get all text items from comment property.
    MultilingualTextItemComposition texts = multiLingualComment.Items;
    //find text from text list based on language.
    MultilingualTextItem textItemEnglish = texts.Find(language);
    //get value in selected culture.
    string commentEng = textItemEnglish.Text;
}
```

Verwendung des Datentyps UDT für Variablen

Sie können den folgenden Programmcode ändern und verwenden, um den Datentyp UDT zu Variablen zuzuweisen und auf Member-Variablen zuzugreifen:

HMI-Variablen den Datentyp UDT zuweisen

Ändern Sie folgenden Programmcode, um HMI-Variablen den Datentyp UDT ohne Verbindung zuzuweisen:

```
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagComposition tags = hmiSoftware.HmiTags;
tags[0].Data Type = @"\\Project library\\Types\\New folder _3\\HmiUdt_string\\V
0.0.1";
```

Ändern Sie folgenden Programmcode, um HMI-Variablen den Datentyp UDT mit Verbindung zuzuweisen:

```
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagComposition tags = hmiSoftware.HmiTags;
tags[0].Connection = "Connection_1";
tags[0].Data Type = @"\\Project library\\Types\\New folder _3\\HmiUdt_string\\V
0.0.1";
```

Hinweis

Sie können vollständige Pfade verwenden, um HMI UDT als Datentyp zuzuweisen.

Ändern Sie folgenden Programmcode, um HMI-Variablen den Datentyp PLC UDT mit Verbindung zuzuweisen:

```
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiTagComposition tags = hmiSoftware.HmiTags;
tags[0].Connection = "HMI_Connection_1";
tags[0].Data Type = "PlcTag_1";
```

Auf Member-Eigenschaften des Datentyps für HMI-Variable zugreifen

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Ändern Sie folgenden Programmcode, um auf Member-Eigenschaften des Datentyps für HMI-Variablen zuzugreifen:

```
private static void GetProperties(HmiTagComposition hmiTags)
{
foreach (HmiTag hmiTag in hmiTags)
{
switch(hmiTag.TagType)
{
case HmiTagType.Simple
Console.WriteLine("Name : " + hmiTag.Name);
GetSimpleTagProperties(HmiTag hmiTag)
break;
case HmiTagType.UDT;
case HmiTagType.Array;
GetProperties(hmiTag.Members); //recursive call to current function.
break;
}
}
}

private static void GetSimpleTagProperties(HmiTag hmiTag)
{
Console.WriteLine("Name : " +hmiTag.Name);
Console.WriteLine("Date type : " +hmiTag.DataType);
// List the properties of hmi tag
//
//...
}

private static void GetProperties(HmiTag hmiTag.Members)
{
Console.Writeline("Name : " +hmiTag.Members.Name);
Console.Writeline("Date type: " +hmiTag.Members.DataType);
// List the properties of hmi tag members
//...
//...
}
```

Eigenschaften von Member-Variablen von UDT

Sie können den folgenden Programmcode ändern und verwenden, um die Eigenschaften von Member-Variablen des Datentyps UDT (User Defined Datatype) festzulegen und abzurufen:

Ändern Sie folgenden Programmcode, um die Eigenschaften von Member-Variablen des benutzerdefinierten Datentyps festzulegen:

```

public HmiTag GetTag()
{
DeviceItem deviceItem = Tiaproject.Devices[0].DeviceItems[1];
SoftwareContainer softCont = deviceItem.GetService<SoftwareContainer>();
HmiSoftware hmiSoftware = softCont.Software as HmiSoftware;
if (hmiSoftware == null)
{
Device device = Tiaproject.Devices.Find("PC-System_1");
DeviceItem deviceItem1 = device.DeviceItems[1];
SoftwareContainer softCont1 = deviceItem1.GetService<SoftwareContainer>();
hmiSoftware = softCont1.Software as HmiSoftware;
}
HmiTagComposition tagComposition = hmiSoftware.Tags;
var tag = tagComposition.Create("Test_Tag");
return tag;
}
public void ReadWrite_HMIUDTTagDisplayNamePropertyWithInternalConn_Succeeds()
{
HmiTag tag = GetTag();
tag.DataType = "HmiUdt_1_WithInternalConn V 0.0.1";
for (int i = 0; i < tag.Members.Count; i++)
{
for (int j = 0; j < tag.Members[i].DisplayName.Items.Count; j++)
{
tag.Members[i].DisplayName.Items[j].Text = "DisplayNameTest" + j;
Console.WriteLine("Display Name : ", tag.Members[i].DisplayName.Items[j].Text);
}
}
}
///
/// <summary>
/// HMI UDT Tag Display Name Property with S7 1500 PLC Connection.
/// </summary>
public void ReadWrite_HMIUDTTagDisplayNamePropertyWithS71500_Succeeds()
{
HmiTag tag = GetTag();
tag.Connection = "HMI_Connection_1";
tag.DataType = "HmiUdt_1_WithS71500Conn V 0.0.1";
for (int i = 0; i < tag.Members.Count; i++)
{
for (int j = 0; j < tag.Members[i].DisplayName.Items.Count; j++)
{
tag.Members[i].DisplayName.Items[j].Text = "DisplayNameTest" + j;
Console.WriteLine("Display Name : ", tag.Members[i].DisplayName.Items[j].Text);
}
}
}

```

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Ändern Sie folgenden Programmcode, um auf die Eigenschaften von Member-Variablen des benutzerdefinierten Datentyps zuzugreifen:

```
public HmiTag GetTag()
{
DeviceItem deviceItem = Tiaproject.Devices[0].DeviceItems[1];
SoftwareContainer softCont = deviceItem.GetService<SoftwareContainer>();
HmiSoftware hmiSoftware = softCont.Software as HmiSoftware;
if (hmiSoftware == null)
{
Device device = Tiaproject.Devices.Find("PC-System_1");
DeviceItem deviceItem1 = device.DeviceItems[1];
SoftwareContainer softCont1 = deviceItem1.GetService<SoftwareContainer>();
hmiSoftware = softCont1.Software as HmiSoftware;
}
HmiTagComposition tagComposition = hmiSoftware.Tags;
var tag = tagComposition.Create("Test_Tag");
return tag;
}
public void Read_HMIUDTTagNamePropertyWithInternalConn_Succeeds()
{
HmiTag tag = GetTag();
tag.DataType = "HmiUdt_1_WithInternalConn V 0.0.1";
for (int i = 0; i < tag.Members.Count; i++)
{
var name = tag.Members[i].Name;
Console.WriteLine("Name : ", name );
}
}
/// <summary>
/// HMI UDT Tag Display Name Property with S71500 connection
/// </summary>
public void Read_HMIUDTTagNamePropertyWithS71500Conn_Succeeds()
{
HmiTag tag = GetTag();
tag.Connection = "HMI_Connection_1";
tag.DataType = "HmiUdt_1_WithS71500Conn V 0.0.1";
for (int i = 0; i < tag.Members.Count; i++)
{
var name = tag.Members[i].Name;
Console.WriteLine("Name : ", name );
}
```

Siehe auch

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

7.18.6.3 Arbeiten mit Systemvariablen

Einleitung

Mit Systemvariablen können Sie folgende Aktionen durchführen, während Sie TIA Portal Openness verwenden:

- Auf Systemvariable zugreifen
- Auf Eigenschaften von Systemvariablen zugreifen

Eigenschaften

Die folgenden Eigenschaften werden in einer Systemvariable unterstützt:

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
Name	String	Gibt den Namen der Systemvariable an	R
DataType	String	Gibt den Datentyp der Systemvariable an	R

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Zugriff auf HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)

Programmcode

Auf Systemvariable zugreifen

Ändern Sie den folgenden Programmcode, um auf eine einzelne Systemvariable über den Namen zuzugreifen:

```
private void SystemTagName()
{
    HmiSoftware hmiSoftware = GetHmiSoftware();
    HmiSystemTagComposition hmiSystemTags = hmiSoftware.HmiSystemTags;
    //Find system tag by using its name.
    HmiSystemTag hmiSystemTagObj = hmiSystemTags.Find("@UserName");
}
```

Auf Eigenschaften von Systemvariablen zugreifen

Ändern Sie den folgenden Programmcode, um auf Eigenschaften von Systemvariablen zuzugreifen:

```
private void SystemTagProperties()
{
    HmiSoftware hmiSoftware = GetHmiSoftware();
    HmiSystemTagComposition hmiSystemTags = hmiSoftware.HmiSystemTags;
    HmiSystemTag hmiSystemTag = hmiSystemTags.Find("@UserName");
    string name = hmiSystemTag.Name;
    string dataType = hmiSystemTag.DataType;
}
```

Siehe auch

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

7.18.7 Verbindungen

7.18.7.1 Mit Zusammensetzungen arbeiten

Einleitung

Mit Verbindungen können Sie folgende Aktionen durchführen, während Sie TIA Portal Openness verwenden:

- Verbindungen erstellen
- Verbindungen löschen
- Verbindungen enumerieren
- Auf Eigenschaften von Verbindungen zugreifen

Eigenschaften

Die folgenden Eigenschaften werden in HMI-Verbindungen unterstützt:

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
Name	String	Name der Verbindung	R/W
DisabledAtStartup	Boolean	Verbindung ist in Runtime anfänglich online oder nicht.	R/W
CommunicationDriver	String	Gibt den Kommunikations-treiber an	R/W
Node	String	Zeigt den Zugangspunkt des Partners (z.B. PLC)	R

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
Partner	String	Name der verbundenen PLC	R
Station	String	Name der Station, der die PLC zugeordnet ist	R
Comment	String	Gegebenenfalls zusätzliche Kommentare	R/W
InitialAddress	String	Parameter der Verbindung wie Schnittstellentyp (DP, TCP/IP), IP-Adresse, Baugruppenträger usw.	R/W

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden. Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet. Projekt öffnen (Seite 111)

Programmcode

Bei der Arbeit mit Verbindungen können Sie den folgenden Beispiel-Programmcode ändern und verwenden.

Verbindungen erstellen

Ändern Sie folgenden Programmcode, um eine Verbindung zu erstellen:

```
private void ConnectionCreate()
{
//Create Connection
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiConnectionComposition connections = hmiSoftware.HmiConnections;
HmiConnection connection = connections.Create("Connection_1");
}
```

Verbindungen löschen

Ändern Sie den folgenden Programmcode, um eine Verbindung zu löschen:

```
private void ConnectionDelete()
{
//User wants to delete connection with name "Connection_1"
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiConnectionComposition connections = hmiSoftware.HmiConnections;
HmiConnection connection = connections.Find("Connection_1")
connection.Delete();
}
```

Verbindungen enumerieren

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Ändern Sie den folgenden Programmcode, um Verbindungen zu enumerieren:

```
private void ConnectionBrowse()
{
//Browse Connections
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiConnectionComposition connections = hmiSoftware.HmiConnections;
foreach (HmiConnection connection in connections)
{
//Work with connections
}
//Other way to get connection by using Find function on connection Composition
HmiConnection connectionObj = connections.Find("Connection_1");
}
```

Auf Eigenschaften von Verbindungen zugreifen

Ändern Sie den folgenden Programmcode, um auf die Eigenschaften von Verbindungen zuzugreifen:

```
private void ConnectionProperties()
{
//Set/Get Connection properties.
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiConnectionComposition connections = hmiSoftware.HmiConnections;
HmiConnection connection = connections.Find("Connection_1");
string name = connection.Name;
connection.Name = "Connection_2";
bool online = connection.DisabledAtStartup;
connection.DisabledAtStartup = true;
string node = connection.Node;
string station = connection.Station;
string partner = connection.Partner;
string communicationDriver = connection.CommunicationDriver;
connection.CommunicationDriver = "SIMATIC S7 300/400";
//valid Initial Address strings
connection.InitialAddress = "PlcRack = 4";
connection.InitialAddress = "HostAddress = 127.157.2.2";
connection.InitialAddress = "hostAddress = 127.157.2.2";//key name is case insensitive so
it is valid"
connection.InitialAddress = "CommunicationInterface = Industrial Ethernet; HostAddress =
127.157.8.1; HostAccessPoint = S7ONLINE; PlcExpansionSlot = 1; PlcRack = 1;
PlcIsCyclicOperation = false";
connection.InitialAddress = "CommunicationInterface = Industrial Ethernet; HostAddress =
127.157.8.1; HostAccessPoint = S7ONLE; PlcRack = 5; PlcIsCyclicOperation = true";
//Below are Wrong Value or Wrong Formats which leads to recoverable exceptions.
//Wrong Value
connection.InitialAddress = null;
connection.InitialAddress = string.Empty;
connection.InitialAddress = "HostAddress = 127.157.2.2.1";
connection.InitialAddress = "HostAddress = 127.157.2.2.1; PlcRack = 5";
//key value format is not followed.
connection.InitialAddress = "HostAccessPoint = S7ONINE; PlcAddress * 155.166.8.2;
PlcExpansionSlot ; 1; PlcRack = 5; PlcIsCyclicOperation = true";
//semicolon with empty key value pair
connection.InitialAddress = ";;CommunicationInterface = Industrial Ethernet; HostAddress =
127.157.0.1";
connection.InitialAddress = ";CommunicationInterface = Industrial Ethernet; HostAddress =
127.157.0.1";
connection.InitialAddress = "CommunicationInterface = Industrial Ethernet; HostAddress =
127.157.0.1;";
connection.InitialAddress = "CommunicationInterface = Industrial Ethernet;; HostAddress =
127.157.0.1;";
connection.CommunicationDriver = "SIMATIC S7 1500";
//wrong as PlcRack is not valid key / property for Simatic S71500 connection.
connection.InitialAddress = "PlcRack = 4";
}
```

7.18.8 Laufzeiteinstellungen

7.18.8.1 Arbeiten mit Laufzeiteinstellungen

Einleitung

Mit Laufzeiteinstellungen können Sie folgende Aktionen durchführen, während Sie TIA Portal Openness verwenden:

- Laufzeiteinstellungen lesen/schreiben

Eigenschaften

Die folgenden Eigenschaften werden in Runtime-Einstellungen unterstützt:

Name der Eigenschaft	Datentyp	Beschreibung	Zugriff
HmiSoftware.RuntimeSettings	HmiRuntimeSetting	Runtime-Einstellungsobjekt	R
RuntimeSettings.StartScreen	String	Startbildschirm	R/W
RuntimeSettings.OperateAsOpcServer	Boolean	Gibt an, dass ein gegebenes Gerät als OPC-Server fungieren soll	R/W
RuntimeSettings.LanguageAndFonts	HmiLanguageAndFont Association	Liste der Sprachen und Schriftarten	R
RuntimeSettings. LanguageAndFonts [index].Enable	Boolean	Runtime-Sprache aktivieren/deaktivieren	R/W
RuntimeSettings. LanguageAndFonts [index].EnableForLogging	Boolean	Gibt Sprache für Archivierung in Runtime an	R/W
RuntimeSettings. LanguageAndFonts [index].Order	Short	Gibt Befehl für Übertragung von Schriftarten an Gerät	R
RuntimeSettings. LanguageAndFonts [index].Language	String	Name der Sprache	R
RuntimeSettings. LanguageAndFonts [index].FixedFont1	String	Vordefinierte Schriftartenfamilie für Schriftartenkonfiguration verfügbar	R
RuntimeSettings. LanguageAndFonts [index].FixedFont2	String	Vordefinierte Schriftartenfamilie für Schriftartenkonfiguration verfügbar	R
RuntimeSettings. LanguageAndFonts [index].FixedFont3	String	Vordefinierte Schriftartenfamilie für Schriftartenkonfiguration verfügbar	R
RuntimeSettings. LanguageAndFonts [index].FixedFont4	String	Vordefinierte Schriftartenfamilie für Schriftartenkonfiguration verfügbar	R

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Zugriff auf das HMI-Softwareobjekt.
Siehe HMI-Unified-Software-Objekt (Seite 281)

Programmcode

```
private void RuntimeSettingsPropertiesAccess()
{
    HmiSoftware hmiSoftware = GetHmiSoftware();
    hmiSoftware.RuntimeSettings.OperateAsOpcServer = true;
    hmiSoftware.RuntimeSettings.StartScreen = "Screen_1";
    hmiSoftware.RuntimeSettings.LanguageAndFonts[0].Enable = true;
    hmiSoftware.RuntimeSettings.LanguageAndFonts[0].EnableForLogging = true;
}
```

7.18.9 Bilder und Dynamisierungen

7.18.9.1 Mit Bildern arbeiten

Einleitung

Mit Bildern können Sie folgende Aktionen durchführen, während Sie TIA Portal Openness verwenden:

- Bilder erstellen
- Bilder löschen
- Bilder enumerieren
- Auf Eigenschaften von Bildern zugreifen

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Zugriff auf HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)

Programmcode

Bei der Arbeit mit Bildern können Sie den folgenden Beispiel-Programmcode ändern und verwenden.

Bilder erstellen

Ändern Sie folgenden Programmcode, um Bilder zu erstellen:

```
HmiSoftware hmisoftware = GetHMISoftware();
HmiScreenComposition objScreens = hmisoftware.Screens;
HmiScreen objHmiScreen = objScreens.Create("TestScreen");
```

Recoverable Exception wird ausgelöst, wenn die Methode Create mit einem Bildnamen aufgerufen wird, zu dem bereits ein Bild im TIA Portal vorhanden ist.

Bilder löschen

Ändern Sie folgenden Programmcode, um Bilder zu löschen:

```
public void Delete()
//Case 1
HmiSoftware hmisoftware = GetHMISoftware();
HmiScreenComposition objScreens = hmisoftware.Screens;
HmiScreen objHmiScreen = objScreens.Create("TestScreen");
if (objHmiScreen != null)
{
objHmiScreen.Delete();
}
//Case 2
HmiSoftware hmisoftware = GetHMISoftware();
HmiScreenComposition objScreens = hmisoftware.Screens;
objScreens.Create("TestScreen1");
IEngineeringObject alarmClassEnggObj = (IEngineeringObject)objScreens[0];
if (alarmClassEnggObj != null)
{
alarmClassEnggObj.Invoke("Delete", null);
}
```

Bilder enumerieren

Ändern Sie folgenden Programmcode, um alle Bilder eines Geräts zu enumerieren:

```
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreenComposition objScreens = hmiSoftware.Screens;
foreach (HmiScreen ObjHmiScreen in objScreens);
{
//work with Screens
}
```

Ändern Sie folgenden Programmcode, um Bilder in Bilderlisten anhand des Namens zu suchen:

```
HmiScreen objHmiScreen = objScreens.Find("TestScreen3");
```

Ändern Sie folgenden Programmcode, um Bilder in Bilderlisten anhand des Index zu finden:

```
HmiScreen objHmiScreen = hmiSoftware.Screens[0];
```

Ändern Sie folgenden Programmcode, um mit der Methode Contains zu ermitteln, ob in einer Liste von Bildobjekten ein bestimmtes Bildobjekt vorhanden ist:

```
HmiScreenComposition objScreens = hmisoftware.Screens;
HmiScreen screen1 = objScreen.Create("TestScreen1");
bool isExist = objScreens.Contains(screen1);
```

Auf Eigenschaften von Bildern zugreifen

Ändern Sie folgenden Programmcode, um die Eigenschaften eines Bildes festzulegen und abzurufen:

```
HmiSoftware hmisoftware = GetHmiSoftware();
HmiScreenComposition objScreens = hmisoftware.Screens;
HmiScreen objHmiScreen = objScreens.Create("TestScreen");
objHmiScreen.Width = 50;
objHmiScreen.ScreenNumber = 1;
objHmiScreen.BackGraphic = "DownArrow";
objHmiScreen.DisplayName.Items[0].Text = "screen";
objHmiScreen.Name = "ScreenTest";
objHmiScreen.Enabled = true;
objHmiScreen.Height = 123;
```

Ändern und verwenden Sie den folgenden Programmcode, um mit GetAttributes() alle Eigenschaften eines Bildes abzurufen:

```
HmiSoftware hmisoftware = GetHMISoftware();
HmiScreenComposition objScreens = hmisoftware.Screens;
HmiScreen objHmiScreen = objScreens.Create("TestScreen");
List<string> getlststring = new List<string>();
{
  "AlternateBackColor", "BackColor", "BackFillPattern", "BackGraphic",
  "BackGraphicStretchMode", "BackgroundFillMode", "DisplayName", "Dynamization", "Enabled",
  "EnabledExplicitRelease", "Height", "HorizontalAlignment", "Layers", "Name", "Parent",
  "ScreenElements", "ScreenItems", "ScreenNumber", "VerticalAlignment", "Width"
};
var getallpropValue = objHmiScreen.GetAttributes(getlststring);
```

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Ändern Sie folgenden Programmcode, um mit SetAttributes() alle Eigenschaften eines Bildes festzulegen und abzurufen:

```
HmiSoftware hmisoftware = GetHMISoftware();
HmiScreenComposition objScreens = hmisoftware.Screens;
HmiScreen objHmiScreen = objscreens.Create("TestScreen");
Dictionary<string, object> setPropertyName = new Dictionary<string, object>();
setPropertyName.Add("AlternateBackColor", Color.Aqua);
setPropertyName.Add("BackColor", Color.Aqua);
setPropertyName.Add("BackFillPattern", HmiFillPattern.GradientHorizontalTricolor);
setPropertyName.Add("BackGraphic", "DownArrow");
setPropertyName.Add("BackGraphicStretchMode", HmiGraphicStretchMode.Fill);
setPropertyName.Add("BackgroundFillMode", HmiBackgroundFillMode.Screen);
setPropertyName.Add("DisplayName", "AnalogAlarmByDiffMethod");
setPropertyName.Add("Enabled", "AnalogAlarmByDiffMethod");
setPropertyName.Add("EnabledExplicitRelease", "AnalogAlarmByDiffMethod");
setPropertyName.Add("Height", "AnalogAlarmByDiffMethod");
setPropertyName.Add("HorizontalAlignment", "AnalogAlarmByDiffMethod");
setPropertyName.Add("Name", "AnalogAlarmByDiffMethod");
setPropertyName.Add("ScreenNumber", "AnalogAlarmByDiffMethod");
setPropertyName.Add("VerticalAlignment", "AnalogAlarmByDiffMethod");
setPropertyName.Add("Name", "AnalogAlarmByDiffMethod");
setPropertyName.Add("Width", "AnalogAlarmByDiffMethod");
objHmiScreen.SetAttributes(setPropertyName);
```

Ändern und verwenden Sie den folgenden Programmcode, um mit IEngineeringObject's GetAttribute den Wert einer Eigenschaft abzurufen:

```

HmiSoftware hmisoftware = GetHMISoftware();
HmiScreenComposition objScreens = hmisoftware.Screens;
HmiScreen objHmiScreen = objScreens.Create("Testscreen");
IEngineeringObject obj = objHmiScreen;
Color clraltnvbk = (Color)obj.GetAttribute("AlternateBackColor");
Color clrbk = (Color)obj.GetAttribute("BackColor");
HmiFillPattern filptrns = (HmiFillPattern)obj.GetAttribute("BackFillPattern");
string bkgrphic = (string)obj.GetAttribute("BackGraphic");
HmiGraphicStretchMode strmode =
(HmiGraphicStretchMode)obj.Getattribute("BackGraphicStretchmode");
HmiBackgroundFillMode bkgrndfilmode = (HmiBackground
dFillMode)obj.GetAttribute("BackgroundFillMode");
string dsplyName = (string)obj.GetAttribute("DisplayName");
DynamizationBaseComposition dymztns =
(DynamizationBaseComposition)obj.GetAttribute("Dynamizations");
b001 enble = (b001)obj.GetAttritute("Enabled");
SomRef objsom = (SomRef)obj.GetAttribute("EnableExplicitRelease");
uint fight = (uint)obj.GetAttribute("Height");
HmiHorizontalAlignment hrzntl =
(HmiHorizontalAlignment)obj.GetAttribute("HorizontalAlignment");
HmiverticalAlignment vrtcle = (HmiverticalAlignment)obj.GetAttribute("VerticalAlignment");
HmiLayerPartComposition objlyrs = (HmiLayerPartComposition)Obj.GetAttribute("layers");
string name = (string)obj.GetAttribute("Name");
IEngineeringObject objparen = (IEngineeringobject)obj.GetAttribute("Parent");
HmiscreenElementBaseComposition objelemnt =
(HmiscreenElementBaseComposition)obj.GetAttribute("ScreenElements");
HmiScreenItemBaseComposition objitems =
(HmiScreenItemBaseComposition)obj.GetAttribute("ScreenItems");
byte scrnnumber = (byte)obj.GetAttribute("ScreenNumber");
uint width = (uint)obj.GetAttribute(Width");

```

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Ändern Sie folgenden Programmcode, um festzustellen, ob alle Eigenschaften gültige Werte haben:

```
HmiSoftware hmiSoftware = GetHMISoftware ();
HmiScreen objHmiScreen = hmiSoftware.Screens.Create("HmiScreen_1");
objHmiScreen.DisplayName.Items[0].Text = GetMorethan512BoundaryString();
IList<IValidator> objectsToValidate = new List<IValidator>();
foreach (var item in hmiSoftware.Screens)
{
    objectsToValidate.Add(item);
}
foreach (IValidator validator in objectsToValidate)
{
    IList<HmiValidationResult> errors = validator.Validate();
    if (errors != null && errors.Count > 0)
    {
        foreach (var errornotification in errors)
        {
            var propName = errornotification.PropertyName;
            foreach (var errormassage in errornotification.Errors)
            {
                Console.WriteLine(errormassage);
            }
        }
    }
}
```

Hinweis

Wenn beim Festlegen von Eigenschaften der Wert nicht durch den Schreibvorgang festgelegt werden kann, wird eine Recoverable exception ausgelöst.

7.18.9.2 Grundlegende Bildobjekte

Mit grundlegenden Bildobjekten arbeiten

Einleitung

Mit grundlegenden Bildobjekten können Sie folgende Aktionen durchführen, während Sie TIA Portal Openess verwenden:

- (Grundlegende) Bildobjekte erstellen
- (Grundlegende) Bildobjekte löschen
- (Grundlegende) Bilder enumerieren

Die nachstehend aufgeführten grundlegenden Bildobjekte werden von TIA Portal Openness unterstützt.

Tabelle 7-1 Typen und Namensraum/-räume

Bildobjekt	Klasse	Namensraum/-räume
Linie	HmiLine	Siemens.Engineering.HmiUnified.UI.Shapes
Polygonzug	HmiPolyline	Siemens.Engineering.HmiUnified.UI.Base
Polygon	HmiPolygon	
Ellipse	HmiEllipse	
Ellipsensegment	HmiEllipseSegment	
Kreissegment	HmiCircleSegment	
Ellipsenbogen	HmiEllipticalArc	
Kreisbogen	HmiCircularArc	
Kreis	HmiCircle	
Rechteck	HmiRectangle	
Grafikanzeige	HmiGraphicView	

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 79)

Programmcode

Bei der Arbeit mit grundlegenden Bildobjekten können Sie den folgenden Beispiel-Programmcode ändern und verwenden.

Bildobjekte erstellen (grundlegende)

Ändern Sie folgenden Programmcode, um grundlegende Bildobjekte zu erstellen:

```
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiLine hmiLine = screenitems.Create<HmiLine>("ScreenItem_1");
```

Hinweis

Das obige Beispiel gilt für alle Bildobjekte, die in der Typentabelle oben definiert wurden. Die HmiLine muss mit einem Typ/mit Typen definiert werden, der/die in der Tabelle "Typen und Namensraum/-räume" definiert ist/sind.

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Recoverable Exception wird ausgelöst, wenn die Methode Create mit dem Namen eines Bildobjekts aufgerufen wird, das in TIA bereits vorhanden ist.

Bildobjekte löschen (grundlegende)

Ändern Sie folgenden Programmcode, um grundlegende Bildobjekte zu löschen:

```
public void Delete()
//Case 1
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screen[0].ScreenItems;
HmiLine hmiLine = screenitems.Create<HmiLine>("ScreenItem_1");
if (hmiLine != null)
{
hmiLine.Delete();
}
//Case 2
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Scree1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiLine hmiLine = screenitems.Create<HmiLine>("ScreenItem_1");
IEngineeringObject ObjhmiLineEnggObj = hmiLine;
if (ObjhmiLineEnggObj != null)
{
ObjHmiScreenItemssEnggObj.Invoke("Delete", null);
}
```

Hinweis

Das obige Beispiel gilt für alle Bildobjekte, die in der Typentabelle oben definiert wurden. Die HmiLine muss mit einem Typ/mit Typen definiert werden, der/die in der Tabelle "Typen und Namensraum/-räume" definiert ist/sind.

(Grundlegende) Bilder enumerieren

Ändern Sie folgenden Programmcode, um alle Bildobjekte eines Bildes zu enumerieren:

```
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
foreach (var item in screenitems)
{
//work with screenitems
}
```

Ändern Sie folgenden Programmcode, um Bildobjekte in einer Bildobjektliste anhand des Namens zu finden:

```
HmiScreenItemBase screenitems = hmiSoftware.Screens[0].ScreenItems.Find("ScreenItems_1");
```

Ändern Sie folgenden Programmcode, um Bildobjekte in einer Bildobjektliste anhand des Index zu finden:

```
HmiScreenItemBase screenitem = hmiSoftware.Screens[0].ScreenItems[0];
```

Ändern Sie folgenden Programmcode, um mit der Methode Contains zu ermitteln, ob in einer Liste von Bildobjekten ein bestimmtes Bildobjekt vorhanden ist:

```
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiLine hmiLine = screenitems.Create<HmiLine>("ScreenItems_1");
bool isexists = screenitems.Contains(hmiLine);
```

Hinweis

Das obige Beispiel gilt für alle Bildobjekte, die in der Typentabelle definiert wurden. Die HmiLine muss mit einem Typ/mit Typen definiert werden, der/die in der Tabelle "Typen und Namensraum/-räume" definiert ist/sind.

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

Auf Eigenschaften von Linien zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften eines Linien-Bildobjekts abrufen und einstellen.

Die folgenden Eigenschaften einer Linie werden im Bildobjekt unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
X1	int	False
Y1	int	False
X2	int	False
Y2	int	False
Enabled	Bool	True
CurrentQuality	HmiQuality	False
LineColor	Color	False
AlternateLineColor	Color	False
DashType	HmiDashType	False
EndType	HmiLineEndType	False
StartType	HmiLineStartType	False

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
CapType	HmiCapType	False
LineWidth	byte	False
Top	int	False
Left	int	False
Width	uint	False
Height	uint	False
RotationAngle	short	False
RotationCenterX	float	False
RotationCenterY	float	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
Opacity	float	False
Name	string	False
Visible	bool	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt werden erstellt
Siehe Mit grundlegenden Bildobjekten arbeiten (Seite 328)

Programmcode

Ändern Sie folgenden Code, um auf grundlegende Eigenschaften von Linien zuzugreifen:

```
//Name
var name = hmiline.Name;
hmiline.Name = "Default Value";
//AlternateLineColor
var linecolor = hmiline.AlternateLineColor;
hmiline.AlternateLineColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//Height
var height = hmiline.Height;
hmiline.Height = 100;
```

Ändern Sie folgenden Code, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText
var tooltip = hmiline.ToolTipText;
var tooltiptext = hmiline.ToolTipText.Items[0].Text;
hmiline.ToolTipText.Items[0].Text = "<body><p>TestforMultilinugualProperty</p></body>";
```

Ändern Sie folgenden Code, um auf eine andere typische Eigenschaft der Linie zuzugreifen:

```
//DashType
var dashtype = hmiline.DashType;
hmiline.DashType = HmiDashType.DashDotDot;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[Mit grundlegenden Bildobjekten arbeiten \(Seite 328\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

Auf Eigenschaften eines Polygonzugs zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften eines Polygonzugs abrufen und einstellen.

Die folgenden Eigenschaften eines Polygonzugs werden im Bildobjekt unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
LineColor	Color	False
Enabled	Bool	False
CurrentQuality	HmiQuality	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
AlternateLineColor	Color	False
DashType	HmiDashType	False
EndType	HmiLineEndType	False
StartType	HmiLineStartType	False
CapType	HmiCapType	False
LineWidth	byte	False
JoinType	HmiLineJoinType	False
Top	int	False
Left	int	False
Width	uint	False
Height	uint	False

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
RotationAngle	short	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Name	string	False
Visible	bool	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt werden erstellt
Siehe Mit grundlegenden Bildobjekten arbeiten (Seite 328)

Programmcode

Ändern Sie folgenden Programmcode, um auf die grundlegenden Eigenschaften eines Polygonzugs zuzugreifen:

```
//Name
var name = polyline.Name;
polyline.Name = "Default Value";
//AlternateLineColor
var linecolor = polyline.AlternateLineColor;
polyline.AlternateLineColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//Height
var height = polyline.Height;
polyline.Height = 100;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText
var tooltip = polyline.ToolTipText;
var tooltiptext = polyline.ToolTipText.Items[0].Text;
polyline.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Ändern Sie folgenden Programmcode, um auf eine andere typische Eigenschaft zuzugreifen:

```
//Joint Type
var dashtype = polyline.JoinType;
polyline.JoinType = HmiLineJoinType.Miter;
//Points
var points = polyline.Points;
var point = points[0];
var x = point.X;
point.X = 10;
var y = point.Y;
point.Y = 10;
var newPoint = points.Create(15, 100);
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit grundlegenden Bildobjekten arbeiten \(Seite 328\)](#)

Auf Eigenschaften einer Ellipse zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften eines Ellipsen-Bildobjekts abrufen und einstellen.

Die folgenden Eigenschaften einer Ellipse werden im Bildobjekt unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
BorderColor	Color	False
AlternateBorderColor	Color	False
BackColor	Color	False
AlternateBackColor	Color	False
BorderWidth	byte	False
Enabled	Bool	False
CurrentQuality	HmiQuality	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
BackFillPattern	HmiFillPattern	False
FillLevel	byte	False
FillDirection	HmiFillDirection	False
DashType	HmiDashType	False
ShowFillLevel	bool	False
RadiusX	uint	False
RadiusY	uint	False

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
CenterX	int	False
CenterY	int	False
RotationAngle	short	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Name	string	False
Visible	bool	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit grundlegenden Bildobjekten arbeiten (Seite 328)

Programmcode

Ändern Sie folgenden Programmcode, um auf die grundlegenden Eigenschaften einer Ellipse zuzugreifen:

```
//Name
var name = ellipse.Name;
ellipse.Name = "Default Value";
//AlternateBorderColor
var bordercolor = ellipse.AlternateBorderColor;
ellipse.AlternateBorderColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//BorderWidth
var borderwidth = ellipse.BorderWidth;
ellipse.BorderWidth = 10;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText
var tooltip = ellipse.ToolTipText;
var tooltiptext = ellipse.ToolTipText.Items[0].Text;
ellipse.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Ändern Sie folgenden Programmcode, um auf eine andere typische Eigenschaft einer Ellipse zuzugreifen:

```
//BackFillPattern
var backfill = ellipse.BackFillPattern;
ellipse.BackFillPattern = HmiFillPattern.GradientBackwardDiagonal;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit grundlegenden Bildobjekten arbeiten \(Seite 328\)](#)

Auf Eigenschaften eines Ellipsensegments zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften eines Ellipsensegments abrufen und einstellen.

Die folgenden Eigenschaften eines Ellipsensegments werden im Bildobjekt unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
StartAngle	int	False
AngleRange	int	False
BorderColor	Color	False
AlternateBorderColor	Color	False
BackColor	Color	False
CurrentQuality	HmiQuality	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
BorderWidth	byte	False
BackFillPattern	HmiFillPattern	False
FillLevel	byte	False
FillDirection	HmiFillDirection	False
DashType	HmiDashType	False
ShowFillLevel	bool	False
RadiusX	uint	False
RadiusY	uint	False
CenterX	int	False
CenterY	int	False
RotationAngle	short	False
RotationCenterX	float	False
RotationCenterY	float	False

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
RotationCenterPlacement	HmiRotationCenterPlacement	False
Opacity	float	False
Name	string	False
Visible	bool	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt werden erstellt
Siehe Mit grundlegenden Bildobjekten arbeiten (Seite 328)

Programmcode

Ändern Sie folgenden Programmcode, um auf die grundlegenden Eigenschaften eines Ellipsensegments zuzugreifen:

```
//Name
var name = ellipsesegment.Name;
ellipsesegment.Name = "Default Value";
//AlternateBorderColor
var bordercolor = ellipsesegment.AlternateBorderColor;
ellipsesegment.AlternateBorderColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//BorderWidth
var borderwidth = ellipsesegment.Height;
ellipsesegment.BorderWidth = 10;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText
var tooltip = ellipsesegment.ToolTipText;
var tooltiptext = ellipsesegment.ToolTipText.Items[0].Text;
ellipsesegment.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Ändern Sie folgenden Programmcode, um auf eine andere typische Eigenschaft zuzugreifen:

```
//BackFillPattern
var backfill = ellipsesegment.BackFillPattern;
ellipsesegment.BackFillPattern = HmiFillPattern.GradientBackwardDiagonal;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

Auf Eigenschaften eines Kreissegments zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften eines Kreissegments abrufen und einstellen.

Die folgenden Eigenschaften eines Kreissegments werden im Bildobjekt unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
StartAngle	int	False
AngleRange	int	False
BorderColor	Color	False
AlternateBorderColor	Color	False
BackColor	Color	False
AlternateBackColor	Color	False
Enabled	Bool	False
CurrentQuality	HmiQuality	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
BorderWidth	byte	False
BackFillPattern	HmiFillPattern	False
FillLevel	byte	False
FillDirection	HmiFillDirection	False
DashType	HmiDashType	False
ShowFillLevel	bool	False
Radius	uint	False
CenterX	int	False
CenterY	int	False
RotationCenterX	float	False
RotationCenterY	float	False
RotationAngle	short	False
Opacity	float	False
Name	string	False
Visible	bool	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit grundlegenden Bildobjekten arbeiten (Seite 328)

Programm

Ändern Sie folgenden Programmcode, um auf eine grundlegende Eigenschaft eines Kreissegments zuzugreifen:

```
//Name
var name = circlesegment.Name;
circlesegment.Name = "Default Value";
//AlternateBorderColor
var bordercolor = circlesegment.AlternateBorderColor;
circlesegment.AlternateBorderColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//BorderWidth
var borderwidth = circlesegment.BorderWidth;
circlesegment.BorderWidth = 10;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText
var tooltip = circlesegment.ToolTipText;
var tooltiptext = circlesegment.ToolTipText.Items[0].Text;
circlesegment.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Ändern Sie folgenden Programmcode, um auf eine andere typische Eigenschaft zuzugreifen:

```
//ToolTipText
var tooltip = circlesegment.ToolTipText;
var tooltiptext = circlesegment.ToolTipText.Items[0].Text;
circlesegment.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Siehe auch

Verbindung zum TIA Portal aufbauen (Seite 79)

Projekt öffnen (Seite 111)

HMI-Unified-Software-Objekt (Seite 281)

Mit grundlegenden Bildobjekten arbeiten (Seite 328)

Auf Eigenschaften eines Ellipsenbogens zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften eines Ellipsenbogens abrufen und einstellen.

Die folgenden Eigenschaften eines Ellipsenbogens werden im Bildobjekt unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
StartAngle	int	False
AngleRange	int	False
LineColor	Color	False
AlternateLineColor	Color	False
StartType	HmiLineStartType	False
EndType	HmiLineEndType	False
CapType	HmiCapType	False
Enabled	Bool	False
CurrentQuality	HmiQuality	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
DashType	HmiDashType	False
LineWidth	byte	False
RadiusX	uint	False
RadiusY	uint	False
CenterX	int	False
CenterY	int	False
RotationAngle	short	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Name	string	False
Visible	bool	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt werden erstellt
Siehe Mit grundlegenden Bildobjekten arbeiten (Seite 328)

Programmcode

Ändern Sie folgenden Programmcode, um auf eine grundlegende Eigenschaft von Ellipsenbögen zuzugreifen:

```
//Name
var name = ellipticalarc.Name;
ellipticalarc.Name = "Default Value";
//LineColor
var linecolor = ellipticalarc.LineColor;
ellipticalarc.LineColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//LineWidth
var linewidth = ellipticalarc.LineWidth;
ellipticalarc.LineWidth = 10;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText
var tooltip = ellipticalarc.ToolTipText;
var tooltiptext = ellipticalarc.ToolTipText.Items[0].Text;
ellipticalarc.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Ändern Sie folgenden Programmcode, um auf eine andere typische Eigenschaft zuzugreifen:

```
//CapType
var captype = ellipticalarc.CapType;
ellipticalarc.CapType = HmiCapType.Round;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit grundlegenden Bildobjekten arbeiten \(Seite 328\)](#)

Auf Eigenschaften eines Kreisbogens zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften eines Kreisbogens abrufen und einstellen.

Die folgenden Eigenschaften eines Kreisbogens werden im Bildobjekt unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
StartAngle	int	False
AngleRange	int	False
LineColor	Color	False
AlternateLineColor	Color	False
DashType	HmiDashType	False
EndType	HmiLineEndType	False
StartType	HmiLineEndType	False
Enabled	Bool	False
CurrentQuality	HmiQuality	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
CapType	HmiCapType	False
LineWidth	byte	False
Radius	uint	False
CenterX	int	False
CenterY	int	False
RotationAngle	short	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Name	string	False
Visible	bool	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt werden erstellt
Siehe Mit grundlegenden Bildobjekten arbeiten (Seite 328)

Programmcode

Ändern Sie folgenden Programmcode, um auf eine grundlegende Eigenschaft eines Kreisbogens zuzugreifen:

```
//Name
var name = circulararc.Name;
circulararc.Name = "Default Value";
//LineColor
var linecolor = circulararc.LineColor;
circulararc.LineColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//LineWidth
var linewidth = circulararc.LineWidth;
circulararc.LineWidth = 10;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText
var tooltip = circulararc.ToolTipText;
var tooltiptext = circulararc.ToolTipText.Items[0].Text;
circulararc.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Ändern Sie folgenden Programmcode, um auf eine andere typische Eigenschaft zuzugreifen:

```
//CapType
var captype = circulararc.CapType;
circulararc.CapType = HmiCapType.Round;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit grundlegenden Bildobjekten arbeiten \(Seite 328\)](#)

Auf Eigenschaften eines Kreises zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften eines Kreises abrufen und einstellen.

Die folgenden Eigenschaften eines Kreises werden im Bildobjekt unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
BorderColor	Color	False
AlternateBorderColor	Color	False
BackColor	Color	False

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
AlternateBackColor	Color	False
Enabled	Bool	False
CurrentQuality	HmiQuality	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
BorderWidth	byte	False
BackFillPattern	HmiFillPattern	False
FillLevel	byte	False
FillDirection	HmiFillDirection	False
DashType	HmiDashType	False
ShowFillLevel	bool	False
Radius	uint	False
CenterX	int	False
CenterY	int	False
RotationAngle	short	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Name	string	False
Visible	bool	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt werden erstellt
Siehe Mit grundlegenden Bildobjekten arbeiten (Seite 328)

Programmcode

Ändern Sie folgenden Programmcode, um auf eine grundlegende Eigenschaft eines Kreises zuzugreifen:

```
//Name
var name = circle.Name;
circle.Name = "Default Value";
//BorderColor
var bordercolor = circle.BorderColor;
circle.BorderColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);
//BorderWidth
var borderwidth = circle.BorderWidth;
circle.BorderWidth = 10;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText
var tooltip = circle.ToolTipText;
var tooltiptext = circle.ToolTipText.Items[0].Text;
circle.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Ändern Sie folgenden Programmcode, um auf eine andere typische Eigenschaft zuzugreifen:

```
//DashType
var dashtype = circle.DashType;
circle.DashType = HmiDashType.Solid;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit grundlegenden Bildobjekten arbeiten \(Seite 328\)](#)

Auf Eigenschaften eines Rechtecks zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften eines Rechtecks abrufen und einstellen.

Die folgenden Eigenschaften eines Rechtecks werden im Bildobjekt unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
BorderColor	Color	False
AlternateBorderColor	Color	False

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
BackColor	Color	False
AlternateBackColor	Color	False
BorderWidth	byte	False
BackFillPattern	HmiFillPattern	False
Enabled	Bool	False
CurrentQuality	HmiQuality	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
Corners	HmiCornersPart	True
FillLevel	byte	False
FillDirection	HmiFillDirection	False
DashType	HmiDashType	False
ShowFillLevel	bool	False
Top	int	False
Left	int	False
Width	uint	False
Height	uint	False
RotationAngle	short	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Name	string	False
Visible	bool	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt werden erstellt
Siehe Mit grundlegenden Bildobjekten arbeiten (Seite 328)

Programmcode

Ändern Sie folgenden Programmcode, um auf grundlegende Eigenschaften zuzugreifen:

```
//Name  
var name = rectangle.Name;  
rectangle.Name = "Default Value";  
//BorderColor  
var bordercolor = rectangle.BorderColor;  
rectangle.BorderColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);  
//BorderWidth  
var borderwidth = rectangle.BorderWidth;  
rectangle.BorderWidth = 10;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText  
var tooltip = rectangle.ToolTipText;  
var tooltiptext = rectangle.ToolTipText.Items[0].Text;  
rectangle.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Ändern Sie folgenden Programmcode, um auf eine andere typische Eigenschaft zuzugreifen:

```
//Corners  
var corner = rectangle.Corners;  
var bottomleftradius = corner.BottomLeftRadius;  
corner.BottomLeftRadius = 50;  
var bottomrightradius = corner.BottomRightRadius;  
corner.BottomRightRadius = 30;  
var toplefradius = corner.TopLeftRadius;  
corner.TopLeftRadius = 50;  
var toprightradius = corner.TopRightRadius;  
corner.TopRightRadius = 30;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit grundlegenden Bildobjekten arbeiten \(Seite 328\)](#)

Auf Eigenschaften einer Grafikanzeige zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften einer Grafikanzeige abrufen und einstellen.

Die folgenden Eigenschaften eines Rechtecks werden im Bildobjekt unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
GraphicStretchMode	HmiGraphicStretchMode	False
FillLevel	byte	False
FillDirection	HmiFillDirection	False
ShowFillLevel	bool	False
BackFillPattern	HmiFillPattern	False
AlternateBackColor	Color	False
Enabled	Bool	False
CurrentQuality	HmiQuality	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
Top	int	False
Left	int	False
Width	uint	False
Height	uint	False
Padding	HmiPaddingPart	True
RotationAngle	short	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Name	string	False
Visible	bool	False
Graphic	string	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Authorization	string	False

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt werden erstellt
Siehe Mit grundlegenden Bildobjekten arbeiten (Seite 328)

Programmcode

Ändern Sie folgenden Programmcode, um auf eine grundlegende Eigenschaft zuzugreifen:

```
//Name  
var name = graphicview.Name;  
graphicview.Name = "Default Value";  
//AlternateBackColor  
var backcolor = graphicview.AlternateBackColor;  
graphicview.AlternateBackColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);  
//Height  
var height = graphicview.height;  
graphicview.Height = 10;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText  
var tooltip = graphicview.ToolTipText;  
var tooltiptext = graphicview.ToolTipText.Items[0].Text;  
graphicview.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Ändern Sie folgenden Programmcode, um auf eine andere typische Eigenschaft zuzugreifen:

```
//Padding  
var padding = graphicview.Padding;  
var bottom = padding.Bottom;  
padding.Bottom = 50;  
var left = padding.Left;  
padding.Left = 60;  
var right = padding.Right;  
padding.Right = 70;  
var top = padding.Top;  
padding.Top = 50;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit grundlegenden Bildobjekten arbeiten \(Seite 328\)](#)

7.18.9.3 Element-Bildobjekte

Mit Element-Bildobjekten arbeiten

Einleitung

Mit Bildobjekten (Element) können Sie folgende Aktionen durchführen, während Sie TIA Portal Openness verwenden:

- Bildobjekt erstellen (Element)
- Bildobjekt löschen (Element)
- Bildobjekt enumerieren (Element)

Die folgenden Bildobjekte werden von TIA Portal Openness unterstützt. Verwenden Sie für den Zugriff auf die Klassen und Datentypen im Zusammenhang mit Bildobjekten die folgenden Namensräume und Datentypen:

Tabelle 7-2 Typen und Namensraum/-räume

Bildobjekt	Klasse	Namensraum/-räume
E/A-Feld	HmiIOField	Siemens.Engineering.HmiUnified.ModernUI.Widgets
Schaltfläche	HmiButton	Siemens.Engineering.HmiUnified.ModernUI.Base
Schalter	HmiToggleSwitch	
Kontrollkästchen	HmiCheckBoxGroup	
Balken	HmiBar	
Zeigerinstrument	HmiGauge	
Schieberegler	HmiSlider	
Optionsschaltfläche	HmiRadioButtonGroup	
Listenfeld	HmiListBox	
Uhr	HmiClock	
Textfeld	HmiTextBox	

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)

Programmcode

Bei der Arbeit mit Element-Bildobjekten können Sie den folgenden Beispiel-Programmcode ändern und verwenden.

Bildobjekte erstellen (Element)

Ändern Sie folgenden Programmcode, um Element-Bildobjekte zu erstellen:

```
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screen[0].ScreenItems;
HmiIOField hmiIOField = screenitems.Create<HmiIOField>("ScreenItem_1");
```

Recoverable Exception wird ausgelöst, wenn die Methode Create mit dem Namen eines Bildobjekts aufgerufen wird, das in TIA bereits vorhanden ist.

Hinweis

Das obige Beispiel gilt für alle in der obigen Typentabelle definierten Bildobjekte. Das HmiIOField muss mit einem Typ/mit Typen definiert werden, der/die in der Tabelle "Typen und Namensraum/-räume" definiert ist/sind.

Bildobjekte löschen (Element)

Ändern Sie folgenden Programmcode, um ein Element-Bildobjekt zu löschen:

```
public void Delete()
//Case 1
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiIOField hmiIOField = screenitems.Create<HmiIOField>("ScreenItems_1");
if (hmiIOField != null)
{
hmiIOField.Delete();
}
//Case 2
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiIOField hmiIOField = screenitems.Create<HmiIOField>("ScreenItem_1");
IEngineeringObject ObjhmiIOFieldEnggObj = hmiIOField;
if (ObjhmiIOFieldEnggObj != null)
{
ObjhmiIOFieldEnggObj.Invoke("Delete", null);
}
```

Hinweis

Das obige Beispiel gilt für alle in der obigen Typentabelle definierten Bildobjekte. Das HmiIOField muss mit einem Typ/mit Typen definiert werden, der/die in der Tabelle "Typen und Namensraum/-räume" definiert ist/sind.

Bildobjekte enumerieren (Element)

Ändern Sie folgenden Programmcode, um alle Bildobjekte eines Bilds zu enumerieren:

```
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
foreach (var item in screenitems)
{
//work with screenitems
}
```

Ändern Sie folgenden Programmcode, um Bildobjekte in einer Bildobjektliste anhand des Namens zu suchen:

```
HmiScreenItemBase screenitems = hmiSoftware.Screens[0].ScreenItems.Find("ScreenItems_1");
```

Ändern Sie folgenden Programmcode, um Bildobjekte in einer Bildobjektliste anhand des Index zu finden:

```
HmiScreenItemBase screenitem = hmiSoftware.Screens[0].ScreenItems[0];
```

Ändern Sie folgenden Programmcode, um mit der Methode Contains zu ermitteln, ob in einer Liste von Bildobjekten ein bestimmtes Bildobjekt vorhanden ist:

```
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiIOField hmiIOField = screenitems.Create<HmiIOField>("ScreenItems_1");
bool isexists = screenitems.Contains(hmiIOField);
```

Hinweis

Das obige Beispiel gilt für alle in der obigen Typentabelle definierten Bildobjekte. Das HmiIOField muss mit einem Typ/mit Typen definiert werden, der/die in der Tabelle "Typen und Namensraum/-räume" definiert ist/sind.

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

Auf Eigenschaften eines EA-Felds zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften eines EA-Felds abrufen und einstellen.

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Die folgenden Eigenschaften von EA-Feldern werden im Bildobjekt unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
AlternateBackColor	Color	False
Enabled	Bool	False
AlternateBorderColor	Color	False
Authorization	String	True
BackColor	Color	False
BorderColor	Color	False
BorderWidth	byte	False
CurrentQuality	HmiQuality	True
Font	HmiFontPart	False
ForegroundColor	Color	False
Height	uint	False
HorizontalTextAlignment	HmiHorizontalAlignment	False
IOFieldType	HmiIOFieldType	False
Top	int	False
Left	int	False
Rotation	short	False
VisualizeQuality	bool	False
TextTrimming	HmiTextTrimming	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Name	string	False
OutputFormat	string	False
Padding	HmiPaddingPart	True
RotationCenterPlacement	HmiRotationCenterPlacement	False
ProcessValue	string	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
VerticalTextAlignment	HmiVerticalAlignment	False
Visible	bool	False
Width	uint	False
Require Explicit Release	bool	False
InputBehavior	HmiInputBehaviorPart	True

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt werden erstellt
Siehe Mit grundlegenden Bildobjekten arbeiten (Seite 351)

Programmcode

Ändern Sie folgenden Programmcode, um auf grundlegende Eigenschaften eines EA-Felds zuzugreifen:

```
//Name
var name = iofield.Name;
iofield.Name = "DefaultName";
//AlternateBackColor
var altbackcolor = iofield.AlternateBackColor;
iofield.AlternateBackColor = Color.Beige;
//Height
var height = iofield.Height;
iofield.Height = 100;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText
HmiIOField ioField = createdscreen.ScreenItems.Create<HmiIOField>("hmiIOField");
string culture = "en-US";
Language lang = Tiaproject.LanguageSettings.Languages.Find(new CultureInfo(culture));
MultilingualText mltp = ioField.ToolTipText;
MultilingualTextItemComposition textItemComp = mltp.Items;
MultilingualTextItem mltextitem = textItemComp.Find(lang);
mltextitem.Text = "CommentInEnglish";
```

Ändern Sie folgenden Programmcode, um auf andere typische Eigenschaften zuzugreifen:

```
//Padding
var padding = iofield.Padding;
var bottom = padding.Bottom;
padding.Bottom = 50;
padding.Left = 60;
padding.Right = 70;
padding.Top = 50;
```

Siehe auch

- Verbindung zum TIA Portal aufbauen (Seite 79)
- Projekt öffnen (Seite 111)
- HMI-Unified-Software-Objekt (Seite 281)
- Mit Element-Bildobjekten arbeiten (Seite 351)

Auf Eigenschaften einer Schaltfläche zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften von Schaltflächen in Bildobjekten abrufen und einstellen.

Die folgenden Eigenschaften einer Schaltfläche werden in Bildobjekten unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
AlternateBackgroundColor	Color	False
AlternateBorderColor	Color	False
Authorization	String	True
BackgroundColor	Color	False
BorderColor	Color	False
BorderWidth	byte	False
Contents	HmiContentPart	False
CurrentQuality	HmiQuality	True
Enabled	bool	False
Font	HmiFontPart	False
ForegroundColor	Color	False
Graphic	string	False
Height	uint	False
Top	int	False
Left	int	False
Rotation	short	False
VisualizeQuality	bool	False
RotationCenterX	float	False
RotationCenterY	float	False
Style Item Appearance	HmiStyleItem Appearance	False
Opacity	float	False
Name	string	False
Padding	HmiPaddingPart	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
TabIndex	ushort	False
Text	MultiLingualText	True
ToolTipText	MultilingualText	True
Visible	bool	False
Width	uint	False
Require Explixit	bool	False
AlternateGraphic	string	False
AlternateText	MultiLingualText	False

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit Element-Bildobjekten arbeiten (Seite 351)

Programmcode

Ändern Sie folgenden Programmcode, um auf die grundlegenden Eigenschaften einer Schaltfläche zuzugreifen:

```
//Name
var name = hmiButton.Name;
hmiButton.Name = "Default Value";
//AlternateBackColor
var altbackcolor = hmiButton.AlternateBackColor;
hmiButton.AlternateBackColor = Color.Beige;
//Height
var height = hmiButton.Height;
hmiButton.Height = 100;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText
string culture = "en-US";
Language lang = Tiaproject.LanguageSettings.Languages.Find(new CultureInfo(culture));
MultilingualText mltp = hmiButton.ToolTipText;
MultilingualTextItemComposition textItemComp = mltp.Items;
MultilingualTextItem mlttextitem = hmiButton.Find(lang);
mlttextitem.Text = "TestforMultilingualProperty";
```

Ändern Sie folgenden Programmcode, um auf eine andere typische Eigenschaft zuzugreifen:

```
//Padding
var padding = hmiButton.Padding;
var bottom = padding.Bottom;
padding.Bottom = 50;
padding.Left = 60;
padding.Right = 70;
padding.Top = 50;
```

Siehe auch

- [Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- [Projekt öffnen \(Seite 111\)](#)
- [HMI-Unified-Software-Objekt \(Seite 281\)](#)
- [Mit Element-Bildobjekten arbeiten \(Seite 351\)](#)

Auf Eigenschaften eines Schalters zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften von Schaltern in Bildobjekten abrufen und einstellen.

Die folgenden Eigenschaften von Schaltern werden in Bildobjekten unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
AlternateBackgroundColor	Color	False
AlternateBorderColor	Color	False
IsAlternateState	bool	False
Authorization	string	True
BackgroundColor	Color	False
BorderColor	Color	False
BorderWidth	byte	False
Contents	HmiContentPart	False
CurrentQuality	HmiQuality	True
Enabled	bool	False
Font	HmiFontPart	False
ForegroundColor	Color	False
Height	uint	False
Top	int	False
Left	int	False
Rotation	short	False
Text	MultiLingualText	True
VisualizeQuality	bool	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Name	string	False
Padding	HmiPaddingPart	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
TabIndex	ushort	False
ToolTipText	MultilingualText	False
Visibility	bool	False
Width	uint	False

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
Require Explcit	bool	False
Graphic	string	False
AlternateGraphic	string	False
AlternateText	MultiLingualText	True
Padding	HmiPaddingPart	True
SystemItemClass	HmiButtonStyleItemClass	True

Hinweis

Da die Eigenschaft StyleItemClass schreibgeschützt ist, wird der Abrufvorgang für sie unterstützt, während der Schreibvorgang eine Ausnahme auslöst.

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit Element-Bildobjekten arbeiten (Seite 351)

Programmcode

Ändern Sie folgenden Programmcode, um auf die grundlegenden Eigenschaften eines Schalters zuzugreifen:

```
//Name
var name = toggleswitch.Name;
toggleswitch.Name = "Default Value";
//AlternateBackColor
var altbackcolor = toggleswitch.AlternateBackColor;
toggleswitch.AlternateBackColor = Color.Beige;
//Height
var height = toggleswitch.Height;
toggleswitch.Height = 100;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText
var tooltip = toggleswitch.ToolTipText;
var tooltiptext = toggleswitch.ToolTipText.Items[0].Text;
toggleswitch.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Ändern Sie folgenden Programmcode, um auf eine andere typische Eigenschaft eines Schalters zuzugreifen:

```
//Font
var font = toggleswitch.Font;
var italic = font.Italic;
font.Italic = false;
var fontname = font.FontName;
font.FontName = HmiFontName.SimSun;
var size = font.Size;
font.Size = 10.2f;
var stike = font.StrikeOut;
font.StrikeOut = true;
var underline = font.Underline;
font.Underline = false;
font.Bold = true;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit Element-Bildobjekten arbeiten \(Seite 351\)](#)

Auf Eigenschaften von Kontrollkästchen zugreifen

Einleitung

Mit TIA Portal Openess können Sie die Eigenschaften von Kontrollkästchen in Bildobjekten abrufen und einstellen.

Die folgenden Eigenschaften von Kontrollkästchen werden in Bildobjekten unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
AlternateBackgroundColor	Color	False
AlternateBorderColor	Color	False
Authorization	string	False
BackgroundColor	Color	False
BorderColor	Color	True
BorderWidth	byte	False
Content	HmiContentPart	False
CurrentQuality	HmiQuality	False
Enabled	bool	False
Font	HmiFontPart	True
ForegroundColor	Color	False
Height	uint	False

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
Top	int	False
Left	int	False
Name	string	False
Rotation	short	False
VisualizeQuality	bool	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
Process value	string	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
SelectionItemHeight	uint16	False
SelectionItems	HmiSelectionItemPartComposition	True
TabIndex	ushort	False
ToolTipText	MultilingualText	True
Visible	bool	False
Width	uint	False
Require Explicit Release	bool	False
Padding	HmiPaddingPart	False
SelectionPosition	HmiHorizontalAlignment	False

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit Element-Bildobjekten arbeiten (Seite 351)

Programmcode

Ändern Sie folgenden Programmcode, um auf die grundlegenden Eigenschaften eines Kontrollkästchens zuzugreifen:

```
//Name  
var name = chkbox.Name;  
chkbox.Name = "DefaultName";  
//AlternateBackColor  
var altbackcolor = chkbox.AlternateBackColor;  
chkbox.AlternateBackColor = Color.Beige;  
//Height  
var height = chkbox.Height;  
chkbox.Height = 100;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText  
var tooltip = chkbox.ToolTipText;  
var tooltiptext = chkbox.ToolTipText.Items[0].Text;  
chkbox.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Ändern Sie folgenden Programmcode, um auf eine andere typische Eigenschaft eines Kontrollkästchens zuzugreifen:

```
//Font  
var font = chkbox.Font;  
var italic = font.Italic;  
font.Italic = false;  
var fontname = font.FontName;  
font.FontName = HmiFontName.SimSun;  
var size = font.Size;  
font.Size = 10.2f;  
var stike = font.StrikeOut;  
font.StrikeOut = true;  
var underline = font.Underline;  
font.Underline = false;  
font.Bold = true;  
//Selection Items  
var selectionItem = chkbox.SelectionItems;  
var newselectionitem = selectionItem.Create("NewSelectionItem");  
var graphic = newselection.Graphic;  
newselectionitem.Graphic = "abcd";  
newselectionitem.Isselected = false;  
newselectionitem.Text.Item[0].Text = "Testformultilingual";
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit Element-Bildobjekten arbeiten \(Seite 351\)](#)

Auf Eigenschaften eines Balkens zugreifen

Einleitung

Mit TIA Openness API V16 können Sie Eigenschaften von Balken in Bildobjekten abrufen und einstellen.

Die folgenden Eigenschaften von Balken werden in Bildobjekten unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
AlternateBackgroundColor	Color	False
AlternateBorderColor	Color	False
Authorization	string	True
BackgroundColor	Color	False
BorderColor	Color	False
BarMode	HmiBarMode	False
BorderWidth	byte	False
CurrentQuality	HmiQuality	True
Enabled	bool	False
Font	HmiFontPart	False
Label	HmiTextPart	
Height	uint	False
StraightScale	HmiStraightScalePart	
Top	int	False
Left	int	False
Name	string	False
NormalRangeColor	Color	False
OriginValue	double	False
OutputFormat	string	False
PeakIndicators	HmiPeakIndicator	False
Process Value	string	False
ProcessValueIndicatorBackColor	Color	False
ProcessValueIndicatorForeColor	Color	False
ProcessValueIndicatorMode	HmiProcessIndicatorMode	False
RotationAngle	short	False
VisualizeQuality	bool	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
ScaleBackColor	uint16	False

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
ScaleForeColor	Color	False
ShowTrendIndicator	bool	False
Visible	bool	False
Width	uint	False
TabIndex	ushort	False
TrendIndicateColor	Color	False
ToolTipText	MultilingualText	False
Require Explicit Release	bool	False
Title	HmiTextPart	True

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit Element-Bildobjekten arbeiten (Seite 351)

Programmcode

Ändern Sie folgenden Programmcode, um auf die grundlegenden Eigenschaften eines Balkens zuzugreifen:

```
//Name
var name = bar.Name;
bar.Name = "Default Value";
//RotationCenterPlacement
var rotation = bar.RotationCenterPlacement;
bar.RotationCenterPlacement = HmiRotationCenterPlacement.AbsoluteToContainer;
//Width
var width = bar.Width;
bar.Width = 100;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText
var tooltip = bar.ToolTipText;
var tooltiptext = bar.ToolTipText.Items[0].Text;
bar.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Ändern Sie folgenden Programmcode, um auf eine andere typische Eigenschaft eines Balkens zuzugreifen:

```
//Font
var title = bar.Title;
title.Text.items[0].Text = "teststing";
title.Visible = false;
title.Forecolor = Color.Black;
var font = title.Font;
var italic = font.Italic;
font.Italic = false;
var fontname = font.FontName;
font.FontName = HmiFontName.SimSun;
var size = font.Size;
font.Size = 10.2f;
var stike = font.StrikeOut;
font.StrikeOut = true;
var underline = font.Underline;
font.Underline = false;
font.Bold = true;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit Element-Bildobjekten arbeiten \(Seite 351\)](#)

Auf Eigenschaften eines Zeigerinstruments zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften eines Zeigerinstruments in Bildobjekten abrufen und einstellen.

Die folgenden Eigenschaften eines Zeigerinstruments werden im Bildobjekt unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
AlternativebackgroundBackColor	Color	False
AlternativeBorderColor	Color	False
Authorization	string	True
BackgroundColor	Color	False
BorderColor	Color	False
BorderWidth	byte	False
RelativeToOrigin	bool	False
CurrentQuality	HmiQuality	True
Enabled	bool	False

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
Title	HmiTextPart	True
Font	HmiFontPart	False
Label	HmiTextPart	True
Height	uint	False
Top	int	False
Left	int	False
Name	string	False
NormalRangeColor	Color	False
OriginValue	double	False
OutputFormat	string	False
PeakIndicators	HmiPeakIndicator	False
Process Value	string	False
ProcessValueIndicatorBackColor	Color	False
ProcessValueIndicatorForeColor	Color	False
ProcessValueIndicatorMode	HmiProcessIndicatorMode	False
Rotation	short	False
VisualizeQuality	bool	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
ScaleBackgroundColor	Color	False
ScaleForegroundColor	Color	False
TrendIndicatorColor	Color	False
Visible	bool	False
Width	uint	False
TabIndex	ushort	False
ShowTrendIndicator	bool	False
ToolTipText	string	False
RequireExplixit	bool	False
CurvedScale	HmiCurvedScalePart	True

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit Element-Bildobjekten arbeiten (Seite 351)

Programmcode

Ändern Sie folgenden Programmcode, um auf die grundlegenden Eigenschaften eines Zeigerinstruments zuzugreifen:

```
//Name
var name = gauge.Name;
gauge.Name = "Default Value";
//RotationalCenterPlacement
var rotation = gauge.RotationCenterPlacement;
gauge.RotationCenterPlacement = HmiRotationCenterPlacement.AbsoluteToContainer;
//Width
var width = gauge.Width;
gauge.Width = 100;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText
var tooltip = gauge.ToolTipText;
var tooltiptext = gauge.ToolTipText.Items[0].Text;
gauge.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Ändern Sie folgenden Programmcode, um auf eine andere typische Eigenschaft eines Zeigerinstruments zuzugreifen:

```
//Font
var title = gauge.Title;
title.Text.items[0].Text = "teststing";
title.Visible = false;
title.Forecolor = Color.Black;
var font = title.Font;
var italic = font.Italic;
font.Italic = false;
var fontname = font.FontName;
font.FontName = HmiFontName.SimSun;
var size = font.Size;
font.Size = 10.2f;
var stike = font.StrikeOut;
font.StrikeOut = true;
var underline = font.Underline;
font.Underline = false;
font.Bold = true;
```

Siehe auch

- [Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- [Projekt öffnen \(Seite 111\)](#)
- [HMI-Unified-Software-Objekt \(Seite 281\)](#)
- [Mit Element-Bildobjekten arbeiten \(Seite 351\)](#)

Auf Eigenschaften eines Schiebereglers zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften von Schiebereglern in Bildobjekten abrufen und einstellen.

Die folgenden Eigenschaften von Schiebereglern werden in Bildobjekten unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
AlternativeBackColor	Color	False
AlternativeBorderColor	Color	False
Authorization	string	True
BackgroundColor	Color	False
BorderColor	Color	False
BorderWidth	byte	False
BarMode	HmiBarMode	False
RelativeToOrigin	bool	False
CurrentQuality	HmiQuality	True
Enabled	bool	False
Title	HmiTextPart	True
Font	HmiFontPart	False
Label	HmiTextPart	True
Height	uint	False
Top	int	False
Left	int	False
Name	string	False
NormalRangeColor	Color	False
OriginValue	double	False
OutputFormat	string	False
PeakIndicators	HmiPeakIndicator	False
Process Value	string	False
ProcessValueIndicatorBackColor	Color	False
ProcessValueIndicatorForeColor	Color	False
ProcessValueIndicatorMode	HmiProcessIndicatorMode	False
Rotation	short	False
VisualizeQuality	bool	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
ScaleBackgroundColor	Color	False
ScaleForegroundColor	Color	False
TrendIndicatorColor	bool	False
Visible	bool	False

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
Width	uint	False
ValuePosition	HmiSimplePosition	False
WriteDuringChange(Write process value Immediately)	bool	False
TabIndex	ushort	False
ShowTrendIndicator	bool	False
ToolTipText	string	False
ShowValue	bool	False
Require Explicit Release	bool	False
ThumbBackColor	color	False
ThumbForeColor	color	False
StraightScale	HmiStraightScalePart	True

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit Element-Bildobjekten arbeiten (Seite 351)

Programmcode

Ändern Sie folgenden Programmcode, um auf die grundlegenden Eigenschaften eines Schiebereglers zuzugreifen:

```
//Name
var name = slider.Name;
slider.Name = "Default Value";
//RotationalCenterPlacement
var rotation = slider.RotationCenterPlacement;
slider.RotationCenterPlacement = HmiRotationCenterPlacement.AbsoluteToContainer;
//Width
var width = slider.Width;
slider.Width = 100;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText
var tooltip = slider.ToolTipText;
var tooltiptext = slider.ToolTipText.Items[0].Text;
slider.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Ändern Sie folgenden Programmcode, um auf andere typische Eigenschaften eines Schiebereglers zuzugreifen:

```
//Font
var title = slider.Title;
title.Text.items[0].Text = "teststring";
title.Visible = false;
title.Forecolor = Color.Black;
var font = title.Font;
var italic = font.Italic;
font.Italic = false;
var fontname = font.FontName;
font.FontName = HmiFontName.SimSun;
var size = font.Size;
font.Size = 10.2f;
var stike = font.StrikeOut;
font.StrikeOut = true;
var underline = font.Underline;
font.Underline = false;
font.Bold = true;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit Element-Bildobjekten arbeiten \(Seite 351\)](#)

Auf Eigenschaften einer Optionsschaltfläche zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften von Optionsschaltflächen in Bildobjekten abrufen und einstellen.

Die folgenden Eigenschaften von Optionsschaltflächen werden in Bildobjekten unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
AlternateBackgroundColor	Color	False
AlternateBorderColor	Color	False
Authorization	string	True
BackgroundColor	Color	False
BorderColor	Color	False
BorderWidth	byte	False
Content	HmiContentPart	False
CurrentQuality	HmiQuality	True
Enabled	bool	False

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
Font	HmiFontPart	False
ForegroundColor	Color	False
Height	uint	False
Top	int	False
Left	int	False
Name	string	False
Process Value	string	False
Rotation	short	False
VisualizeQuality	bool	False
RotationCenterX	float	False
RotationCenterY	float	False
SelectionItemHeight	uint16	False
SelectionItems	IList<IHmiSelectionItemPart>	True
SelectorPosition	HmiHorizontalAlignment	
Opacity	float	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
Visible	bool	False
Width	uint	False
TabIndex	ushort	False
ToolTipText	string	False
Require Explicit Release	bool	False
Padding	HmiPaddingPart	True

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit Element-Bildobjekten arbeiten (Seite 351)

Programmcode

Ändern Sie folgenden Programmcode, um auf die grundlegenden Eigenschaften einer Optionsschaltfläche zuzugreifen:

```
//Name  
var name = radio.Name;  
radio.Name = "Default Value";  
//AlternateBackColor  
var altpbackcolor = radio.AlternateBackColor;  
radio.AlternateBackColor = Color.Beige;  
//Height  
var height = radio.Height;  
radio.Height = 100;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText  
var tooltip = radio.ToolTipText;  
var tooltiptext = radio.ToolTipText.Items[0].Text;  
radio.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Ändern Sie folgenden Programmcode, um auf andere typische Eigenschaften einer Optionsschaltfläche zuzugreifen:

```
//Font  
var font = radio.Font;  
var italic = font.Italic;  
font.Italic = false;  
var fontname = font.FontName;  
font.FontName = HmiFontName.SimSun;  
var size = font.Size;  
font.Size = 10.2f;  
var stike = font.StrikeOut;  
font.StrikeOut = true;  
var underline = font.Underline;  
font.Underline = false;  
font.Bold = true;  
//Selection Items  
var selectionItem = radio.SelectionItems;  
var newselectionitem = selectionItem.Create("NewSelectionItem");  
var graphic = newselectionitem.Graphic;  
newselectionitem.Graphic = "abcd";  
newselectionitem.IsSelected = false;  
newselectionitem.Text.Items[0].Text = "Testformultilingual";
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit Element-Bildobjekten arbeiten \(Seite 351\)](#)

Auf Eigenschaften eines Listenfelds zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften von Listenfeldern in Bildobjekten abrufen und einstellen.

Die folgenden Eigenschaften von Listenfeldern werden in Bildobjekten unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
AlternateBackgroundColor	Color	False
AlternateBorderColor	Color	False
Authorization	string	True
BackgroundColor	Color	False
BorderColor	Color	False
BorderWidth	byte	False
Content	HmiContentPart	False
CurrentQuality	HmiQuality	True
Enabled	bool	False
Font	HmiFontPart	False
ForeGroundColor	Color	False
Top	int	False
Left	int	False
Name	string	False
Process Value	string	False
Rotation	short	False
VisualizeQuality	bool	False
RotationCenterX	float	False
RotationCenterY	float	False
SelectionItemHeight	uint16	False
SelectionItems	IList<IHmiSelectionItemPart>	True
SelectionMode	HmiSelectionMode	False
SelectorPosition	HmiHorizontalAlignment	
Opacity	float	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
Visible	bool	False
Width	uint	False
TabIndex	ushort	False
ToolTipText	string	False
Require Explicit Release	bool	False
Padding	HmiPaddingPart	True

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit Element-Bildobjekten arbeiten (Seite 351)

Programmcode

Ändern Sie folgenden Programmcode, um auf die grundlegenden Eigenschaften eines Listenfelds zuzugreifen:

```
//Name
var name = listbox.Name;
listbox.Name = "Default Value";
//AlternateBackColor
var altpbackcolor = listbox.AlternateBackColor;
listbox.AlternateBackColor = Color.Beige;
//Height
var height = listbox.Height;
listbox.Height = 100;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText
var tooltip = listbox.ToolTipText;
var tooltiptext = listbox.ToolTipText.Items[0].Text;
listbox.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Ändern Sie folgenden Programmcode, um auf eine andere typische Eigenschaft eines Listenfelds zuzugreifen:

```
//Font
var font = listbox.Font;
var italic = font.Italic;
font.Italic = false;
var fontname = font.FontName;
font.FontName = HmiFontName.SimSun;
var size = font.Size;
font.Size = 10.2f;
var stike = font.StrikeOut;
font.StrikeOut = true;
var underline = font.Underline;
font.Underline = false;
font.Bold = true;
//Selection Items
var selectionItem = listbox.SelectionItems;
var newselectionitem = selectionItem.Create("NewSelectionItem");
var graphic = newselectionitem.Graphic;
newselectionitem.Graphic = "abcd";
newselectionitem.Isselected = false;
newselectionitem.Text.Item[0].Text = "Testformultilingual";
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit Element-Bildobjekten arbeiten \(Seite 351\)](#)

Auf Eigenschaften eines Textfelds zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften von Textfeldern in Bildobjekten abrufen und einstellen.

Die folgenden Eigenschaften von Textfeldern werden in Bildobjekten unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
ReadOnly	bool	False
TextWrapping	HmiTextWrapping	False
VerticalTextAlignment	HmiVerticalAlignment	False
HorizontalTextAlignment	HmiHorizontalAlignment	False
TextTrimming	HmiTextTrimming	False
ForeColor	Color	False
BorderColor	Color	False

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
AlternateBorderColor	Color	False
Authorization	string	False
BackColor	Color	False
BorderWidth	byte	False
CurrentQuality	HmiQuality	True
Enabled	bool	False
Font	HmiFontPart	False
Height	uint	False
Top	int	False
Left	int	False
Name	string	False
RotationAngle	short	False
VisualizeQuality	bool	False
AlternateBackColor	Color	False
RotationCenterX	float	False
RotationCenterY	float	False
Opacity	float	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
Visible	bool	False
Width	uint	False
TabIndex	ushort	False
ToolTipText	string	False
Require Explixit Release	bool	False
Text	MultilingualText	True
Padding	HmiPaddingPart	True

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit Element-Bildobjekten arbeiten (Seite 351)

Programmcode

Ändern Sie folgenden Programmcode, um auf eine grundlegende Eigenschaft eines Textfelds zuzugreifen:

```
//Name  
var name = textbox.Name;  
textbox.Name = "Default Value";  
//BorderColor  
var bordercolor = textbox.BorderColor;  
textbox.BorderColor = Color.FromArgb(0xCC, 0xF0, 0x80, 0x80);  
//BorderWidth  
var borderwidth = textbox.BorderWidth;  
textbox.BorderWidth = 10;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText  
var tooltip = textbox.ToolTipText;  
var tooltiptext = textbox.ToolTipText.Items[0].Text;  
textbox.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Ändern Sie folgenden Programmcode, um auf eine andere typische Eigenschaft eines Textfelds zuzugreifen:

```
//Font  
var font = textbox.Font;  
var italic = font.Italic;  
font.Italic = false;  
var fontname = font.FontName;  
font.FontName = HmiFontName.SimSun;  
var size = font.Size;  
font.Size = 10.2f;  
var stike = font.StrikeOut;  
font.StrikeOut = true;  
var underline = font.Underline;  
font.Underline = false;  
font.Bold = true;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit Element-Bildobjekten arbeiten \(Seite 351\)](#)

Auf Eigenschaften einer Uhr zugreifen:

Einleitung

Mit TIA Portal Openness können Sie Eigenschaften einer Uhr in einem Bildobjekt abrufen und einstellen.

Die folgenden Eigenschaften einer Uhr werden in einem Bildobjekt unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
AlternativeBackgroundColor	Color	False
AlternativeBorderColor	Color	False
BorderColor	Color	False
Authorization	string	True
BackgroundColor	Color	False
BorderWidth	byte	False
CurrentQuality	HmiQuality	True
DialBackColor	Color	False
DialLabelColor	Color	False
DialMode	HmiScaleMode	False
DialTickColor	Color	False
Enabled	bool	False
Height	uint	False
Top	int	False
Left	int	False
Name	string	False
VisualizeQuality	bool	False
RotationCenterX	float	False
RotationCenterY	float	False
ShowHours	bool	False
ShowMinutes	bool	False
ShowSeconds	bool	False
Opacity	float	False
TimeSource	string	False
Rotation	short	False
RotationCenterPlacement	HmiRotationCenterPlacement	False
Visibility	bool	False
Width	uint	False
TabIndex	ushort	False
Title	HmiTextPart	True
ToolTipText	MultiLingualText	False
Require Explixit	bool	False
DialLabelFont	HmiFontPart	True

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit Element-Bildobjekten arbeiten (Seite 351)

Programmcode

Ändern Sie folgenden Programmcode, um auf eine grundlegende Eigenschaft einer Uhr zuzugreifen:

```
//Name
var name = clock.Name;
clock.Name = "Default Value";
//AlternateBackColor
var altbackcolor = clock.AlternateBackColor;
clock.AlternateBackColor = Color.Beige;
//Height
var height = clock.Height;
clock.Height = 100;
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//ToolTipText
var tooltip = clock.ToolTipText;
var tooltiptext = clock.ToolTipText.Items[0].Text;
clock.ToolTipText.Items[0].Text = "TestforMultilingualProperty";
```

Ändern Sie folgenden Programmcode, um auf andere typische Eigenschaften zuzugreifen:

```
//DialLabelFont
var dialfont = clock.DialLabelFont;
dialfont.Italic = false;
var fontname = dialfont.FontName;
dialfont.FontName = HmiFontName.SimSun;
var size = dialfont.Size;
dialfont.Size = 10.2f;
var stike = dialfont.StrikeOut;
dialfont.StrikeOut = true;
var underline = dialfont.Underline;
dialfont.Underline = false;
dialfont.Bold = true;
//ShowHours
var showhour = clock.ShowHours;
clock.ShowHours = false;
```

Hinweis

Wenn während des Festlegens von Eigenschaften der Wert durch den Schreibvorgang nicht festgelegt werden kann, wird eine Recoverable exception ausgelöst. Wenn Werte außerhalb des zulässigen Bereichs nicht festgelegt werden, wird eine Recoverable-Ausnahme ausgelöst. Wenn diese Werte festgelegt werden, wird die Eigenschaft in einen ungültigen Zustand versetzt.

Siehe auch

- [Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- [Projekt öffnen \(Seite 111\)](#)
- [HMI-Unified-Software-Objekt \(Seite 281\)](#)
- [Mit Element-Bildobjekten arbeiten \(Seite 351\)](#)

7.18.9.4 Steuerbildobjekte

Mit Steuerbildobjekten arbeiten

Einleitung

Mit Steuerbildobjekten können Sie folgende Aktionen durchführen, während Sie TIA Portal Openness verwenden:

- Bildobjekte erstellen (Controls)
- Bildobjekte löschen (Controls)
- Bildobjekte enumerieren (Controls)

Die folgenden Bildobjekte werden von TIA Portal Openness unterstützt. Verwenden Sie für den Zugriff auf die Klassen und Datentypen im Zusammenhang mit Bildobjekten die folgenden Namensräume und Datentypen:

Tabelle 7-3 Typen und Namensraum/-räume

Bildobjekt	Klasse	Namensraum/-räume
Meldeanzeige	HmiAlarmControl	Siemens.Engineering.HmiUnified.ModernUI.Controls
Media Player	HmiMediaControl	Siemens.Engineering.HmiUnified.ModernUI.Base
Bildfenster	HmiScreenWindow	
Kurvenanzeige	HmiTrendControl	
Wertetabelle	HmiTrendCompanion	
Tabellenanzeige	HmiProcessControl	
f(x)-Kurvenanzeige	HmiFunctionTrendControl	
Browser	HmiWebControl	
Parametersatzanzeige	HmiDetailedParameterControl	

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)

Programmcode

Bei der Arbeit mit Steuerbildobjekten können Sie den folgenden Beispiel-Programmcode ändern und verwenden.

Bildobjekte erstellen (Controls)

Ändern Sie folgenden Programmcode, um Steuerbildobjekte zu erstellen:

```
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiAlarmControl hmiAlarmControl = screenitems.Create<HmiAlarmControl>("ScreenItem_1");
```

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Recoverable Exception wird ausgelöst, wenn die Methode Create mit dem Namen eines Bildobjekts aufgerufen wird, das in TIA bereits vorhanden ist.

Hinweis

Das obige Beispiel gilt für alle in der obigen Typentabelle definierten Bildobjekte. Die HmiAlarmControl muss mit einem Typ/mit Typen definiert werden, der/die in der Tabelle "Typen und Namensraum/-räume" definiert ist/sind.

Bildobjekte löschen (Controls)

Ändern Sie folgenden Programmcode, um Steuerbildobjekte zu löschen:

```
public void Delete()
//Case 1
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiAlarmControl hmiAlarmControl = screenitems.Create<HmiAlarmControl>("ScreenItem_1");
if (hmiAlarmControl != null)
{
    hmiAlarmControl.Delete();
}
//Case 2
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiAlarmControl hmiAlarmControl = screenitems.Create<HmiAlarmControl>("ScreenItem_1");
IEngineeringObject ObjhmiAlarmControlEnggObj = hmiAlarmControl;
if (ObjhmiAlarmControlEnggObj != null)
{
    ObjhmiAlarmControlEnggObj.Invoke("Delete", null);
}
```

Hinweis

Das obige Beispiel gilt für alle in der obigen Typentabelle definierten Bildobjekte. Die HmiAlarmControl muss mit einem Typ/mit Typen definiert werden, der/die in der Tabelle "Typen und Namensraum/-räume" definiert ist/sind.

Bildobjekte enumerieren (Control)

Ändern Sie folgenden Programmcode, um alle Bildobjekte eines Bildes zu enumerieren:

```
HmiSoftware hmiSoftware = GetHMISoftware();
HmiScreen hmiScreen = hmiSoftware.Screens.Create("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
foreach (var item in screenitems)
{
    //work with screenitems
}
```

Ändern Sie folgenden Programmcode, um Bildobjekte in einer Bildobjektliste anhand des Namens zu suchen:

```
HmiScreenItemBase screenitems = hmiSoftware.Screens[0].ScreenItems.Find("ScreenItems_1");
```

Ändern Sie folgenden Programmcode, um Bildobjekte in einer Bildobjektliste anhand des Index zu finden:

```
HmiScreenItemBase screenitem = hmiSoftware.Screens[0].ScreenItems[0];
```

Ändern Sie folgenden Programmcode, um mit der Methode Contains zu ermitteln, ob in einer Liste von Bildobjekten ein bestimmtes Bildobjekt vorhanden ist:

```
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiAlarmControl hmiAlarmControl = screenitems.Create<HmiAlarmControl>("ScreenItems_1");
bool isexists = screenitems.Contains(hmiAlarmControl);
```

Hinweis

Das obige Beispiel gilt für alle in der obigen Typentabelle definierten Bildobjekte. Das HmiAlarmControl muss mit einem Typ/mit Typen definiert werden, der/die in der Tabelle "Typen und Namensraum/-räume" definiert ist/sind.

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

Auf Eigenschaften eines Bildfensters zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften eines Bildfensters in Steuerbildobjekten abrufen und einstellen.

Die folgenden Eigenschaften eines Bildfensters werden in Steuerbildobjekten unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
TabIntoWindow	bool	False
Screen	string	False
ScreenName	string	True
ScreenNumber	byte	True
System	byte	False
CurrentZoomFactor	double	False

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
VerticalScrollBarVisibility	HmiScrollBarVisibility	False
VerticalScrollBarPosition	int	False
HorizontalScrollBarVisibility	HmiScrollBarVisibility	False
HorizontalScrollBarPosition	int	False
Adaption	HmiScreenWindowAdaption	False
InteractiveZooming	bool	True
Caption	MultilingualProperty	False
WindowFlags	HmiWindowFlag	False
Icon	string	False
Top	int	False
Left	int	False
Width	uint	False
Height	uint	False
CurrentQuality	HmiQuality	True
RequireExplicitRelease	bool	False
Authorization	string	False
Name	string	False
Visible	bool	False
Enabled	bool	False
TabIndex	ushort	False

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit Steuerbildobjekten arbeiten (Seite 380)

Programmcode

Ändern Sie folgenden Programmcode, um auf die grundlegenden Eigenschaften eines Bildfensters zuzugreifen:

```
//Width
var autoplay = hmiscreenwindow.Width;
hmiscreenwindow.Width = 1;
//HorizontalScrollBarVisibility
var horizontalscrollbarvisibilityValue = hmiscreenwindow.HorizontalScrollBarVisibility;
hmiscreenwindow.HorizontalScrollBarVisibility = HmiScrollBarVisibility.Visible;
//Name
var name = hmiscreenwindow.Name;
hmiscreenwindow.Name = "DefaultName";
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//Caption
var caption = hmiscreenwindow.Caption.Items[0].Text;
hmiscreenwindow.Caption.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit Steuerbildobjekten arbeiten \(Seite 380\)](#)

Auf Eigenschaften eines Browsers zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften eines Browsers in Steuerbildobjekten abrufen und einstellen.

Die folgenden Eigenschaften eines Browsers werden in Steuerbildobjekten unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
Url	string	False
BackColor	Color	False
Caption	MultilingualProperty	False
WindowFlags	HmiWindowFlag	False
Icon	string	False
Top	Int	False
Left	int	False
Width	uint	False
Height	uint	False

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
CurrentQuality	HmiQuality	False
RequireExplicitRelease	bool	False
Authorization	string	False
Name	string	False
Visible	bool	False
Enabled	bool	False
TabIndex	ushort	False
ToolBar	HmiToolBarPart	True
StatusBar	HmiStatusBarPart	True

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit Steuerbildobjekten arbeiten (Seite 380)

Programmcode

Ändern Sie folgenden Programmcode, um auf die grundlegenden Eigenschaften eines Browsers zuzugreifen:

```
//Width
var autoplay = hmiwebcontrol.width;
hmiwebcontrol.Width = 1;
//BackColor
var backcolor = hmiwebcontrol.BackColor;
hmiwebcontrol.BackColor = Color.Beige;
//Name
var name = hmiwebcontrol.Name;
hmiwebcontrol.Name = "DefaultName";
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//Caption
var caption = hmiwebcontrol.Caption.Items[0].Text;
hmiwebcontrol.Caption.Items[0].Text= "<body><p>TestforMultilingualProperty</p></body>";
```

Ändern Sie folgenden Programmcode, um auf andere typische Eigenschaften zuzugreifen:

```
//Status bar
var statusBar = hmiwebcontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit Steuerbildobjekten arbeiten \(Seite 380\)](#)

Auf Eigenschaften einer Parametersatzanzeige zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften einer Parametersatzanzeige in Steuerbildobjekten abrufen und einstellen.

Die folgenden Eigenschaften einer Parametersatzanzeige werden in Steuerbildobjekten unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
ParameterSetTypeFixed	string	False
BackColor	Color	False
Caption	MultilingualProperty	False
WindowFlags	HmiWindowFlag	False
Icon	string	False
Top	int	False
Left	int	False
Width	uint	False
Height	uint	False
CurrentQuality	HmiQuality	True
RequireExplicitRelease	bool	False
Authorization	string	False
Name	string	False
Visible	bool	False
Enabled	bool	False
TabIndex	ushort	False

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
ToolBar	HmiToolBarPart	True
StatusBar	HmiStatusBarPart	True
EditMode	HmiEditMode	False
TimeZone	int	False
ParameterView	HmiDataGridViewPart	True
ForeColor	Color	False
SelectionBackColor	Color	False
SelectionForeColor	Color	False
Font	HmiFontPart	True

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit Steuerbildobjekten arbeiten (Seite 380)

Programmcode

Ändern Sie folgenden Code, um auf die grundlegenden Eigenschaften einer Parametersatzanzeige zuzugreifen:

```
//Width
var autoplay = hmidedetailedparametercontrol.Width;
hmidedetailedparametercontrol.Width = 1;
//BackColor
var backcolor = hmidedetailedparametercontrol.BackColor;
hmidedetailedparametercontrol.BackColor = Color.Beige;
//Name
var name = hmidedetailedparametercontrol.Name;
hmidedetailedparametercontrol.Name = "Default";
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//Caption
var caption = hmidedetailedparametercontrol.Caption.Items[0].Text;
hmidedetailedparametercontrol.Caption.Items[0].Text=
"<body><p>TestforMultilingualProperty</p></body>";
```

Ändern Sie folgenden Programmcode, um auf eine andere typische Eigenschaft zuzugreifen:

```
//Status bar
var statusBar = hmidetailedparametercontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit Steuerbildobjekten arbeiten \(Seite 380\)](#)

Auf Eigenschaften einer Meldeanzeige zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften einer Meldeanzeige in Steuerbildobjekten abrufen und einstellen.

Die folgenden Eigenschaften einer Meldeanzeige werden in Steuerbildobjekten unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
UseAlarmColors	bool	False
Filter	string	False
AlwaysShowRecent	bool	False
AlarmSourceType	HmiAlarmSourceType	False
Caption	MultilingualProperty	False
WindowFlags	HmiWindowFlag	False
Icon	string	False
Top	Int	False
Left	int	False
Width	uint	False
Height	uint	False
CurrentQuality	HmiQuality	True
RequireExplicitRelease	bool	False
Authorization	string	False
Name	string	False
Visible	bool	False
Enabled	bool	False

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
TabIndex	ushort	False
ToolBar	HmiToolBarPart	True
StatusBar	HmiStatusBarPart	True
EditMode	HmiEditMode	False
TimeZone	int	False
AlarmDefinitionViewSetup	HmiVisibleAlarms	False
ActiveAlarmsViewSetup	HmiVisibleAlarms	False
SuppressFlashing	bool	False
AcknowledgmentFlashingRate	HmiFlashingRate	False
ResetFlashingRate	HmiFlashingRate	False
DefaultSortDirection	HmiSortDirection	False
BackColor	Color	False
AlarmView	HmiDataGridViewPart	True

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit Steuerbildobjekten arbeiten (Seite 380)

Programmcode

Ändern Sie folgenden Programmcode, um auf die grundlegenden Eigenschaften einer Meldeanzeige zuzugreifen:

```
//Width
var autoplay = hmialarmcontrol.Width;
hmialarmcontrol.Width = 1;
//BackColor
var backcolor = hmialarmcontrol.BackColor;
hmialarmcontrol.BackColor = Color.Beige;
//Name
var name = hmialarmcontrol.Name;
hmialarmcontrol.Name = "DefaultName";
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//Caption
var caption = hmialarmcontrol.Caption.Items[0].Text;
hmialarmcontrol.Caption.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
```

Ändern Sie folgenden Programmcode, um auf andere typische Eigenschaften zuzugreifen:

```
//Status bar
var statusBar = hmialarmcontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit Steuerbildobjekten arbeiten \(Seite 380\)](#)

Auf Eigenschaften einer Wertetabelle zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften einer Wertetabelle in Steuerbildobjekten abrufen und einstellen.

Die folgenden Eigenschaften einer Wertetabelle werden in Steuerbildobjekten unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
TrendCompanionMode	HmiTrendCompanionMode	False
UseSourceControlTrendColors	bool	False
ShowAlways	bool	False
UseSourceControlBackColor	bool	False
Caption	MultilingualProperty	False
WindowFlags	HmiWindowFlag	False
Icon	string	False
Top	int	False
Left	int	False
Width	uint	False
Height	uint	False
CurrentQuality	HmiQuality	True
RequireExplicitRelease	bool	False
Authorization	string	False
Name	string	False
Visible	bool	False
Enabled	bool	False

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
TabIndex	ushort	False
ToolBar	HmiToolBarPart	True
StatusBar	HmiStatusBarPart	True
SourceTrendControl	string	False
TimeZone	int	False
SnapToSourceControl	bool	False
TrendRulerView	HmiDataGridViewPart	True
TrendStatisticAreaView	HmiDataGridViewPart	True
BackColor	Color	False
TrendStatisticResultView	HmiDataGridViewPart	True

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit Steuerbildobjekten arbeiten (Seite 380)

Programmcode

Ändern Sie folgenden Programmcode, um auf die grundlegenden Eigenschaften einer Wertetabelle zuzugreifen:

```
//Width
var autoplay = hmitrendcompanion.Width;
hmitrendcompanion.Width = 1;
//BackColor
var backcolor = hmitrendcompanion.BackColor;
hmitrendcompanion.BackColor = Color.Beige;
//Name
var name = hmitrendcompanion.Name;
hmitrendcompanion.Name = "DefaultName";
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//Caption
var caption = hmitrendcompanion.Caption.Items[0].Text;
hmitrendcompanion.Caption.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
```

Ändern Sie folgenden Programmcode, um auf andere typische Eigenschaften einer Wertetabelle zuzugreifen:

```
//Status bar
var statusBar = hmitrendcompanion.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit Steuerbildobjekten arbeiten \(Seite 380\)](#)

Auf Eigenschaften einer Kurvenanzeige zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften einer Kurvenanzeige in Steuerbildobjekten abrufen und einstellen.

Die folgenden Eigenschaften einer Kurvenanzeige werden in Steuerbildobjekten unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
ShowStatisticRulers	bool	False
AreaSpacing	ushort	False
Online	bool	False
ExtendRulerToAxis	bool	False
Caption	MultilingualProperty	False
WindowFlags	HmiWindowFlag	False
Icon	string	False
Top	int	False
Left	int	False
Width	uint	False
Height	uint	False
CurrentQuality	HmiQuality	True
RequireExplicitRelease	bool	False
Authorization	string	False
Name	string	False
Visible	bool	False

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
Enabled	bool	False
TabIndex	ushort	False
ToolBar	HmiToolBarPart	True
StatusBar	HmiStatusBarPart	True
Font	HmiFontPart	True
TimeZone	int	False
ShowRuler	bool	False
ShiftAxes	bool	False
BackColor	Color	False
Legend	HmiLegendPart	True
ForeColor	Color	False

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit Steuerbildobjekten arbeiten (Seite 380)

Programmcode

Ändern Sie folgenden Programmcode, um auf die grundlegenden Eigenschaften einer Kurvenanzeige zuzugreifen:

```
//Width
var autoplay = hmitrendcontrol.Width;
hmitrendcontrol.Width = 1;
//BackColor
var backcolor = hmitrendcontrol.BackColor;
hmitrendcontrol.BackColor = Color.Beige;
//Name
var name = hmitrendcontrol.Name;
hmitrendcontrol.Name = "DefaultName";
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//Caption
var caption = hmitrendcontrol.Caption.Items[0].Text;
hmitrendcontrol.Caption.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
```

Ändern Sie folgenden Programmcode, um auf andere typische Eigenschaften zuzugreifen:

```
//Status bar
var statusBar = hmitrendcontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit Steuerbildobjekten arbeiten \(Seite 380\)](#)

Auf Eigenschaften eines Media Players zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften eines Media Players in Steuerbildobjekten abrufen und einstellen.

Die folgenden Eigenschaften eines Media Players werden in Steuerbildobjekten unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
Url	string	False
AutoPlay	bool	False
VideoOutput	HmiVideoOutput	False
BackColor	Color	False
Caption	MultilingualProperty	False
WindowFlags	HmiWindowFlag	False
Icon	string	False
Top	int	False
Left	int	False
Width	uint	False
Height	uint	False
CurrentQuality	HmiQuality	True
RequireExplicitRelease	bool	False
Authorization	string	False
Name	string	False
Visible	bool	False
Enabled	bool	False

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
TabIndex	ushort	False
ToolBar	HmiToolBarPart	True
StatusBar	HmiStatusBarPart	True
EventHandlers	HmiMediaControlEventHandlerComposition	True

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit Steuerbildobjekten arbeiten (Seite 380)

Programmcode

Ändern Sie folgenden Programmcode, um auf die grundlegenden Eigenschaften eines Media Players zuzugreifen:

```
//Autoplay
var autoplay = hmimediicontrol.AutoPlay;
hmimediicontrol.AutoPlay = true;
//BackColor
var backcolor = hmimediicontrol.BackColor;
hmimediicontrol.BackColor = Color.Beige;
//Name
var name = hmimediicontrol.Name;
hmimediicontrol.Name = "DefaultName";
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//Caption
var caption = hmimediicontrol.Caption.Items[0].Text;
hmimediicontrol.Caption.Items[0].Text = "<body><p>TestforMultilingualProperty</p></body>";
```

Ändern Sie folgenden Programmcode, um auf eine andere typische Eigenschaft zuzugreifen:

```
var statusBar = hmimediacontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit Steuerbildobjekten arbeiten \(Seite 380\)](#)

Auf Eigenschaften einer Tabellenanzeige zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften einer Tabellenanzeige in Steuerbildobjekten abrufen und einstellen.

Die folgenden Eigenschaften einer Tabellenanzeige werden in Steuerbildobjekten unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
Online	bool	False
TimeStepSmoothingBase	HmiTimeRangeBase	False
TimeStepSmoothingFactor	int	False
BackColor	Color	False
Caption	MultilingualProperty	False
WindowFlags	HmiWindowFlag	False
Icon	string	False
Top	int	False
Left	int	False
Width	uint	False
Height	uint	False
CurrentQuality	HmiQuality	True
RequireExplicitRelease	bool	False
Authorization	string	False
Name	string	False
Visible	bool	False
Enabled	bool	False
TabIndex	ushort	False

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
ToolBar	HmiToolBarPart	True
StatusBar	HmiStatusBarPart	True
EditMode	HmiEditMode	False
TimeZone	int	False
ProcessView	HmiDataGridViewPart	True
EventHandlers	HmiProcessControlEventHandlerComposition	True

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit Steuerbildobjekten arbeiten (Seite 380)

Programmcode

Ändern Sie folgenden Programmcode, um auf die grundlegenden Eigenschaften einer Tabellenanzeige zuzugreifen:

```
//Width
var autoplay = hmiprocesscontrol.Width;
hmiprocesscontrol.Width = 1;
//BackColor
var backcolor = hmiprocesscontrol.BackColor;
hmiprocesscontrol.BackColor = Color.Beige;
//Name
var name = hmiprocesscontrol.Name;
hmiprocesscontrol.Name = "DefaultName";
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//Caption
var caption = hmiprocesscontrol.Caption.Items[0].Text;
hmiprocesscontrol.Caption.Items[0].Text= "<body><p>TestforMultilingualProperty</p></body>";
```

Ändern Sie folgenden Programmcode, um auf andere typische Eigenschaften zuzugreifen:

```
//status bar
var statusBar = hmiprocesscontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Mit Steuerbildobjekten arbeiten \(Seite 380\)](#)

Auf Eigenschaften einer f(x)-Kurvenanzeige zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften einer f(x)-Kurvenanzeige in Steuerbildobjekten abrufen und einstellen.

Die folgenden Eigenschaften einer f(x)-Kurvenanzeige werden in Steuerbildobjekten unterstützt:

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
ShowStatisticRulers	bool	False
AreaSpacing	ushort	
Online	bool	
ExtendRulerToAxis	bool	
Caption	MultilingualProperty	False
WindowFlags	HmiWindowFlag	False
Icon	string	False
Top	Int	False
Left	int	False
Width	uint	False
Height	uint	False
CurrentQuality	HmiQuality	True
RequireExplicitRelease	bool	False
Authorization	string	False

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Name der Eigenschaft	Eigenschaftstyp	Zugänglichkeit
Name	string	False
Visible	bool	False
Enabled	bool	False
TabIndex	ushort	False
ToolBar	HmiToolBarPart	True
StatusBar	HmiStatusBarPart	True
Font	HmiFontPart	True
TimeZone	int	False
ShowRuler	bool	False
ShiftAxes	bool	False
BackColor	Color	False
Legend	HmiLegendPart	True
EventHandlers	HmiFunctionTrendControl EventHandlerComposition	True
FunctionTrendAreas	HmiFunctionTrendAreaPartComposition	True

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Bild und Bildobjekt wird erstellt
Siehe Mit Steuerbildobjekten arbeiten (Seite 380)

Programmcode

Ändern Sie folgenden Programmcode, um auf die grundlegenden Eigenschaften einer f(x)-Kurvenanzeige zuzugreifen:

```
//Width
var autoplay = hmifunctiontrendcontrol.Width;
hmifunctiontrendcontrol.Width = 1;
//BackColor
var backcolor = hmifunctiontrendcontrol.BackColor;
hmifunctiontrendcontrol.BackColor = Color.Beige;
//Name
var name = hmifunctiontrendcontrol.Name;
hmifunctiontrendcontrol.Name = "DefaultName";
```

Ändern Sie folgenden Programmcode, um auf eine mehrsprachige Eigenschaft zuzugreifen:

```
//Caption
var caption = hmifunctiontrendcontrol.Caption.Items[0].Text;
hmifunctiontrendcontrol.Caption.Items[0].Text =
"<body><p>TestforMultilingualProperty</p></body>";
```

Ändern Sie folgenden Programmcode, um auf andere typische Eigenschaften einer f(x)-Kurvenanzeige zuzugreifen:

```
//Status bar
var statusBar = hmifunctiontrendcontrol.StatusBar;
var backColor = statusBar.BackColor;
statusBar.BackColor = Color.Aqua;
var enabled = statusBar.Enabled;
statusBar.Enabled = true;
var visible = statusBar.Visible;
statusBar.Visible = true;
```

Programmcode: TrendArea in FunctionTrendControl using part

Ändern Sie folgenden Programmcode, um in einem HmiFunctionTrendControl, das Part-Objekte verwendet, einen Kurvenbereich zu erstellen:

```
HmiSoftware hmiSoftware = GetHmiSoftware();
var screen = hmiSoftware.Screens;
var createdscreen = screen.Create("TestScreen_1");
HmiFunctionTrendControl funtntrend =
hmiSoftware.Screens[0].ScreenItems.Create<HmiFunctionTrendControl>("CTrendControl_1");
HmiFunctionTrendAreaPart part = funtntrend.FunctionTrendAreas.Create("part1");
```

Ändern Sie folgenden Programmcode, um in einem HmiFunctionTrendControl, das Part-Objekte verwendet, einen Kurvenbereich zu suchen:

```
HmiFunctionTrendAreaPart part = funtntrend.FunctionTrendAreas.Find("part1");
```

Ändern Sie folgenden Programmcode, um in einem HmiFunctionTrendControl, das Part-Objekte verwendet, einen Kurvenbereich zu prüfen:

```
HmiSoftware hmiSoftware = GetHmiSoftware();
HmiFunctionTrendControl funtntrend =
hmiSoftware.Screens[0].ScreenItems.Create<HmiFunctionTrendControl>("CTrendControl_1");
HmiFunctionTrendAreaPart part = funtntrend.FunctionTrendAreas.Create("part1");
bool bPresent = funtntrend.FunctionTrendAreas.Contains(part);
```

Siehe auch

- Verbindung zum TIA Portal aufbauen (Seite 79)
- Projekt öffnen (Seite 111)
- Mit Steuerbildobjekten arbeiten (Seite 380)
- HMI-Unified-Software-Objekt (Seite 281)

7.18.9.5 Bildbausteininstanz-Bildobjekte

Mit Bildbausteininstanzen in Steuerbildobjekten arbeiten

Einleitung

Mit Bildbausteininstanzen können Sie folgende Aktionen durchführen, während Sie TIA Portal Openness verwenden:

- Bildbausteininstanzen erstellen (Controls)
- Bildbausteininstanzen löschen (Controls)
- Bildbausteininstanzen enumerieren (Controls)

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)
- Anlagenobjekttyp muss erstellt werden

Programmcode

Bei der Arbeit mit Bildbausteininstanzen können Sie den folgenden Beispiel-Programmcode ändern und verwenden.

Bildbausteininstanzen erstellen

Ändern Sie folgenden Programmcode, um Bildbausteininstanz-Bildobjekte zu erstellen:

```
HmiSoftware hmiSoftware = GetHMISoftware ();
HmiScreen hmiScreen= hmiSoftware.Screens.Create ("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens [0].ScreenItems;
HmiFaceplateContainer faceplate = hmiScreen.ScreenItems.Create<HmiFaceplateContainer>
("abcd");
```

Recoverable Exception wird ausgelöst, wenn die Methode Create mit einem Bildnamen aufgerufen wird, zu dem bereits ein Bild im TIA Portal vorhanden ist.

Bildbausteininstanzen löschen

Ändern Sie folgenden Programmcode, um Bildbausteininstanz-Bildobjekte zu löschen:

```
//Case 1
HmiSoftware hmiSoftware = GetHMISoftware ();
HmiScreen hmiScreen = hmiSoftware.Screens.Create ("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens [0].ScreenItems;
HmiFaceplateContainer faceplate = hmiScreen .ScreenItems.Create<HmiFaceplateContainer>
("abcd");
If (faceplate! = null)
{
faceplate.Delete ();
}
//Case 2
HmiSoftware hmiSoftware = GetHMISoftware ();
HmiScreen hmiScreen = hmiSoftware.Screens.Create ("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens [0].ScreenItems;
HmiFaceplateContainer faceplate = hmiScreen .ScreenItems.Create<HmiFaceplateContainer>
("abcd");
IEngineeringObject ObjhmiLineEnggObj = faceplate;
if (ObjhmiLineEnggObj != null)
{
ObjhmiLineEnggObj.Invoke ("Delete", null);
}
```

Bildbausteininstanzen enumerieren

Ändern Sie folgenden Programmcode, um Bildbausteininstanz-Bildobjekte eines Bildes zu enumerieren:

```
HmiSoftware hmiSoftware = GetHMISoftware ();
HmiScreen hmiScreen = hmiSoftware.Screens.Create ("Screen_1");
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens [0].ScreenItems;
foreach (var item in screenitems)
{
//work with screenitems
}
```

Ändern Sie folgenden Programmcode, um Bildbausteininstanzen in einer Bildobjektliste anhand des Namens zu suchen:

```
HmiScreenItemBase screenitems = hmiSoftware.Screens [0].ScreenItems.Find ("ScreenItems_1");
```

Ändern Sie folgenden Programmcode, um Bildbausteininstanzen in einer Bildobjektliste anhand des Index zu finden:

```
HmiScreenItemBase screenitem = hmiSoftware.Screens [0].ScreenItems [0];
```

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Ändern Sie folgenden Programmcode, um mit Contains zu ermitteln, ob in einer Liste von Bildobjekten eine bestimmte Bildbausteininstanz vorhanden ist:

```
HmiScreenItemBaseComposition screenitems = hmiSoftware.Screens[0].ScreenItems;
HmiFaceplateContainer faceplate = screenitems .Create<HmiFaceplateContainer> ("abcd");
bool isexists = screenitems.Contains(faceplate);
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

[Projekt öffnen \(Seite 111\)](#)

Auf Eigenschaften einer Bildbausteininstanz zugreifen

Einleitung

Mit TIA Portal Openness können Sie die Eigenschaften einer Bildbausteininstanz in Steuerbildobjekten abrufen und einstellen.

Die folgenden Eigenschaften einer Bildbausteininstanz werden in Steuerbildobjekten unterstützt:

Name der Eigenschaft	Datentyp	Beschreibung	Zugänglichkeit
Name	System.String	Name der Bildbausteininstanz	R/W
ContainedType	System.String	Gibt den Typ des enthaltenen Bildbausteins an	R/W
Authorization	System.String	Autorisierungsinformationen für die Bildbausteininstanz	R/W
Enabled	System.Boolean	Bedienung zulassen	R/W
CurrentQuality	Siemens.Engineering.HmiUnified.UI.Enum.HmiQuality	Verbindungsstatus	R
Height	System.UInt32	Gibt die Höhe des Steuerfensters an	R/W
Icon	System.String	Gibt das Symbol im Steuerfenster an	R/W
Caption	Siemens.Engineering.MultilingualText	Im Titel eines Bildfensters oder Fensterbedienelements (Label) anzuzeigender Text	R
RequireExplicitRelease	System.Boolean	Wenn auf "True" gesetzt, wird das im Bild (oder in einem übergeordneten Bild, falls nicht lokal konfiguriert) konfigurierte Bildobjekt erst beim Loslassen der gedrückten Schaltfläche aktiviert	R/W

Name der Eigenschaft	Datentyp	Beschreibung	Zugänglichkeit
TabIndex	System.UInt16	Bildobjekte, die einen Tab-Index 0 vorgeben, sind nicht Teil der Tab-Reihenfolge	R/W
Visible	System.Boolean	Gibt die Sichtbarkeit eines Bildobjekts an	R/W
Width	System.UInt32	Gibt die Breite eines Steuerfens-ters an	R/W
WindowFlags	Siemens.Engineering.miUnified.UI.Enum.WindowFlag	Gibt die Fensterkonfiguration an, wie ShowCaption, ShowBorder, AlwaysOnTop.	R/W
Top	System.Int32	Gibt den Wert der Y-Koordinaten des Steuerfensters an	R/W
Left	System.Int32	Gibt den Wert der X-Koordinaten des Steuerfensters an	R/W
Interface	Siemens.Engineering.HmiUnified.UI.Parts.HmiFaceplateInterfaceComposition	Schnittstellen für Bildbausteine	R

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)

Programmcode

Ändern Sie folgenden Programmcode, um die Eigenschaften der Bildbausteininstanz abzurufen:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
HmiFaceplateContainer faceplate = screens[0].ScreenItems[0] as HmiFaceplateContainer;
//Get_Caption
string Caption = faceplate.Caption.Items[0].Text;
Console.WriteLine("Faceplate Caption: " + Caption);
//Get_WindowFlags
HmiWindowFlag WindowFlags = faceplate.WindowFlags;
Console.WriteLine("Faceplate WindowFlags: " + WindowFlags);
//Get_Icon
string Icon = faceplate.Icon;
Console.WriteLine("Faceplate Icon: " + Icon);
//Get_Top
int Top = faceplate.Top;
Console.WriteLine("Faceplate Top: " + Top);
//Get_Left
int Left = faceplate.Left;
Console.WriteLine("Faceplate Left: " + Left);
//Get_Width
uint Width = faceplate.Width;
Console.WriteLine("Faceplate Width: " + Width);
//Get_Height
uint Height = faceplate.Height;
Console.WriteLine("Faceplate Height: " + Height);
//Get_CurrentQuality
HmiQuality CurrentQuality = faceplate.CurrentQuality;
Console.WriteLine("Faceplate CurrentQuality: " + CurrentQuality);
//Get_RequireExplicitRelease
bool RequireExplicitRelease = faceplate.RequireExplicitRelease;
Console.WriteLine("Faceplate RequireExplicitRelease: " + RequireExplicitRelease);
//Get_Authorization
string Authorization = faceplate.Authorization;
Console.WriteLine("Faceplate Authorization: " + Authorization);
//Get_Name
string Name = faceplate.Name;
Console.WriteLine("Faceplate Name : " + Name);
//Get_Visible
bool Visible = faceplate.Visible;
Console.WriteLine("Faceplate Visible: " + Visible);
//Get_Enabled
bool Enabled = faceplate.Enabled;
Console.WriteLine("Faceplate Enabled: " + Enabled);
//Get_TabIndex
ushort TabIndex = faceplate.TabIndex;
Console.WriteLine("Faceplate TabIndex: " + TabIndex);
```

Ändern Sie den folgenden Programmcode, um die Eigenschaften der Bildbausteininstanz einzustellen:

```
//Set_Caption:  
faceplate.Caption.Items[0].Text = "testCaption";  
//Set_WindowFlags  
faceplate.WindowFlags = HmiWindowFlag.CanSize;  
//Set_Icon  
faceplate.Icon = "testCaption";  
//Set_Top  
faceplate.Top = 50;  
//Set_Left  
faceplate.Left = 50;  
//Set_Width  
faceplate.Width = 50;  
//Set_Height  
faceplate.Height = 50;  
//Set_RequireExplicitRelease  
faceplate.RequireExplicitRelease = true;  
//Set_Authorization  
faceplate.Authorization = Authorization;  
//Set_Name  
faceplate.Name = "TestName";  
//Set_Visible  
faceplate.Visible = true;  
//Set_Enabled  
faceplate.Enabled = true;  
//Set_TabIndex  
faceplate.TabIndex = 10;
```

Hinweis

Get - Recoverable exception wird ausgelöst, wenn die betreffende Konsistenzprüfung für die angegebene Eigenschaft fehlschlägt. Wenn während des Festlegens von Eigenschaften der Schreibvorgang die Konsistenzprüfungen nicht besteht, wird eine Recoverable exception ausgelöst.

Siehe auch

- Verbindung zum TIA Portal aufbauen (Seite 79)
- Projekt öffnen (Seite 111)
- HMI-Unified-Software-Objekt (Seite 281)
- Mit Steuerbildobjekten arbeiten (Seite 380)

Mit Dynamisierung und Ereignissen einer Bildbausteininstanz arbeiten

Einleitung

Sie können mit TIA Portal Openness folgende Aktionen in Verbindung mit Dynamisierungen und Ereignisobjekten von Bildbausteineigenschaften durchführen:

- Auf Dynamisierung (Variablen-Verbindungen, Animationen, lokale Skripte) für Bildbausteininstanzen zugreifen
- Auf Bildbausteininstanzereignisse mit Skripten und Systemfunktionen zugreifen
- Auf Eigenschaftereignisse einer Bildbausteininstanz zugreifen

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)

Programmcode: Auf Dynamisierung für Bildbausteininstanzen zugreifen

Eine ausführliche Beschreibung, wie Sie auf die Dynamisierung für Bildbausteininstanzen zugreifen, finden Sie unter Mit Dynamisierung für Bilder/Bildobjekte arbeiten (Seite 415).

Programmcode: Auf Bildbausteininstanzereignisse mit Skripten und Systemfunktionen zugreifen

Eine ausführliche Beschreibung, wie Sie auf Bildbausteininstanzereignisse mit Skripten und Systemfunktionen zugreifen, finden Sie unter Mit Dynamisierung und Ereignissen für Bilder/Bildobjekte über Skripte arbeiten (Seite 412).

Programmcode: Auf Eigenschaftereignisse einer Bildbausteininstanz zugreifen

Eine ausführliche Beschreibung, wie Sie auf die Eigenschaftereignisse einer Bildbausteininstanz zugreifen, finden Sie unter Mit Ereignissen für Bilder/Bildobjekte und Eigenschaften arbeiten (Seite 409).

Siehe auch

- Verbindung zum TIA Portal aufbauen (Seite 79)
- Projekt öffnen (Seite 111)
- HMI-Unified-Software-Objekt (Seite 281)
- Mit Dynamisierung für Bilder/Bildobjekte arbeiten (Seite 415)

Mit Dynamisierung und Ereignissen für Bilder/Bildobjekte über Skripte arbeiten (Seite 412)

Mit Ereignissen für Bilder/Bildobjekte und Eigenschaften arbeiten (Seite 409)

7.18.9.6 Mit Ereignissen für Bilder/Bildobjekte und Eigenschaften arbeiten

Einleitung

Sie können folgende Aktionen mit Eigenschaften-Ereignissen und Ereignissen für Bilder und Bildobjekte durchführen, während Sie TIA Portal Openness verwenden:

- Eigenschaften-Ereignisse erstellen
- Eigenschaften-Ereignisse enumerieren
- Eigenschaften-Ereignisse löschen
- Auf Eigenschaften von Ereignissen zugreifen
- Ereignisse für Bilder und Bildobjekte erstellen
- Ereignisse für Bilder und Bildobjekte enumerieren
- Auf Eigenschaften von Ereignissen für Bilder und Bildobjekte zugreifen
- Ereignisse für Bilder und Bildobjekte löschen

Voraussetzung

- TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Zugriff auf HMI-Unified-Software-Objekt
Siehe HMI-Unified-Software-Objekt (Seite 281)

Programmcode

Bei der Arbeit mit Ereignissen für Bilder/Bildobjekte können Sie den folgenden Beispiel-Programmcode ändern und verwenden.

Eigenschaften-Ereignisse erstellen

Ändern Sie folgenden Programmcode, um Eigenschaften-Ereignisse zu erstellen:

```
HmiScreen screen = ((HmiSoftware)targetSW).Screens.Create("MyScreen");
HmiCircle hmiCircle = screen.ScreenItems.Create<HmiCircle>("MYCircle");
//Event Creation
PropertyEventHandler propertyEvent = hmiCircle.PropertyEventHandlers.Create("ToolTipText",
PropertyEventType.Change);
```

Eigenschaften-Ereignisse enumerieren

Ändern Sie folgenden Programmcode, um Eigenschaften-Ereignisse zu enumerieren:

```
HmiScreen screen = ((HmiSoftware)targetSW).Screens.Create("MyScreen");
HmiCircle hmiCircle = screen.ScreenItems.Create<HmiCircle>("MYCircle");
var hmiScreenPropeventDyn = screen.PropertyEventHandlers.Create("Enabled",
PropertyEventType.Change);
var enabledEvent = screen.PropertyEventHandlers.Find("Enabled", PropertyEventType.Change);
//Event Browse
Console.WriteLine ("CountofEventActions"+ screen.PropertyEventHandlers.Count.ToString());
```

Eigenschaften-Ereignisse löschen

Ändern Sie folgenden Programmcode, um Eigenschaften-Ereignisse zu löschen:

```
public void Delete()
PropertyEventHandler propertyEvent =
((HmiSoftware)targetSW).Screens[0].ScreenItems[0].PropertyEventHandlers.Create("ToolTipText",
PropertyEventType.Change);
var changedEvent = screen.PropertyEventHandlers.Find("ToolTipText",
PropertyEventType.Change);
//Event Deletion
changedEvent.Delete();
```

Auf Eigenschaften von Ereignissen zugreifen

Ändern Sie folgenden Programmcode, um Ereignis-Eigenschaften abzurufen:

```
//Event Get
PropertyEventType eventType = propertyEvent.EventType;
string propertyName = propertyEvent.PropertyName;
IHmiScript hmiScript = propertyEvent.Script;
//Script Action in Event Get
bool hmiScriptEventAsync = hmiScript.Async;
string hmiScriptEventGlobal = hmiScript.GlobalDefinitionAreaScriptCode;
string hmiScriptEventCode = hmiScript.ScriptCode;
HmiValidationResult result = hmiScript.SyntaxCheck();
```

Ereignisse für Bilder und Bildobjekte erstellen

Ändern Sie folgenden Programmcode, um das Ereignis im Bild oder das Objekt auf Bildobjektebene mit dem EventHandler zu erstellen:

```
var screenComp = ((HmiSoftware)targetSWTag).Screens;
HmiScreen screenOne = screenComp.Create("Screen2");
HmiScreenEventHandler screenHandler =
screenOne.EventHandlers.Create(HmiScreenEventType.Unloaded);
```

Ereignisse für Bilder und Bildobjekte enumerieren

Ändern Sie folgenden Programmcode, um Ereignisse für Bilder und Bildobjekte zu enumerieren:

```
public EventHandler Find(string propertyName);
var screenComp = ((HmiSoftware)targetSWTag).Screens;
HmiScreen screenOne = screenComp.Create("Screen2");
HmiScreenEventHandler screenHandler =
screenOne.EventHandlers.Create(HmiScreenEventType.Unloaded);
HmiScreenEventHandler screenHandlerSearched =
screenOne.EventHandlers.Find(HmiScreenEventType.Unloaded);
```

Auf Eigenschaften von Ereignissen für Bilder und Bildobjekte zugreifen

Ändern Sie folgenden Programmcode, um Eigenschaften für Bilder und Bildobjekte abzurufen:

```
var screenComp = ((HmiSoftware)targetSWTag).Screens;
HmiScreen screenOne = screenComp.Create("Screen2");
HmiScreenEventHandler screenHandler =
screenOne.EventHandlers.Create(HmiScreenEventType.Unloaded);
Console.WriteLine(screenOne.Name + ".EventHandler.EventType - {0}",
screenHandler.EventType);
Console.WriteLine(screenOne.Name + ".EventHandler.Script.Async - {0}",
screenHandler.Script.Async);
Console.WriteLine(screenOne.Name + ".EventHandler.Script.GlobalDefinitionAreaScriptCode - {0}",
screenHandler.Script.GlobalDefinitionAreaScriptCode);
Console.WriteLine(screenOne.Name + ".EventHandler.Script.ScriptCode - {0}",
screenHandler.Script.ScriptCode);
```

Ereignisse für Bilder und Bildobjekte löschen

Ändern Sie folgenden Programmcode, um Ereignisse für Bilder und Bildobjekte zu löschen:

```
public void Delete ()
HmiSoftware hmisoftware = GetHMISoftware();
HmiScreen hmiscreen = hmisoftware.Screens.Create(Guid.NewGuid().ToString());
var hmiscreenPropEventDyn = hmiscreen.PropertyEventHandlers.Create("Enabled",
PropertyEventType.Change);

var enabledevent = hmiscreen.PropertyEventHandlers.Find("Enabled",
PropertyEventType.Change);
enabledevent.Delete();
```

Hinweis

Create - Recoverable exception wird ausgelöst, wenn die betreffenden Eigenschaften-Ereignisse und Ereignisse für Bilder und Bildobjekte für die angegebene Eigenschaft nicht unterstützt werden. Wenn während des Festlegens von Eigenschaften der Schreibvorgang die Konsistenzprüfungen nicht besteht, wird eine Recoverable exception ausgelöst.

Siehe auch

- Verbindung zum TIA Portal aufbauen (Seite 79)
- Projekt öffnen (Seite 111)
- HMI-Unified-Software-Objekt (Seite 281)

7.18.9.7 Mit Dynamisierung und Ereignissen für Bilder/Bildobjekte über Skripte arbeiten

Einleitung

Sie können folgende Aktionen mit Skripten zur Konfiguration der Eigenschaften-Dynamisierung und von Ereignissen für Bilder/Bildobjekte durchführen, während Sie TIA Portal Openness verwenden:

- Skript-Dynamisierungen erstellen
- Skript-Dynamisierungen enumerieren
- Skript-Dynamisierungen löschen
- Auf Eigenschaften von Skript-Dynamisierungen zugreifen
- Auf Eigenschaften eines Triggers zugreifen
- Syntax prüfen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Unified-Softwareobjekt
Siehe HMI-Unified-Software-Objekt (Seite 281)

Programmcode

Bei der Arbeit mit der Skript-Dynamisierung für Bilder/Bildobjekte können Sie den folgenden Beispiel-Programmcode ändern und verwenden.

Skript-Dynamisierungen erstellen

Ändern Sie folgenden Programmcode, um die Skript-Dynamisierungen für eine Eigenschaft zu erstellen:

```
public DynamizationBase Create<DynamizationBase>(string propertyName);  
HmiScreen screen = m_hmisoftware.Screen[0];  
DynamizationBaseComposition dys = screen.Dynamizations;  
ScriptDynamization scriptDynamic = dys.Create<ScriptDynamization>("BackColor");
```

Skript-Dynamisierungen enumerieren

Ändern Sie folgenden Programmcode, um die Skript-Dynamisierung einer Eigenschaft zu enumerieren:

```
public DynamizationBase Find(string propertyName);
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dyns = screen.Dynamizations;
if (dys is ScriptDynamization)
{
ScriptDynamization scriptDynamic = (ScriptDynamization)dyns.Find("BackColor");
}
```

Skript-Dynamisierungen löschen

Ändern Sie folgenden Programmcode, um die Skript-Dynamisierung für eine Eigenschaft zu löschen:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dyns = screen.Dynamizations;
if (dys is ScriptDynamization)
ScriptDynamization scriptDynamic = (ScriptDynamization)dyns.Find("BackColor");
scriptDynamic.Delete();
```

Auf Eigenschaften von Skript-Dynamisierungen zugreifen

Ändern Sie folgenden Programmcode, um Eigenschaften der Skript-Dynamisierung abzurufen und festzulegen:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dyns = screen.Dynamizations;
foreach (DynamizationBase dynamic in dys)
{
if (dynamic.DynamizationType == DynamizationType.Script)
{
ScriptDynamization scriptDynamization = (ScriptDynamization)dynamic;
scriptDynamization.Async = true;
scriptDynamization.GlobalDefinitionAreaScriptCode = @"Add(parameter1,parameter2)";
scriptDynamization.ScriptCode = @"
var value;
let tag1 = Tags('Tag_1');
let tagValue1 = tag1.Read();
HMIRuntime.Trace('value of MyTag1: ' + tagValue1);
return value; ";
Trigger triggerObj = scriptDynamization.Trigger;
}
}
```

Auf Eigenschaften eines Triggers zugreifen

Ändern Sie folgenden Programmcode, um Eigenschaften eines Triggers abzurufen und festzulegen:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dysn = screen.Dynamizations;
foreach (DynamizationBase dynamic in dysn)
{
if (dynamic.DynamizationType == DynamizationType.Script)
{
ScriptDynamization scriptDynamization = (ScriptDynamization)dynamic;
// Property write
scriptDynamization.Trigger.Type = TriggerType.CustomCycle;
scriptDynamization.Trigger.CustomDuration = "T500ms";
scriptDynamization.Trigger.Tags = new List<string>() { "Tag_1" };
// Property read
Console.WriteLine(scriptDynamization.Trigger.Type);
Console.WriteLine(scriptDynamization.Trigger.CustomDuration);
Console.WriteLine(scriptDynamization.Trigger.Tags);
}
}
```

Syntaxprüfung

Ändern Sie folgenden Programmcode, um mit der Skriptdynamisierungsaktion (Methode) SyntaxCheck die Syntax von Skript-Code oder von Code eines globalen Skripts zu prüfen.

```
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dysn = screen.Dynamizations;
foreach (DynamizationBase dynamic in dysn)
{
if (dynamic.DynamizationType == DynamizationType.Script)
{
ScriptDynamization scriptDynamization = (ScriptDynamization)dynamic;
scriptDynamization.ScriptCode = @""
var value;let tag1 = Tags('Tag_1');
let tagValue1 = tag1.Read();
HMRuntime.Trace('value of MyTag1: ' + tagValue1);
return value; "
HmiValidationResult syntaxCkResult = scriptDynamization.SyntaxCheck();
IEnumerable<string> error = syntaxCkResult.Errors;
IEnumerable<string> warnings = syntaxCkResult.Warning;
}
}
```

Hinweis

Create - Recoverable exception wird ausgelöst, wenn die betreffende Dynamisierung für die angegebene Eigenschaft nicht unterstützt wird. Wenn während des Festlegens von Eigenschaften der Schreibvorgang die Konsistenzprüfungen nicht besteht, wird eine Recoverable exception ausgelöst.

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[HMI-Unified-Software-Objekt \(Seite 281\)](#)

7.18.9.8 Mit Dynamisierung für Bilder/Bildobjekte arbeiten

Einleitung

Sie können folgende Aktionen mit der Dynamisierungsfunktion von Bildern/Bildobjekten durchführen, während Sie TIA Portal Openness verwenden:

- Dynamisierung für eine Eigenschaft erstellen
- Dynamisierung für eine Eigenschaft enumerieren
- Dynamisierung für eine Eigenschaft löschen
- Auf gemeinsame Eigenschaften der Dynamisierung zugreifen
- Auf Eigenschaften der Variablen-Dynamisierung zugreifen
- Auf Eigenschaften der Blink-Dynamisierung zugreifen
- Auf Eigenschaften der Ressourcenlisten-Dynamisierung zugreifen

Hinweis

Um auf die Dynamisierung für eine Eigenschaft auf Part-Objekt-Ebene zuzugreifen, müssen Sie auf das entsprechende Part-Objekt der Controls zugreifen. Informationen zu Parts finden Sie unter [Auf Eigenschaften einer Kurvenanzeige zugreifen \(Seite 79\)](#)

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe [Verbindung zum TIA Portal herstellen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
Siehe [Öffnen eines Projekts \(Seite 111\)](#)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe [HMI-Unified-Software-Objekt \(Seite 281\)](#)

Programmcode

Bei der Arbeit mit Dynamisierungen für Bilder/Bildobjekte können Sie den folgenden Beispiel-Programmcode ändern und verwenden.

Dynamisierung für eine Eigenschaft erstellen

Ändern Sie folgenden Programmcode, um die Dynamisierung für eine Eigenschaft zu erstellen:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dysns = screen.Dynamizations;
// Flushing dynamization
FlashingDynamization FlashingDyn = dysns.Create<FlashingDynamization>("BackColor");
// ResourceList dynamization
ResourceListDynamization ResListDyn = dysns.Create<ResourceListDynamization>("DisplayName");
// Tag dynamization
TagDynamization tagDyn = dysns.Create<TagDynamization>("Width");
```

Dynamisierung für eine Eigenschaft enumerieren

Ändern Sie folgenden Programmcode, um die Dynamisierung für eine Eigenschaft zu enumerieren:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dysns = screen.Dynamizations;
// Flushing dynamization
FlashingDynamization FlashDyn = (FlashingDynamization)dysns.Find("BackColor");
// ResourceList dynamization
ResourceListDynamization resourceDyn = (ResourceListDynamization)dysns.Find("DisplayName");
// Tag dynamization
TagDynamization tagDyn = (TagDynamization)dysns.Find("Width");
```

Dynamisierung für eine Eigenschaft löschen

Ändern Sie folgenden Programmcode, um die Dynamisierung einer Eigenschaft zu löschen:

```
public void Delete ()
HmiScreen screen = m_hmiSoftware.Screens[0];
DynamizationBaseComposition dysns = screen.Dynamizations;
DynamizationBase dynamization = dysns.Find("BackColor");
dynamization.Delete();
```

Auf gemeinsame Eigenschaften der Dynamisierung zugreifen

Ändern Sie folgenden Programmcode, um gemeinsame Eigenschaften der Dynamisierung abzurufen:

```
HmiSoftware m_hmiSoftware = ...;
DynamizationBaseComposition screenDynamizations = m_hmiSoftware.Screens[0].Dynamizations;
DynamizationBase screenDynamization = screenDynamizations.Find("BackColor");
string propertyName = screenDynamization.PropertyName;
DynamizationType dynamizationType = screenDynamization.DynamizationType;
```

Auf Eigenschaften der Variablen-Dynamisierung zugreifen

Ändern Sie folgenden Programmcode, um Eigenschaften der Variablen-Dynamisierung abzurufen und festzulegen:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
TagDynamization tagDynamization = (TagDynamization)screen.Dynamizations.Find("Width");
foreach (DynamizationBase dynamization in screen.Dynamizations)
{
if (dynamization.DynamizationType == DynamizationType.Tag)
{
TagDynamization TagDynamization = (TagDynamization)dynamization;
TagDynamization.Tag = "HmiTagName";
TagDynamization.UseIndirectAddressing = true;
TagDynamization.ReadOnly = true;
}
}
```

Auf Eigenschaften der Blink-Dynamisierung zugreifen

Ändern Sie folgenden Programmcode, um Eigenschaften der Blink-Dynamisierung abzurufen und festzulegen:

```
HmiScreen screen = m_hmiSoftware.Screens[0];
foreach (DynamizationBase dynamization in screen.Dynamizations)
{
if (dynamization.DynamizationType == DynamizationType.Floating)
{
FlashingDynamization flashingDynamization = (FlashingDynamization)dynamization;
flashingDynamization.AlternateColor = Color.Beige;
flashingDynamization.Color = Color.FromArgb(12, 22, 152, 200);
flashingDynamization.FlashingCondition = FlashingCondition.Always;
flashingDynamization.FlashingRate = FlashingRate.Medium;
}
}
```

Auf Eigenschaften der Ressourcenlisten-Dynamisierung zugreifen

Ändern Sie folgenden Programmcode, um Eigenschaften der Ressourcenlisten-Dynamisierung abzurufen und festzulegen

```
HmiScreen screen = m_hmiSoftware.Screens[0];
foreach (DynamizationBase dynamization in screen.Dynamizations)
{
if (dynamization.DynamizationType == DynamizationType.ResourceList)
{
ResourceListDynamization resourceListDyn = (ResourceListDynamization)dynamization;
resourceListDyn.Tag = "Tag name";
resourceListDyn.ResourceList = "Resource list name";
}
}
```

Hinweis

Create - Recoverable exception wird ausgelöst, wenn die betreffende Dynamisierung für die angegebene Eigenschaft nicht unterstützt wird. Wenn während des Festlegens von Eigenschaften der Schreibvorgang die Konsistenzprüfungen nicht besteht, wird eine Recoverable exception ausgelöst.

Siehe auch

- [Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- [Projekt öffnen \(Seite 111\)](#)
- [HMI-Unified-Software-Objekt \(Seite 281\)](#)
- [Auf Eigenschaften einer Kurvenanzeige zugreifen \(Seite 393\)](#)

7.18.9.9 Lizenz für den Zugriff auf ein Unified-Gerät prüfen

Einleitung

Über die TIA Openness-Schnittstelle können Sie jede Openness Client-Anwendung auf das Vorhandensein einer WinCC Unified-Lizenz überprüfen, so dass eine Client-Anwendung die Lizenzierung nicht durch Verwendung von Openness-Schnittstellen umgehen kann.

Sie haben nur Zugriff auf ein HMI-Gerät (Unified) und seine unterlagerten Objekte (z. B. Variablen, Alarme, Bilder usw.), wenn eine HMI Unified-Lizenz vorhanden ist. Ist keine Lizenz für WinCC Unified vorhanden, führt jede Aktion auf der Geräteebene (z. B. Erstellen, Suchen, Löschen) zur Rückgabe eines Nullwerts von der Openness API.

Hinweis

Für HMI Unified Software wird bei ungültiger Lizenz ein Nullwert zurückgegeben.

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Es besteht Zugriff auf das HMI-Softwareobjekt
Siehe HMI Unified-Software-Objekt (Seite 281)

Programmcode

Ändern Sie folgenden Programmcode, um auf das Gerät oder Teilsysteme des Geräts zuzugreifen, wenn keine Lizenz vorhanden ist; es wird ein Nullwert zurückgegeben:

```
private static void ListDevices(Project project)
{
foreach (var device in project.Devices)
{
ListDeviceItem(device);
}
}
private static void ListDeviceItem(Device device)
{
Console.WriteLine("HMI device - " + device.Name);
foreach (var item in device.DeviceItems)
{
Console.WriteLine("HMI device type - " + item.TypeIdentifier);
var softContainer = item.GetService<SoftwareContainer>();
if (softContainer != null)
{
var softTarget = softContainer.Software;
if (softTarget != null)
{
Console.WriteLine("HMI device software - " + softTarget.Name);
}
}
}
}
```

Hinweis

Während des Zugriffs auf obigen Programmcode ist das SoftTarget null und es ist kein Zugriff auf Teilsysteme wie Alarme oder Variablen möglich.

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Ändern Sie folgenden Programmcode, um auf das Gerät oder Teilsysteme des Geräts zuzugreifen, wenn eine Lizenz vorhanden ist; es wird ein Nullwert zurückgegeben:

```
private static void ListDevices(Project project)
{
foreach (var device in project.Devices)
{
ListDeviceItem(device);
}
}
private static void ListDeviceItem(Device device)
{
Console.WriteLine("HMI device - " + device.Name);
foreach (var item in device.DeviceItems)
{
Console.WriteLine("HMI device type - " + item.TypeIdentifier);
var softContainer = item.GetService<SoftwareContainer>();
if (softContainer != null)
{
var softTarget = softContainer.Software;
}
if (softTarget != null)
{
Console.WriteLine("HMI device software - " + softTarget.Name);
}
}
}
}
```

Hinweis

Während des Zugriffs auf obigen Programmcode enthält das SoftTarget Gerätbehälter und der Zugriff auf Teilsysteme wie Alarme und Variablen ist möglich.

Ein Rückgabewert NULL für einen Gerätbehälter zeigt an, dass die Lizenz nicht verfügbar ist. Die wiederherstellbare Ausnahme unterstützt keine benutzerdefinierten Ausnahmemeldungen, daher wird NULL als Rückgabewert verwendet.

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
[HMI-Unified-Software-Objekt \(Seite 281\)](#)

7.18.10 Zugriff auf Hierarchien im Common Plant Model

7.18.10.1 Mit der Anlagensicht arbeiten

Einleitung

Sie können die folgenden Aktionen für die Anlagensicht durchführen, während Sie TIA Portal Openness nutzen:

- Anlagensicht erstellen
- Anlagensicht löschen
- Anlagensicht umbenennen

Voraussetzung

- TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Programmcode

Ändern Sie folgenden Programmcode, um eine Anlagensicht zu erstellen:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;  
PlantView plantView = plantViews.Create("ABCManufacturingPlant");
```

Hinweis

Sie können nur eine Anlagensicht innerhalb eines Projekts erstellen.

Ändern Sie folgenden Programmcode, um eine Anlagensicht zu löschen oder zu entfernen:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;  
PlantView plantView = plantViews.Find("ABCManufacturingPlant");  
plantView.Delete();
```

Ändern Sie folgenden Programmcode, um eine Anlagensicht durch Aktualisierung der Eigenschaft 'Name' umzubenennen:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;  
PlantView plantView = plantViews.Find("ABCManufacturingPlant");  
plantView.Name = "New_PlantView_Name";
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.18.10.2 Mit Anlagensichtknoten arbeiten

Einleitung

Sie können die folgenden Aktionen für Anlagensichtknoten durchführen, während Sie TIA Portal Openness nutzen:

- Anlagensichtknoten erstellen
- Anlagensichtknoten löschen
- Anlagensichtknoten umbenennen

Eigenschaften

Die folgenden Eigenschaften werden in einem Anlagensichtknoten mit TIA Portal Openness unterstützt:

Name der Eigenschaft	Datentyp	Zugriffsmöglichkeit
HierarchyPath	String	R
PlantObject	Siemens.Engineering.HmiUnified.Cpm.PlantObject	R
Name	String	R/W
PlantView	String	R
ViewPath	String	R
PlantViewNodes	PlantViewNodeComposition	R

Voraussetzung

- TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Programmcode

Ändern Sie folgenden Programmcode, um einen Anlagensichtknoten zu erstellen:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNodeComposition viewNodes = plantView.PlantViewNodes;
PlantViewNode mixingNode = viewNodes.Create("Mixing");
```

Ändern Sie folgenden Programmcode, um einen Anlagensichtknoten unter einem anderen Anlagensichtknoten zu erstellen:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;  
PlantView plantView = plantViews.Find("ABCManufacturingPlant");  
PlantViewNodeComposition viewNodes = plantView.PlantViewNodes;  
PlantViewNode mixingNode = viewNodes.Create("Mixing");  
PlantViewNode motorNode = mixingNode.Create("Motor");
```

Hinweis

Sie können beliebig viele Anlagensichtknoten innerhalb einer Anlagensicht oder unter einem anderen Anlagensichtknoten erstellen.

Ändern Sie folgenden Programmcode, um einen Anlagensichtknoten zu entfernen oder zu löschen:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;  
PlantView plantView = plantViews.Find("ABCManufacturingPlant");  
PlantViewNode mixingNode = plantView.PlantViewNodes.Find("Mixing");  
mixingNode.Delete();
```

Ändern Sie folgenden Programmcode, um einen Anlagensichtknoten durch Aktualisierung der Eigenschaft 'Name' umzubenennen:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;  
PlantView plantView = plantViews.Find("ABCManufacturingPlant");  
PlantViewNodeComposition viewNodes = plantView.PlantViewNodes;  
PlantViewNode mixingNode = viewNodes.Create("Mixing");  
mixingNode.Name = "New_PlantViewNode_Name";
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.18.10.3 Anlagensicht enumerieren

Einleitung

Sie können mit TIA Portal Openness die in einem bestimmten Projekt im TIA Portal vorhandene Anlagensicht zu enumerieren.

Voraussetzung

- TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Programmcode

Ändern Sie folgenden Programmcode, um alle Anlagensichten des Projekts zu enumerieren:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
foreach (var item in plantViews)
{
// work with plant views (currently only 1 plant view can be created inside a project)
}
```

Ändern Sie folgenden Programmcode, um eine Anlagensicht in einer Liste der Anlagensichten anhand des Namens zu finden:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
```

Ändern Sie folgenden Programmcode, um eine Anlagensicht in einer Liste der Anlagensichten anhand des Index zu finden:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews[0];
```

Ändern Sie folgenden Programmcode, um einen Anlagensichtknoten in einer Liste der Anlagensichten anhand des Hierarchiepfads des gewünschten Knotens zu finden:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantViewNode plantViewNode = plantViews.FindPlantViewNode("ABCManufacturingPlan\Mixing
\Motor");
```

Ändern Sie folgenden Programmcode, um einen Anlagensichtknoten in einer Liste der Anlagensichtknoten anhand des Namens der Anlagensicht zu finden:

```
PlantViewNodeComposition plantViewNodes =
m_tiaPortalApp.Projects[0].PlantViews[0].PlantViewNodes;
PlantViewNode plantViewNode = plantViewNodes.Find("Mixing");
```

Ändern Sie folgenden Programmcode, um alle Listen von Knoten in Anlagensichten, die innerhalb der Anlagensicht verfügbar sind, zu enumerieren:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;  
PlantViewNodeComposition nodes = plantView.PlanViewNodes;  
foreach (var item in nodes)  
{  
    // work with plant view nodes inside plant view  
}
```

Ändern Sie folgenden Programmcode, um alle Knoten in Anlagensichten, die innerhalb sämtlicher Hierarchiestufen verfügbar sind, zu enumerieren:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;  
PlantViewNodeComposition nodes = plantView.PlanViewNodes;  
PlantViewNodeComposition subNodes = nodes[2].PlantViewNodes;  
foreach (var item in subNodes)  
{  
    // work with plant view nodes inside 2nd node within the plant view  
}
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.18.10.4 Mit der Anlagensicht und Geräten arbeiten

Einleitung

Sie können folgende Aktionen für den Zugriff auf Hierarchien in Anlagensichten und Anlagenobjekte durchführen, während Sie TIA Portal Openness verwenden:

- Gerät einer Anlagensicht zuordnen
- Gerät einer Anlagensicht ändern
- Gerät aus einer Anlagensicht entfernen

Voraussetzung

- TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe [Verbindung zum TIA Portal herstellen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
Siehe [Öffnen eines Projekts \(Seite 111\)](#)

Programmcode

Ändern Sie folgenden Programmcode, um durch Festlegen der Eigenschaft 'AssignedHmiDevice' ein Gerät der Anlagensicht zuzuordnen:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;  
PlantView plantView = plantViews.Create("ABCManufacturingPlant");  
plantView.AssignedHmiDevice = "HMI_RT_1";
```

Ändern Sie folgenden Programmcode, um durch Festlegen eines Gerätenamens in der Eigenschaft 'AssignedHmiDevice' ein Gerät in der Anlagensicht zu ändern:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;  
PlantView plantView = plantViews.Create("ABCManufacturingPlant");  
plantView.AssignedHmiDevice = "HMI_RT_2";
```

Ändern Sie folgenden Programmcode, um durch Festlegen eines leeren Gerätenamens in der Eigenschaft 'AssignedHmiDevice' ein Gerät in der Anlagensicht zu entfernen:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;  
PlantView plantView = plantViews.Create("ABCManufacturingPlant");  
plantView.AssignedHmiDevice = string.Empty;
```

Hinweis

Um das Gerät zu entfernen, verwenden Sie nur `string.empty`. `NULL` oder Leerzeichen (" ") wird als Gerätename erkannt und erzeugt eine Fehlermeldung, dass das Gerät nicht gefunden wurde.

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.18.11 Zugriff auf Instanzen im Common Plant Model

7.18.11.1 Mit CPM-Objektinstanzen arbeiten

Einleitung

Sie können folgende Aktionen für den Zugriff auf eine CPM-Instanz anhand des CPM-Typs durchführen, während Sie TIA Portal Openness verwenden:

- CPM-Objektinstanzen erstellen
- CPM-Objektinstanzen löschen
- CPM-Objektinstanzen umbenennen

Voraussetzung

- TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Eine Anlage ist erstellt

Programmcode

Ändern Sie folgenden Programmcode, um eine CPM-Objektinstanz zu erstellen:

```
// Instance inside plant view
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Create("ABCManufacturingPlant");
plantView.AssignedHmiDevice = "HMI_RT_1";
PlantViewNodeComposition nodes = plantView.PlantViewNodes;
PlantViewNode node = nodes.Create("ObjectLevel1", "Plant_Object_Type_1");
//Instance inside plant view node
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer",
"Plant_Object_Type_1");
```

Ändern Sie folgenden Programmcode, um eine CPM-Objektinstanz zu entfernen oder zu löschen:

```
PlantViewNode mixingNode = plantView.PlantViewNodes.Find("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Find("IngredientMixer");
mixerObject.Delete();
```

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Ändern Sie folgenden Programmcode, um durch Aktualisierung der Eigenschaft 'Name' eine CPM-Objektinstanz umbenennen:

```
PlantViewNode mixingNode = plantView.PlantViewNodes.Find("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Find("IngredientMixer");
mixerObject.Name = "New_PlantViewNode_Name";
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.18.11.2 Mit Anlagensichtknoten von CPM-Objektinstanzen arbeiten

Einleitung

Sie können die folgenden Aktionen für den Zugriff auf Anlagensichtknoten von CPM-Objektinstanzen durchführen, während Sie TIA Portal Openness nutzen:

- Anlagensichtknoten erstellen
- Anlagensichtknoten löschen
- Anlagensichtknoten umbenennen

Voraussetzung

- TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe [Verbindung zum TIA Portal herstellen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
Siehe [Öffnen eines Projekts \(Seite 111\)](#)

Programmcode

Ändern Sie folgenden Programmcode, um einen Anlagensichtknoten innerhalb einer CPM-Objektinstanz zu erstellen:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNode mixerObject = plantView.PlantViewNodes.Create("ObjectLevel1",
"Plant_Object_Type_1");
PlantViewNode mixingNode = mixerObject.PlantViewNodes.Create("Mixing");
```

Hinweis

Sie können beliebig viele Anlagensichtknoten innerhalb eines CPM-Objekts erstellen.

Ändern Sie folgenden Programmcode, um eine CPM-Objektinstanz innerhalb eines Anlagensichtknotens zu erstellen:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject = mixingNode.PlantViewNodes.Create("ObjectLevel1",
"Plant_Object_Type_1");
```

Hinweis

Sie können beliebig viele CPM-Objektinstanzen innerhalb eines Anlagensichtknotens erstellen.

Ändern Sie folgenden Programmcode, um einen Anlagensichtknoten aus einem Anlagensichtknoten zu entfernen oder zu löschen:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNode mixerObject = plantView.PlantViewNodes.Find("ObjectLevel1");
PlantViewNode mixingNode = mixerObject.PlantViewNodes.Find("Mixing");
mixingNode.Delete();
```

Ändern Sie folgenden Programmcode, um durch Aktualisierung der Eigenschaft 'Name' einen Anlagensichtknoten einer CPM-Objektinstanz umzubenennen:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews.Find("ABCManufacturingPlant");
PlantViewNode mixerObject = plantView.PlantViewNodes.Find("ObjectLevel1",
"Plant_Object_Type_1");
PlantViewNode mixingNode = mixerObject.PlantViewNodes.Find("Mixing");
mixingNode.Name = "New_PlantViewNode_Name";
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.18.11.3 Schnittstellenvariablen von CPM-Objektinstanzen enumerieren

Einleitung

Sie können mit TIA Portal Openness ein Member einer CPM-Objektinstanz enumerieren.

Voraussetzung

- TIA Portal Openess ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Programmcode

Ändern Sie folgenden Programmcode, um alle Members der CPM-Objektinstanz zu enumerieren:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;  
PlantView plantView = plantViews[0];  
PlantViewNodeComposition plantViewNodes = plantView.PlantViewNodes;  
PlantViewNode plantViewNode = plantViewNodes.Find("MixerObject_1");  
PlantObject plantObject = plantViewNode.PlantObject;  
PlantObjectInterfaceComposition interfaces = plantObject?.PlantObjectInterfaces;
```

Hinweis

Im Beispiel oben bezieht sich der Operator ?. auf eine Nullprüfung.

Ändern Sie folgenden Programmcode, um eine Schnittstellenvariable in der Liste der Schnittstellen anhand des Namens zu finden:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;  
PlantView plantView = plantViews[0];  
PlantViewNodeComposition plantNodes = plantView.PlantViewNodes;  
PlantViewNode plantViewNode = plantNodes.Find("MixerObject_1");  
PlantObject plantObject = plantViewNode.PlantObject;  
PlantObjectInterface interface = plantObject. PlantObjectInterfaces.Find("Interface_1");
```

Ändern Sie folgenden Programmcode, um eine Schnittstellenvariable in der Liste der Schnittstellen anhand des Index zu finden:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;  
PlantView plantView = plantViews[0];  
PlantViewNodeComposition plantViewNodes = plantView.PlantViewNodes;  
PlantViewNode plantViewNode = plantViewNodes.Find("MixerObject_1");  
PlantObject plantObject = plantViewNode.PlantObject;  
PlantObjectInterfaceComposition interfaces = plantObject. PlantObjectInterfaces;  
PlantObjectInterface interface = interfaces[0];
```

Siehe auch

Verbindung zum TIA Portal aufbauen (Seite 79)

Projekt öffnen (Seite 111)

7.18.12 Auf Eigenschaften für Schnittstellen/Archivvariablen von Anlagenobjektinstanzen zugreifen

7.18.12.1 Zugriff auf und Aktualisierung von Eigenschaften von Schnittstellenvariablen von CPM-Anlagenobjektinstanzen

Einleitung

Sie können mit TIA Portal Openness auf die Eigenschaften von Schnittstellenvariablen von CPM-Anlageobjektinstanzen zugreifen und diese aktualisieren.

Auf die folgenden Eigenschaften von Schnittstellenvariablen von CPM-Anlageobjektinstanzen kann zugegriffen werden:

Name der Eigenschaft	Datentyp	Beschreibung	Zugänglichkeit
Name	System.String	Name der Schnittstellenvariablen	R/W
PlcTag	System.String	PLC-Variable zu Schnittstellenvariablen	R/W
Connection	System.String	Anschaltung von Schnittstellenvariablen	R/W
PlcName	System.String	PLC-Name zu Schnittstellenvariablen	R
AccessMode	PlantObjectTagAccessMode	Zugriffsmodus für Schnittstellenvariablen	R
DataType	System.String	Objekttyp der Schnittstellenvariablen	R
MaxLength	System.Int	Länge von Schnittstellenvariablen	R
HmiDataType	System.Int	HMI-Datentyp von Schnittstellenvariablen	R
AcquisitionMode	PlantObjectTagAcquisitionMode	Erfassungsart von Schnittstellenvariablen	R/W
AcquisitionCycle	System.Int	Erfassungszyklus für Schnittstellenvariablen	R/W
Persistent	System.Bool	Persistenz interner Variablen der CPM-Objektinstanz	R
Comment	System.String	Kommentar	R/W
Members	PlantObjectInterfaceMemberComposition	Mitglieder der Schnittstellenvariable	R

Hinweis

Die Eigenschaft PLCTag & Connection kann erst konfiguriert werden, nachdem das HMI-Gerät der Anlagensicht zugeordnet wurde.

Voraussetzung

- TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Programmcode

Ändern Sie folgenden Programmcode, um auf alle Eigenschaften von Schnittstellenvariablen zuzugreifen:

```
// Instance inside plant view node
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer", "Mixer");
PlantObjectInterfaceComposition interfaceTags =
mixerObject.PlantObject.PlantObjectInterfaces;
PlantObjectInterface interfaceTag = interfaceTags.FirstOrDefault();
string name = interfaceTag.Name;
MultilingualText comment = interfaceTag.Comment
```

Ändern Sie folgenden Programmcode, um einige Eigenschaften einer Schnittstellenvariable zu aktualisieren, z. B. Kommentar, PLC-Variable, Anschaltung, Erfassungsmodus, Erfassungszyklus:

```
// Instance inside plant view node
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer", "Mixer");
PlantObjectInterfaceComposition interfaceTags =
mixerObject.PlantObject.PlantObjectInterfaces;
PlantObjectInterface interfaceTag = interfaceTags.FirstOrDefault();
interfaceTag.Name = "New_InterfaceTag_Name";
interfaceTag.Comment.Items[0].Text = "New_InterfaceTag_Comment";
interfaceTag.AcquisitionMode = PlantObjectTagAcquisitionMode.CyclicContinuous;
interfaceTag.AcquisitionCycle = "T12s";
interfaceTag.Connection = "Connection_1";
interfaceTag.PlcTag = "PLC_Tag_1";
```

Siehe auch

Verbindung zum TIA Portal aufbauen (Seite 79)

Projekt öffnen (Seite 111)

7.18.12.2 Zugriff auf und Aktualisierung von Eigenschaften von Member-Variablen von Schnittstellenvariablen

Einleitung

Sie können mit TIA Portal Openness auf die Eigenschaften von Member-Variablen von Schnittstellenvariablen zugreifen und diese aktualisieren.

Auf die folgenden Eigenschaften von Member-Variablen von Schnittstellenvariablen kann zugegriffen werden:

Name der Eigenschaft	Datentyp	Beschreibung	Zugänglichkeit
Name	System.String	Name der Member-Variablen	R
DataType	System.String	Objekttyp der Member-Variablen	R
MaxLength	System.Int	Länge der Member-Variablen	R
HmiDataType	System.String	HMI-Datentyp der Member-Variablen	R
InitialMinValue	PlantObjectTagLowerRange	Unterer Bereichswert	R/W
InitialMaxValue	PlantObjectTagUpperRange	Oberer Bereichswert	R/W
InitialValue	System.Object	Anfangswert	R/W
SubstituteValue	PlantObjectTagSubstituteValue	Ersatzwert	R
Comment	MultilingualText	Kommentar	R/W
Members	PlantObjectInterfaceMemberComposition	Mitglieder der Member-Variable	R
LoggingTags	PlantObjectLoggingTagComposition	Archivvariablen der Member-Variable	R

Voraussetzung

- TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode

Ändern Sie folgenden Programmcode:

```
PlantViewComposition plantViews = m_tiaPortalApp.Projects[0].PlantViews;
PlantView plantView = plantViews[0];
PlantViewNodeComposition plantViewNodeComposition = plantView.PlantViewNodes;
PlantViewNode plantViewNode = plantViewNodeComposition[0];
PlantObjectInterfaceComposition tagComposition =
plantViewNode.PlantObject.PlantObjectInterfaces;
PlantObjectInterface interfaceTag = tagComposition[0];
PlantObjectInterfaceMemberComposition memberTags = interfaceTag.Members;
PlantObjectInterfaceMember memberTag = memberTags.FirstOrDefault();
//GetAttributes
List<string> listOfPropertyNames = new List<string>()
{
    "Name"
};
var listOfPropertyValues = memberTag.GetAttributes(listOfPropertyNames);
//SetAttributes
IEnumerable<KeyValuePair<string, object>> propertyNameValuePairList = new
List<KeyValuePair<string, object>>()
{
    new KeyValuePair<string, object>("Name", "MemberTag_1");
};
memberTag.SetAttributes(propertyNameValuePairList);
```

Ändern Sie folgenden Programmcode, um auf alle Eigenschaften der Member-Variablen zuzugreifen:

```
// Instance inside plant view node
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer", "Mixer");
PlantObjectInterfaceComposition interfaceTags =
mixerObject.PlantObject.PlantObjectInterfaces;
PlantObjectInterfaceMemberComposition memberTags = interfaceTag.Members;
PlantObjectInterfaceMember memberTag = memberTags.FirstOrDefault();
string name = memberTag.Name;
string comment = memberTag.Comment.Items[0].Text;
```

Ändern Sie folgenden Programmcode, um Eigenschaften einer Member-Variable einer CPM-Objektinstanz zu aktualisieren:

```
// Instance inside plant view node
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer", "Mixer");
PlantObjectInterfaceComposition interfaceTags =
mixerObject.PlantObject.PlantObjectInterfaces;
PlantObjectInterfaceMember memberTag = memberTags.FirstOrDefault();
memberTag.Comment.Items[0].Text = "New_MemberTag_Comment";
memberTag.InitialValue = 2;
memberTag.InitialMinValue.Value = 2;
memberTag.InitialMinValue.ValueType = PlantObjectTagLimitValueType.Constant;
memberTag.InitialMaxValue.Value = 2;
memberTag.InitialMaxValue.ValueType = PlantObjectTagLimitValueType.Constant;
```

Ändern Sie folgenden Programmcode, um auf alle Archivvariablen einer Member-Variable einer CPM-Objektinstanz zuzugreifen:

```
// Instance inside plant view node
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer", "Mixer");
PlantObjectInterfaceComposition interfaceTags =
mixerObject.PlantObject.PlantObjectInterfaces;
PlantObjectInterface interfaceTag = interfaceTags.FirstOrDefault();
PlantObjectInterfaceMemberComposition memberTags = interfaceTag.Members;
PlantObjectInterfaceMember memberTag = memberTags.FirstOrDefault();
PlantObjectLoggingTagComposition loggingTags = memberTag.LoggingTags;
PlantObjectLoggingTag loggingTag = loggingTags.FirstOrDefault();
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.18.12.3 Zugriff auf und Aktualisierung von Eigenschaften von Logging-Variablen von Member-Variablen

Einleitung

Sie können mit TIA Portal Openness auf die Archivvariablen von Member-Variablen in CPM-Anlageobjektinstanzen zugreifen.

7.18 Funktionen für den Zugriff auf ein HMI-Gerät (Unified)

Auf die folgenden Eigenschaften von Archivvariablen von Member-Variablen in CPM-Anlageobjektinstanzen kann zugegriffen werden:

Name der Eigenschaft	Datentyp	Beschreibung	Zugänglichkeit
AggregationDelay	System.TimeSpan	Wert für Aggregationsverzögerung	R
AggregationMode	PlantObjectLogging.TagAggregationMode	Wert für Aggregationsmodus	R
Cycle	System.String	Archivierungszyklus der Archivvariable	R
CycleFactor	System.UInt32	Wert für Archivierungszyklusfaktor	R
DataLog	System.String		R
Source	System.string	Quelle der Archivvariable	R
TriggerMode	PlantObjectLogging.TagTriggerMode	Auslösemodus der Archivvariable	R
TriggerTag	System.String	Wert der Auslösevariable	R
TriggerTagBitNumber	System.UInt32	Wert von TriggerTagBitNumber	R
Name	System.string	Name der Archivvariable	R
LogConfiguration	System.String	Datenarchiv der Archivvariable	R
LoggingMode	PlantObjectLogging.TagLoggingMode	Protokollierungsmodus der Archivvariable	R
SmoothingMode	PlantObjectLogging.TagSmoothingMode	Glättungsmodus der Archivvariable	R
SmoothingMinTime	System.TimeSpan	Mindestzeit der Archivvariable	R
SmoothingMaxTime	System.TimeSpan	Höchstzeit der Archivvariable	R
SmoothingDeltaValue	System.Double	Delta der Archivvariable	R
LimitScope	PlantObjectLogging.TagLimitScope	Umfangsbegrenzung der Archivvariable	R
HighLimit	System.Object	Oberer Grenzwert der Archivvariable	R
LowLimit	System.Object	Unterer Grenzwert der Archivvariable	R

Voraussetzung

- TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Programmcode

Ändern Sie folgenden Programmcode, um auf alle Eigenschaften von Archivvariablen zuzugreifen:

```
// Instance inside plant view node
PlantViewNode mixingNode = plantView.PlantViewNodes.Create("Mixing");
PlantViewNode mixerObject =
plantView.PlantViewNodes.Find("Mixing").PlantViewNodes.Create("IngredientMixer", "Mixer");
PlantObjectInterfaceMemberComposition interfaceTags =
mixerObject.PlantObject.PlantObjectInterfaceMembers;
PlantObjectInterfaceMember interfaceTag = interfaceTags.FirstOrDefault();
PlantObjectInterfaceMemberComposition memberTags = interfaceTag.Members;
PlantObjectInterfaceMember memberTag = memberTags.FirstOrDefault();
PlantObjectLoggingTagComposition loggingTags = memberTag.LoggingTags;
PlantObjectLoggingTag loggingTag = loggingTags.FirstOrDefault();
string name = loggingTag.Name;
object lowLimit = loggingTag.LowLimit;
```

Hinweis

Eigenschaften der Klasse PlantObjectLoggingTag zur Archivvariablen einer CPM-Objektinstanz können nicht aktualisiert werden.

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

7.19.1 Status einer PLC ermitteln

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe AUTOHOTSPOT
- Sie haben mit Ihrer TIA Portal Openness-Anwendung ein Projekt geöffnet.
Siehe AUTOHOTSPOT

Verwendung

Sie können den Zustand einer PLC oder aller PLCs in einem Projekt ermitteln.

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

TIA Portal Openness unterscheidet zwischen folgenden Zuständen:

- Offline
- PLC ist verbunden („Verbindung wird hergestellt“)
- Online
- PLC ist nicht verbunden („Verbindung wird getrennt“)
- Inkompatibel
- Nicht erreichbar
- Geschützt

Programmcode

Ändern Sie folgenden Programmcode, um den Zustand eines PLC zu ermitteln:

```
public static OnlineState GetOnlineState(DeviceItem deviceItem)
{
    OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
    return onlineProvider.State;
}
```

Ändern Sie folgenden Programmcode, um den Zustand aller PLCs in einem Projekt zu ermitteln:

```
public static void DetermineOnlineStateOfAllProjectDevices(Project project)
{
    foreach (Device device in project.Devices)
    {
        foreach (DeviceItem deviceItem in device.DeviceItems)
        {
            OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
            if (onlineProvider != null)
            {
                OnlineState state = onlineProvider.State;
            }
        }
    }
}
```

7.19.2 Auf Parameter einer Online-Verbindung zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe AUTOHOTSPOT
- Ein Projekt ist geöffnet.
Siehe AUTOHOTSPOT

Verwendung

Mit der Schnittstelle TIA Portal Openness API können Sie Parameter für eine Online-Verbindung ermitteln oder festlegen:

- Verfügbare Verbindungsarten mit einer PLC enumerieren
- Verfügbare Schnittstellen zu einer PLC enumerieren
- Zugeordnete Steckplätze enumerieren
- Verfügbare Adressen der Subnetze und Gateways enumerieren
- Verbindungsparameter festlegen

Programmcode: Verbindungsparameter ermitteln

Um die verfügbaren Verbindungsarten, PC-Schnittstellen und Steckplätze zu enumerieren, ändern Sie folgenden Programmcode:

```
public static void EnumerateConnectionModesOfPLC(DeviceItem deviceItem)
{
    OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
    if (onlineProvider == null)
    {
        return; // Only cpu device items can provide OnlineProvider service
    }
    // Accessing connection configuration object
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    // Now access connection configuration members
    foreach (ConfigurationMode mode in configuration.Modes)
    {
        Console.WriteLine("Mode name:{0}", mode.Name);
        foreach (ConfigurationPcInterface pcInterface in mode.PcInterfaces)
        {
            Console.WriteLine("PcInterface name:{0}", pcInterface.Name);
            Console.WriteLine("PcInterface number:{0}", pcInterface.Number);
            foreach (ConfigurationTargetInterface targetInterface in
pcInterface.TargetInterfaces)
            {
                Console.WriteLine("TargetInterface:{0}", targetInterface.Name);
            }
        }
    }
}
```

Sie können auch über den Namen auf eine Verbindungsart und eine PC-Schnittstelle zugreifen:

```
public static ConfigurationTargetInterface
GetTargetInterfaceForOnlineConnection(OnlineProvider onlineProvider)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find("PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    ConfigurationTargetInterface slot = pcInterface.TargetInterfaces.Find("2 X3");
    return slot;
}
```

Um die an einer PC-Schnittstelle verfügbaren Adressen von Subnetzen und Gateways zu enumerieren, ändern Sie folgenden Programmcode:

```
public static void EnumeratingPCInterfaceSubnetsAndGateways(ConfigurationPcInterface pcInterface)
{
    foreach (ConfigurationSubnet subnet in pcInterface.Subnets)
    {
        Console.WriteLine("Subnet name:{0}", subnet.Name);
        foreach (ConfigurationGateway gateway in subnet.Gateways)
        {
            //Get the name of the gateway:
            Console.WriteLine("Gateway name:{0}", gateway.Name);
            //Get the IP address of each gateway:
            foreach (ConfigurationAddress gatewayAddress in gateway.Addresses)
            {
                Console.WriteLine("Gateway Address:{0} has {1}", gatewayAddress.Name,
gatewayAddress.Address);
            }
        }
    }
}
```

Sie können auch über den Namen oder die IP-Adresse auf Subnetze und Gateways zugreifen:

```
public static void AccessSubnetAndGatewayOfPCInterface(ConfigurationPcInterface pcInterface)
{
    ConfigurationSubnet subnet = pcInterface.Subnets.Find("PN/IE_1");
    ConfigurationAddress subnetAddress = subnet.Addresses.Find("192.168.0.1");
    ConfigurationGateway gateway = subnet.Gateways.Find("Gateway 1");
    ConfigurationAddress gatewayAddress = gateway.Addresses.Find("192.168.0.2");
}
```

Programmcode: Verbindungsparameter festlegen

Hinweis

Durch Festlegen der Verbindungsparameter werden alle zuvor festgelegten Verbindungsparameter überschrieben. Wenn Sie die Verbindungsparameter bereits direkt im TIA Portal festgelegt haben, brauchen Sie `ApplyConfiguration` nicht aufzurufen. Besteht bereits eine Online-Verbindung mit einem PLC, während `ApplyConfiguration` aufgerufen wird, wird eine Ausnahme ausgelöst.

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Ändern Sie den folgenden Programmcode, um Steckplatzparameter festzulegen:

```
public static void SetConnectionWithSlot(OnlineProvider onlineProvider)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find(@"PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    // or network pc interface that is connected to plc
    ConfigurationTargetInterface slot = pcInterface.TargetInterfaces.Find("2 X3");
    configuration.ApplyConfiguration(slot);
    // After applying configuration, you can go online
    onlineProvider.GoOnline();
}
```

Ändern Sie den folgenden Programmcode, um Gateway-Adressparameter einzustellen:

```
public static void SetConnectionWithGatewayAddress(OnlineProvider onlineProvider, string
subnetName, string gatewayAddressName)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find(@"PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    // or network pc interface that is connected to plc
    ConfigurationSubnet subnet = pcInterface.Subnets.Find(subnetName);
    ConfigurationAddress gatewayAddress = subnet.Addresses.Find(gatewayAddressName);
    configuration.ApplyConfiguration(gatewayAddress);
    // After applying configuration, you can go online
    onlineProvider.GoOnline();
}
```

Ändern Sie den folgenden Programmcode, um Subnetz-Adressparameter einzustellen:

```
public static void SetConnectionWithSubnetAddress(OnlineProvider onlineProvider, string
subnetName)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find(@"PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    // or network pc interface that is connected to plc
    ConfigurationSubnet subnet = pcInterface.Subnets.Find(subnetName);
    ConfigurationAddressComposition addresses = subnet.Addresses;
    configuration.ApplyConfiguration(addresses[0]);
    // After applying configuration, you can go online
    onlineProvider.GoOnline();
}
```

7.19.3 Zugriff auf den Fingerabdruck zum schnellen Vergleich von Stationen

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe AUTOHOTSPOT
- Ein Projekt ist geöffnet.
Siehe AUTOHOTSPOT
- Die PLC ist offline.

Anwendung

Mit TIA Portal Openness können Sie FingerprintData für unterschiedliche Aspekte der PLC-Gerätekonfiguration abrufen, um einen schnellen Vergleich von Stationen durchzuführen. Hierzu verwenden Sie den Service FingerprintDataProvider, den Sie über ein vorgegebenes TIA Portal abrufen können. Über den Aufruf GetService wird eine Instanz von FingerprintDataProvider zurückgegeben. Andernfalls wird null zurückgegeben.

Programmcode: Abrufen von FingerprintDataProvider aus dem TIA Portal

```
TiaPortal tia = new TiaPortal(TiaPortalMode.WithUserInterface);
FingerprintDataProvider fingerprintDataProvider =
tia.GetService<FingerprintDataProvider>();
if (fingerprintDataProvider != null)
{
    ...
}
```

Parameter der Methode FingerprintDataProvider

Um die FingerprintData eines Geräts abzurufen, müssen Sie die Methode GetFingerprintData von FingerprintDataProvider aufrufen.

Im FingerprintDataProvider werden die folgenden Attribute unterstützt:

Parametername	Typ	Beschreibung
configurationAddress	Siemens.Engineering.Online.Configuration	Adresse des Geräts, für das die FingerprintData abgerufen werden sollen.
onlineConfigurationDelegate	Siemens.Engineering.Online	Delegate, der aufgerufen wird, um die Konfiguration vor dem Abrufen der FingerprintData zu prüfen.

Konfigurationsadresse

Sie müssen für GetFingerprintData ein Objekt ConfigurationAddress bereitstellen. Das Adressobjekt wird verwendet, um eine Verbindung zu dem Gerät aufzubauen, für das die FingerprintData abgerufen werden sollen. Das Objekt ConfigurationAddress muss in der ConnectionConfiguration des FingerprintDataProvider erstellt werden.

Sie können beispielsweise folgenden Code verwenden, um ein Adressobjekt zu erstellen:

```
...
ConnectionConfiguration configuration = fingerprintDataProvider.Configuration;
ConfigurationMode configurationMode = configuration.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = configurationMode.PcInterfaces.Find("Intel (R)
Ethernet Connection I217-LM", 1);
// Create an address. This "ConfigurationAddress" is used as parameter for getting the
fingerprintData
ConfigurationAddress fingerprintAddress = pcInterface.Addresses.Create("192.68.0.1");
...
```

Die Konfiguration enthält eine Liste unterstützter Modes. Einer der Modes muss zum Abrufen von Fingerabdrücken ausgewählt werden. Der ausgewählte ConfigurationMode enthält eine Liste aller lokalen PCInterfaces, die den ausgewählten Mode unterstützen. Sie müssen eine der Schnittstellen auswählen. Die gewünschte Adresse kann in der Adressensammlung der ausgewählten ConfigurationPcInterface erstellt werden.

Programmcode: FingerprintData abrufen

Sie können die FingerprintData der Station abrufen, indem Sie die Aktion GetFingerprintData aufrufen. Die folgenden Parameter werden unterstützt:

Parameter	Beschreibung
ConfigurationAddress	Die Adresse des Geräts
OnlineConfigurationDelegate	Der Rückruf zur Behandlung von Inhibits

```
...
OnlineConfigurationDelegate preConfigurationDelegate = PreConfigureFingerprint;
FingerprintDataResult result =
fingerprintDataProvider.GetFingerprintData(fingerprintAddress, preConfigurationDelegate);
// The fingerprints
FingerprintDataItemComposition allFingerprints = result.FingerprintDataItems;
foreach (FingerprintDataItem item in allFingerprints)
{
// do something with the fingerprint ...
}
internal void PreConfigureFingerprint(OnlineConfiguration onlineConfiguration)
{
...
}
```

Delegate für die Online-Konfiguration

Die möglichen Typen von Online-Konfigurationen werden nachstehend aufgeführt:

Konfigurationsname	Beschreibung und Eigenschaften
OnlineConfiguration	<ul style="list-style-type: none"> Basisklasse für andere Konfigurationen OnlineConfiguration.Message: Zeichenkette (schreibgeschützte Eigenschaft, die die Konfigurationsmeldung enthält)

Da dies die Basisklasse für alle anderen Konfigurationen ist, ist dieser Eigenschaftsadapter deshalb in allen Konfigurationen verfügbar.

Der Datentyp der Konfiguration wird nachstehend aufgeführt:

Konfiguration	Datentyp	Beschreibung und Aktion
OnlineConfiguration	OnlineReadAccessPassword	<p>Set password über Methode SetPassword(password:SecureString).</p> <p>Passwort eingeben, um Lesezugriff auf das Modul zu erhalten.</p>

Eine nicht behandelte Konfiguration, die den Zugriff auf den Fingerabdruck verhindern kann, verursacht eine EngineeringDelegateInvocationException und bricht die Aktion ab. Bei einer nicht behandelten Ausnahme im Delegate wird eine EngineeringDelegateInvocationException ausgelöst.

Beispiel für die Implementierung von PreConfigureFingerprint

```
private static void PreConfigureFingerprint(OnlineConfiguration onlineConfiguration)
{
    OnlineReadAccessPassword readAccess = onlineConfiguration as OnlineReadAccessPassword;
    if (readAccess != null)
    {
        string passWD = "passWD";
        var password = new SecureString();
        foreach (var c in passWD) password.AppendChar(c);
        moduleReadAccessPassword.SetPassword(password);
        return;
    }
    throw new NotSupportedException(); // Exception thrown in the delegate will cancel get
                                    fingerprints
}
```

FingerprintDataResult

Das von der Aktion GetFingerprintData zurückgegebene FingerprintDataResult liefert eine Zusammenstellung sämtlicher Fingerabdrücke für das Gerät.

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Sie können beispielsweise folgenden Code verwenden, um Fingerabdruckelemente zu untersuchen:

```
{
...
FingerprintDataResult result = dataProvider.GetFingerprintData(fingerprintAddress,
preConfigurationDelegate);
FingerprintDataItemComposition allFingerprints = result.FingerprintDataItems;
foreach (FingerprintDataItem item in allFingerprints)
{
string fingerprintDataIdentifier = item.FingerprintDataIdentifier;
string fingerprintDataValue = plcItem.FingerprintDataValue;
// Do something, e.g. store Identifier and Value in a file
}
}
```

Fehlerbehandlung innerhalb der Openness-Fingerabdrücke

Alle Fehler innerhalb der Openness-Fingerabdrücke werden als wiederherstellbare Anwenderausnahmen EngineeringTargetException zugeordnet. Deshalb führt kein Fehler in einer Openness-Aktion zu einem Absturz des TIA Portals.

Voraussetzung ist, dass der Client die Ausnahme verarbeitet. Das kann beispielsweise wie in folgendem Code geschehen:

```
TiaPortal currentTIA = ...;
FingerprintDataProvider dataProviderFingerprints =
currentTIA.GetService<FingerprintDataProvider>();
OnlineConfigurationDelegate configurationDelegate = OnlineConfigurationFingerprintData;
try
{
// The used addr refers a PLC300.
FingerprintDataResult dataResult = dataProviderFingerprints.GetFingerprintData(addrObj,
configurationDelegate);
// Try to get a list of all FingerprintDataItems. But a PLC300 supports no fingerprints.
FingerprintDataItemComposition dataItems = dataResult.FingerprintDataItems;
foreach (FingerprintDataItem searchItem in dataItems)
{
string valueToCompareIdent = searchItem.FingerprintDataIdentifier;
string valueToCompareValue = searchItem.FingerprintDataValue;
DoAnything(valueToCompareIdent, valueToCompareValue);
}
}
catch (EngineeringTargetException targetException)
{
// In case of the not supported PLC this exception will be catched.
Console.WriteLine(targetException.Message);
}
```

Helper-Klassen

Für dieses Beispiel sind zwei Klassen nötig. Sie können folgenden Programmcode verwenden, um Fingerabdruck-Datenpaare zu kapseln.

```
/// <summary>
/// Helperclass to handle a "Dictionary" as serializeable class object
/// </summary>
[Serializable]
public class FingerprintDataPairs : Dictionary<string, string>, ISerializable
{
    public FingerprintDataPairs() {}
    public FingerprintDataPairs(SerializationInfo si, StreamingContext context)
    {
        KeyValuePair<string, string>[] dataPair = si.GetValue("KeyValuePair",
        typeof(KeyValuePair<string, string>[])) as KeyValuePair<string, string>[];
        if (dataPair != null)
            foreach (KeyValuePair<string, string> pair in dataPair)
                this.Add(pair.Key, pair.Value);
    }
}
```

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Sie können folgendes Beispiel verwenden, um die Fingerabdruckdaten zu verarbeiten und einen Vergleich durchzuführen:

```

/// <summary>
/// Helperclass for handle the fingerprint data and compare two datasets.
/// </summary> public class FingerprintDataCompare
{
internal FingerprintDataPairs m_FingerprintDataPairs;
internal FingerprintDataPairs FingerprintDataPairSet
{
get
{
if (m_FingerprintDataPairs == null) m_FingerprintDataPairs = new FingerprintDataPairs();
return m_FingerprintDataPairs;
}
set
{
m_FingerprintDataPairs = value;
}
}
internal bool CompareDataSets(FingerprintDataPairs dataSet1, FingerprintDataPairs dataSet2)
{
foreach (string key in dataSet1.Keys)
{
if (dataSet2[key] != dataSet1[key])
return false;
}
return true;
}
internal void SaveFingerprintDataItems(FingerprintDataPairs fingerprintDataPairSet, string filename)
{
using (FileStream fs = new FileStream(filename, FileMode.Create))
{
BinaryFormatter formatter = new BinaryFormatter();
try
{
formatter.Serialize(fs, fingerprintDataPairSet);
}
catch (SerializationException e)
{
Console.WriteLine("Failed to write fingerprints. See: " + e.Message);
}
finally
{
fs.Close();
}
}
}
internal FingerprintDataPairs LoadFingerprintDataItems(string filename)
{
FingerprintDataPairs dataPair = null;
using(FileStream fs = new FileStream(filename, FileMode.Open))
{
BinaryFormatter formatter = new BinaryFormatter();
try
{
dataPair = (FingerprintDataPairs)formatter.Deserialize(fs);
}
}
}

```

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

```
        }
    catch (System.Runtime.Serialization.SerializationException e)
    {
        Console.WriteLine("Failed to read fingerprints. See: " + e.Message);
    }
    finally
    {
        fs.Close();
    }
}
return dataPair;
}
```

Fingerabdrücke vergleichen

Um einen Vergleich durchzuführen, können Sie das FingerprintDataResult verwenden.

```
...
// See chapter "FingerprintDataResult"
FingerprintDataResult fingerprintDataResult = dataProvider.GetFingerprintData(address,
preConfigurationDelegate);
...
```

Um die Fingerabdrücke zu speichern, können Sie die Helper-Klassen verwenden:

```
string pathDat = "FingerprintDataOverview.dat";
// Create a fingerprint comparer and fill the dataitems with the result of
//"GetFingerprintData".
FingerprintDataCompare fingerprintDataSet = new FingerprintDataCompare();
foreach (FingerprintDataItem item in fingerprintDataResult.FingerprintDataItems)
    fingerprintDataSet.FingerprintDataPairSet.Add(item.FingerprintDataIdentifier,
item.FingerprintDataValue);
fingerprintDataSet.SaveFingerprintDataItems(fingerprintDataSet.FingerprintDataPairSet,
pathDat);
```

Mit gespeicherten FingerprintDataItems und einem aktuellen FingerprintDataResult kann ein Vergleich der Fingerabdrücke durchgeführt werden:

```
...
string pathDat = "FingerprintDataOverview.dat";
FingerprintDataCompare fingerprintDataSet = new FingerprintDataCompare();
foreach (FingerprintDataItem item in fingerprintDataResult.FingerprintDataItems)
    fingerprintDataSet.FingerprintDataPairSet.Add(item.FingerprintDataIdentifier,
    item.FingerprintDataValue);
FingerprintDataPairs loadDataSet = new FingerprintDataPairs();
loadDataSet = fingerprintDataSet.LoadFingerprintDataItems(pathDat);
if (!fingerprintDataSet.CompareDataSets(loadDataSet,
    fingerprintDataSet.FingerprintDataPairSet))
{
// Here is place to implement user actions.
}
```

7.19.4 Zugriff auf CM DP als DP-Slave und Transferbereich

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts

Anwendung

Wenn ein CM DP gesteckt ist, können Sie mit TIA Portal Openness eine ET200SP PLC als DP-Slave konfigurieren. Außerdem ist es möglich, in TIA Portal Openness Transferbereiche an der DP-Schnittstelle zu erstellen, zu konfigurieren und zu löschen. Die Handhabung ist ähnlich der Handhabung von Transferbereichen an der PN-Schnittstelle, die unter Openness-Transferbereiche für PnP-Koppler beschrieben wird.

Die folgenden dynamischen Attribute werden an der DP-Schnittstelle des Geräteelements unterstützt:

Attributname	Datentyp	Zugriff	Schreibbar
DpUseForTestCommissioningRouting	Boolean	nur verfügbar, wenn als DP-Slave konfiguriert	r/w
DpWatchdog	Boolean	nur verfügbar, wenn als DP-Slave konfiguriert und DP-Master zugewiesen	r/w

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Programmcode: Transferbereiche erstellen

Ändern Sie folgenden Programmcode, um an der DP-Schnittstelle einen Transferbereich zu erstellen:

```
NetworkInterface cpuItf = CpuInterface.GetService<NetworkInterface>();
// Create TransferAreas
TransferAreaComposition transferAreas = cpuItf.TransferAreas;
// Simple TranferArea
TransferArea transferAreaExample = transferAreas.Create("Example TA MS",
TransferAreaType.MS);
```

Parameter von Transferbereichen

Die folgenden Parameter werden in Transferbereichen der DP-Slave-Konfiguration unterstützt:

Parametername	Datentyp	Zugriff
Name	string	read/write
Direction	enum	read/write
Comment	string	read/write
LocalToPartnerLength	Int32	read/write
PartnerToLocalLength	Int32	read/write
LocalAddresses	AddressComposition object	read
PartnerAddresses	AddressComposition object	read
PositionNumber	Int32	read
Type	enum	read

Programmcode: Transferbereich löschen

Ändern Sie folgenden Programmcode so, dass ein Transferbereich für die DP-Slave-Konfiguration gelöscht wird:

```
TransferArea transferAreaExample = transferAreas.Create("Example TA MS",
TransferAreaType.MS);
transferAreaExample.Delete();
```

7.19.5 PLC von R/H-System online setzen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts

Anwendung

Mit dem Dienst RHOnlineProvider können Sie entweder den primären PLC oder den Backup-PLC eines R/H-Systems online setzen.

Programmcode: Auf Dienst RHOnlineProvider von einem Gerät aus zugreifen

Ändern Sie folgenden Code, um auf RHOnlineProvider zuzugreifen:

```
Device device = project.Devices.Find("S7-1500R/H-System_1");
RHOnlineProvider rhOnlineProvider = device.GetService<RHOnlineProvider>();
```

Programmcode: Verbindungsparameter festlegen

Mit dem Objekt ConnectionConfiguration können Sie eine Verbindung zum Gerät herstellen. Der Zugriff ist möglich über die Eigenschaft Configuration von RHOnlineProvider. Weitere Informationen zur Herstellung der Verbindung finden Sie unter Auf Parameter einer Online-Verbindung zugreifen

Ändern Sie den folgenden Programmcode, um über den Namen eine Verbindungsart einzurichten und auf eine PC-Schnittstelle zuzugreifen:

```
ConnectionConfiguration connectionConfiguration = rhOnlineProvider.Configuration;
ConfigurationMode mode = connectionConfiguration.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("Broadcom NetXtreme Gigabit Ethernet", 1);
ConfigurationTargetInterface targetConfiguration = pcInterface.TargetInterfaces.Find("1 X1");
bool success = connectionConfiguration.ApplyConfiguration(targetConfiguration);
```

Hinweis

R/H-System besteht aus zwei PLCs, eine einzelne Verbindungskonfiguration wird Ihnen bereitgestellt.

Programmcode: R/H-System online setzen

Sie können entweder den primären PLC oder den Backup-PLC online setzen. Wenn versucht wird, beide Ziele gleichzeitig online zu setzen, gibt das System eine Ausnahme EngineeringTargetException zurück.

Ändern Sie folgenden Programmcode, um den primären PLC online zu setzen:

```
OnlineState onlineState = rhOnlineProvider.GoOnlineToPrimary();
```

Ändern Sie folgenden Programmcode, um den Backup-PLC online zu setzen:

```
OnlineState onlineState = rhOnlineProvider.GoOnlineToBackup();
```

Hinweis

Wenn Sie einen PLC eines R/H-Systems online setzen, dürfen Sie ein zuvor gespeichertes Passwort wiederverwenden.

Programmcode: Onlinestatus eines R/H-Systems ermitteln

Sie können mit den Eigenschaften PrimaryState und BackupState von RHOnlineProvider den Online-Verbindungsstatus eines primären PLC und eines Backup-PLC getrennt voneinander feststellen. Beide Eigenschaften geben die Enumeration OnlineState zurück. Weitere Informationen dazu, wie Sie den Onlinezustand eines PLC ermitteln, finden Sie unter Zustand eines PLC feststellen

Ändern Sie folgenden Programmcode, um den Zustand eines primären PLC und eines Backup-PLC zu ermitteln:

```
RHOnlineProvider rhOnlineProvider = ...;
OnlineState primaryState = rhOnlineProvider.PrimaryState;
OnlineState backupState = rhOnlineProvider.BackupState;
```

Programmcode: R/H-System offline setzen

Ändern Sie folgenden Programmcode, um ein R/H-System, das derzeit online ist, durch Aufruf der Methode RHOnlineProvider.GoOffline in den Offline-Zustand zu versetzen:

```
rhOnlineProvider.GoOffline();
```

7.19.6 Auf Softwarebehälter über primären PLC eines R/H-Systems zugreifen

Voraussetzungen

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts

Anwendung

Für den Zugriff auf einen Softwarebehälter können Sie den primären PLC eines R/H-Systems verwenden. Beispielsweise stellt das R/H-System Softwarebehälter für einen primären PLC zur Verfügung, der als PLC_1 dargestellt wird. Wenn Sie jedoch versuchen, auf einen Softwarebehälter für einen Backup-PLC zuzugreifen, der als PLC_2 dargestellt wird, gibt das System null zurück.

Die Besonderheiten eines Softwarebehälters und seine Software-Eigenschaften werden unter Auf Software-Ziel zugreifen beschrieben.

Programmcode: Auf Softwarebehälter zugreifen

Um auf Softwarebehälter über das primäre Gerät eines R/H-Systems zuzugreifen, ändern Sie den folgenden Programmcode:

```
foreach (DeviceItem deviceItem in rhDevice.DeviceItems)
{
    if (deviceItem.Name == "PLC_1")
    {
        SoftwareContainer softwareContainer = deviceItem.GetService<SoftwareContainer>();
        ... //Work with softwareContainer
    }
}
```

7.19.7 PLCs eines R/H-Systems laden

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts

Anwendung

Sie können mit der TIA Portal Openness-Anwendung sowohl in den primären PLC als auch in den Backup-PLC eines R/H-Systems laden. Sie können sowohl Hardware- als auch Softwarekomponenten des Systems laden. (Weitere Informationen finden Sie unter Hardware- und Softwarekomponenten ins PLC-Gerät laden)

Programmcode: RHDownloadProvider abrufen

Über RHDownloadProvider service können Sie aus einem Gerät in ein R/H-System laden.

Ändern Sie folgenden Programmcode, um RHDownloadProvider abzurufen:

```
...
Device device = project.Devices.Find("S7-1500R/H-System_1");
RHDownloadProvider rhDownloadProvider = device.GetService<RHDownloadProvider>();
...
```

Hinweis

Auf den Dienst DownloadProvider wird nicht im Zusammenhang mit CPUs zugegriffen, die zum R/H-System gehören.

Programmcode: IConfiguration abrufen

RHDownloadProvider stellt das Objekt ConnectionConfiguration über die Eigenschaft Configuration zur Verfügung, mit deren Hilfe die Verbindung zum Gerät konfiguriert wird.

Ändern Sie folgenden Programmcode, um das Objekt IConfiguration über ConnectionConfiguration an RHDownloadProvider abzurufen:

```
...
RHDownloadProvider rhDownloadProvider = device.GetService<RHDownloadProvider>();
ConnectionConfiguration connectionConfiguration = rhDownloadProvider.Configuration;
ConfigurationMode mode = connectionConfiguration.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("Broadcom NetXtreme Gigabit Ethernet", 1);
IConfiguration targetConfiguration = pcInterface.TargetInterfaces.Find("1 X1");
...
...
```

Hinweis

R/H Systeme bestehen aus zwei PLCs. Es wird nur ein Verbindungskonfigurationsobjekt bereitgestellt, das zum Laden sowohl in primäre PLCs als auch in Backup-PLCs verwendet werden kann.

Programmcode: In primäre CPU und Backup-CPU laden

Ändern Sie folgenden Programmcode, um durch Aufrufen von RHDownloadProvider.DownloadToPrimary in den primären PLC zu laden:

```
DownloadResult DownloadToPrimary(configuration, preDownloadConfigurationDelegate,
postDownloadConfigurationDelegate, downloadOptions);
```

Ändern Sie folgenden Programmcode, um durch Aufrufen von RHDownloadProvider.DownloadToBackup in den Backup-PLC zu laden:

```
DownloadResult DownloadToBackup(configuration, preDownloadConfigurationDelegate,
postDownloadConfigurationDelegate, downloadOptions);
```

Parameter der Methode RHDownloadProvider

Sowohl RHDownloadProvider.DownloadToPrimary als auch RHDownloadProvider.DownloadToBackup akzeptieren dieselben Parameter und geben ebenfalls ein DownloadResult zurück. Weitere Informationen zu den Details von IConfiguration, DownloadConfigurationDelegate, DownloadOptions und DownloadResult finden Sie unter Hardware- und Softwarekomponenten ins PLC-Gerät laden

Parametername	Typ	Beschreibung
configuration	Siemens.Engineering.Connection.IConfiguration	Konfiguration der Verbindung mit einem Gerät.
preDownloadConfigurationDelegate	Siemens.Engineering.Download.DownloadConfigurationDelegate	Delegate, der aufgerufen wird, um die Konfiguration vor dem Laden zu prüfen
postDownloadConfigurationDelegate	Siemens.Engineering.Download.DownloadConfigurationDelegate	Delegate, der aufgerufen wird, um die Konfiguration nach dem Laden zu prüfen
downloadOptions	Siemens.Engineering.Download.DownloadOptions	Download-Optionen

Sie haben je nach Zustand des R/H-Systems die Möglichkeit, über DownloadConfigurations für den Ladevorgang einen Stopp des Systems anzufordern. Deshalb werden zusätzlich zu der Konfiguration, die unter Hardware- und Softwarekomponenten ins PLC-Gerät laden

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

beschrieben ist, auch die folgenden Datentypen zur Unterstützung von RHDownload hinzugefügt.

Konfiguration	Datentyp	Aktion	Beschreibung
DownloadSelection-Configuration	StopHSystem	Set CurrentSelection:StopH-SystemSelections. Verfügbare Enumerationswerte: <ul style="list-style-type: none"> • NoAction (Keine Aktion) • StopHSystem (R/H-System stoppen) 	Die Module werden gestoppt, um das Laden in ein Gerät zu ermöglichen.
	StopHSystemOrModule	Set CurrentSelection:StopH-SystemOrModuleSelections. Verfügbare Enumerationswerte: <ul style="list-style-type: none"> • NoAction (Keine Aktion) • StopHSystem (R/H-System stoppen) • StopModule (Modul stoppen) 	Die Module werden gestoppt, um das Laden in ein Gerät zu ermöglichen.
	StartBackupModules	Set CurrentSelection:StartBackupModulesSelections. Verfügbare Enumerationswerte: <ul style="list-style-type: none"> • NoAction (Keine Aktion) • SwitchToPrimaryCpu (Auf primär wechseln) • StartModule (Modul starten) 	Module nach dem Laden ins Gerät starten.
	SwitchBackupToPrimary	Set CurrentSelection:SwitchBackupToPrimarySelections. Verfügbare Enumerationswerte: <ul style="list-style-type: none"> • NoAction (Keine Aktion) • SwitchToPrimaryCpu (Auf primär wechseln) 	Module nach dem Laden ins Gerät starten.

Programmcode: Ladekonfiguration in Rückrufen handhaben

Ändern Sie folgenden Programmcode in Aufrufe von DownloadToPrimary und DownloadToBackup während der Behandlung von Konfigurationen in Rückrufen:

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Beispiel für den Aufruf von Download

```

static void Main(string[] args)
{
    ...
    Project project = tiaPortal.Projects[0];
    Device device = project.Devices.Find("S7-1500R/H-System_1");
    RHDownloadProvider rhDownloadProvider = device.GetService<RHDownloadProvider>();
    ConnectionConfiguration connectionConfiguration = rhDownloadProvider.Configuration;
    ConfigurationMode mode = connectionConfiguration.Modes.Find("PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("Broadcom NetXtreme Gigabit
Ethernet", 1);
    IConfiguration targetConfiguration = pcInterface.TargetInterfaces.Find("1 X1");

    // Download to primary
    DownloadResult primaryDownloadResult =
    rhDownloadProvider.DownloadToPrimary(targetConfiguration,
    PreConfigureDownloadCallback,
    PostConfigureDownloadCallback,
    DownloadOptions.Hardware | DownloadOptions.Software);
    WriteDownloadResults(primaryDownloadResult);
    // Download to backup
    DownloadResult backupDownloadResult =
    rhDownloadProvider.DownloadToBackup(targetConfiguration,
    PreConfigureDownloadCallback,
    PostConfigureDownloadCallback,
    DownloadOptions.Hardware | DownloadOptions.Software);
    WriteDownloadResults(backupDownloadResult);
    ...
}
private static void PreConfigureDownloadCallback(DownloadConfiguration
downloadConfiguration)
{
    StopHSystem stopHSystem = downloadConfiguration as StopHSystem;
    if (stopHSystem != null)
    {
        stopHSystem.CurrentSelection = StopHSystemSelections.StopHSystem;
    }
    OverwriteTargetLanguages overwriteTargetLanguages = downloadConfiguration as
OverwriteTargetLanguages;
    if (overwriteTargetLanguages != null)
    {
        overwriteTargetLanguages.Checked = true;
    }
    AlarmTextLibrariesDownload alarmTextLibraries = downloadConfiguration as
AlarmTextLibrariesDownload;
    if (alarmTextLibraries != null)
    {
        alarmTextLibraries.CurrentSelection =
AlarmTextLibrariesDownloadSelections.ConsistentDownload;
        return;
    }
    CheckBeforeDownload checkBeforeDownload = downloadConfiguration as CheckBeforeDownload;
    if (checkBeforeDownload != null)
    {
        checkBeforeDownload.Checked = true;
        return;
    }
}

```

Beispiel für den Aufruf von Download

```

    }
    ConsistentBlocksDownload consistentBlocksDownload = downloadConfiguration as
ConsistentBlocksDownload;
    if (consistentBlocksDownload != null)
    {
        consistentBlocksDownload.CurrentSelection =
ConsistentBlocksDownloadSelections.ConsistentDownload;
        return;
    }
    OverwriteSystemData overwriteSystemData = downloadConfiguration as OverwriteSystemData;
    if (overwriteSystemData != null)
    {
        overwriteSystemData.CurrentSelection = OverwriteSystemDataSelections.Overwrite;
        return;
    }
}
private static void PostConfigureDownloadCallback(DownloadConfiguration
downloadConfiguration)
{
    StartModules startModules = downloadConfiguration as StartModules;
    if (startModules != null)
    {
        startModules.CurrentSelection = StartModulesSelections.StartModule;
        return;
    }
}
private static void WriteDownloadResults(DownloadResult result)
{
    Console.WriteLine("State:" + result.State);
    Console.WriteLine("Warning Count:" + result.WarningCount);
    Console.WriteLine("Error Count:" + result.ErrorCount);
    RecursivelyWriteMessages(result.Messages);
}
private static void RecursivelyWriteMessages(DownloadResultMessageComposition messages,
string indent = "")
{
    indent += "\t";
    foreach (DownloadResultMessage message in messages)
    {
        Console.WriteLine(indent + "DateTime: " + message.DateTime);
        Console.WriteLine(indent + "State: " + message.State);
        Console.WriteLine(indent + "Message: " + message.Message);
        RecursivelyWriteMessages(message.Messages, indent);
    }
}

```

7.19.8 Funktionen zum Laden von Daten ins PLC-Gerät

7.19.8.1 PC-System laden

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Sie haben mit Ihrer TIA Portal Openess-Anwendung ein Projekt geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Sie können PC-Station Plus und die SW-CPU unabhängig voneinander laden. Um die Konfiguration der PC-Station Plus zu laden, müssen Sie das Geräteelement stationmanager des Baugruppträgers des Geräts abrufen und den download provider als Dienst abrufen. Die SW-CPU selbst muss für den gleichen Vorgang als Geräteelement abgerufen werden.

Programmcode: Software- und Hardware-Komponenten laden

Sie können Software- und Hardware-Komponenten über den Dienst DownloadProvider, der über ein angegebenes DeviceItem angefordert werden kann, auf ein Gerät laden. Wenn ein DeviceItem ein ladbares Ziel darstellt, wird bei Aufruf von GetService call eine Instanz von DownloadProvider zurückgegeben. Andernfalls wird null zurückgegeben.

```
DeviceItem stationManager = dev.DeviceItems.First(p => p.PositionNumber == 0).DeviceItems.First(a => a.PositionNumber == 125);
DownloadProvider downloadProviderStationManager =
stationManager.GetService<DownloadProvider>();
if (downloadProviderStationManager == null)
{
    //no download is possible for PC-Station
}
DeviceItem swCpu = dev.DeviceItems.First(p => p.Name == "Software PLC_1");
DownloadProvider downloadProviderSwCpu = swCpu.GetService<DownloadProvider>();
```

Ändern Sie folgenden Programmcode, um Netzwerkparameter zu konfigurieren:

```
ConnectionConfiguration connConfig = downloadProviderStationManager.Configuration;
ConfigurationMode configurationMode = connConfig.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = configurationMode.PcInterfaces.Find("ASIX AX88179
USB 3.0 to Gigabit Ethernet Adapter", 1);
ConfigurationTargetInterface targetInterface = pcInterface.TargetInterfaces.Find("2 X2");
IConfiguration targetConfiguration = pcInterface.TargetInterfaces[0];
bool isConfigured = connConfig.ApplyConfiguration(targetInterface);
if (isConfigured)
...
...
```

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Es ist zu beachten, dass bei einem Neustart des Zielsystems nach dem Laden der Konfiguration von PC-Station Plus das Standardverhalten von Openness **Warten** ist. Das bedeutet, dass der Openness-Codefluss gestoppt wird, bis das Laden der ersten Stufe erfolgreich ist (und das Zielsystem neu gestartet ist) oder die Zeit überschritten oder ein Fehler aufgetreten ist.

Falls Sie nicht auf den Neustart warten wollen, können Sie eine Post-Ladeoption wählen, die sich `WaitOnReboot` nennt.

```
//Post Download Configuration Delegate
DownloadConfigurationDelegate postDownloadForPcStation = downloadConfiguration =>
{
    WaitOnReboot waitOnReboot = downloadConfiguration as WaitOnReboot;
    if (waitOnReboot != null)
    {
        //In case user does not want to wait...
        waitOnReboot.CurrentSelection = WaitOnRebootSelections.NoAction;
        //In case user wants to wait... This is the default option anyway...
        //waitOnReboot.CurrentSelection = WaitOnRebootSelections.Wait;
        return;
    }
};
```

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Sie müssen die SW-CPU weiterhin separat laden, aber es muss sichergestellt werden, dass zuvor die PC-Station geladen wurde.

```
DownloadResult downloadResult = null;
try
{
//WE FIRST DOWNLOAD PC-STATION
downloadResult = downloadProviderStationManager.Download(targetConfiguration,
preDownloadForPcStation, postDownloadForPcStation, DownloadOptions.Hardware);
if (DownloadResultState.Error != downloadResult.State)
{
Console.WriteLine("The download is successful for pc-station");
}
}
catch (EngineeringTargetInvocationException e)
{
Console.WriteLine("Exception Thrown, Message: " + e.Message.ToString());
}
downloadResult = null;
try
{
downloadResult = downloadProviderSwCpu.Download(targetConfiguration, preDownloadForSwCpu,
postDownloadForWinac, DownloadOptions.Hardware | DownloadOptions.Software);
if (DownloadResultState.Error != downloadResult.State)
{
Console.WriteLine("The download is successful for SW-CPU");
}
}
catch (EngineeringTargetInvocationException e)
{
Console.WriteLine("Exception Thrown, Message: " + e.Message.ToString());
}
```

Das Laden der SW-CPU ähnelt dem Laden eines regulären PC. Weitere Informationen zum Laden einer SW-CPU finden Sie unter Hotspot-Text (Seite 465).

Hinweis

Das Laden von F-aktivierten PLCs wird derzeit bei einer SW-CPU genau wie bei einer HW-CPU nicht unterstützt.

Eine Lademethode stößt auch die Übersetzung an. Es ist auch möglich, ein Geräteelement unabhängig zu übersetzen, und zwar mit dem folgenden Code-Snippet:

```
ICompilable compileServiceSwCpu = swCpu.GetService<ICompilable>();
ICompilable compileServiceStationManager = stationManager.GetService<ICompilable>();
CompilerResult compileResultStationManager = compileServiceStationManager.Compile();
CompilerResult compileResultSwCpu = compileServiceStationManager.Compile();
bool compileCheck = !compileResultStationManager.State.Equals(CompilerResultState.Error);
if (compileCheck != true)
{
...
}
```

Siehe auch

- [Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- [Projekt öffnen \(Seite 111\)](#)
- [Hardware- und Softwarekomponenten ins PLC-Gerät laden \(Seite 465\)](#)

7.19.8.2 Hardware- und Softwarekomponenten ins PLC-Gerät laden

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Sie haben mit Ihrer TIA Portal Openness-Anwendung ein Projekt geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Der Openness-Anwender kann über DownloadProvider (Zugriff über das DeviceItem) Software- und Hardwarekomponenten ins PLC-Gerät laden. Wenn ein DeviceItem ein ladbares Ziel darstellt, wird bei Aufruf von GetService eine Instanz von DownloadProvider zurückgegeben. Andernfalls gibt der Dienst null zurück.

Programmcode: DownloadProvider-Dienst aus einem Geräteelement abrufen

```
DeviceItem deviceItem = ...;
DownloadProvider downloadProvider = deviceItem.GetService<DownloadProvider>();
if (downloadProvider != null)
{
    ...
}
```

Parameter der Download-Methode

Um Daten in ein PLC-Gerät zu laden, ruft der Anwender die Methode Download von DownloadProvider auf. Die Download-Methode hat vier Parameter, bei denen es sich um das Objekt IConfiguration, zwei Delegates und DownloadOptions (Hardware, Software oder Hardware und Software) handelt.

Parametername	Typ	Beschreibung
configuration	Siemens.Engineering.Connection.IConfiguration	Konfiguration der Verbindung mit einem Gerät.
preDownloadConfigurationDelegate	Siemens.Engineering.Download.DownloadConfigurationDelegate	Aufzurufender Delegate zum Prüfen der Konfiguration vor dem Ladevorgang.

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Parametername	Typ	Beschreibung
postDownloadConfigurationDelegate	Siemens.Engineering.Download.DownloadConfigurationDelegate	Aufzurufender Delegate zum Prüfen der Konfiguration nach dem Ladevorgang.
downloadOptions	Siemens.Engineering.Download.DownloadOptions	Ladeoptionen.

- Der Openness-Ladevorgang wird nur unterstützt, wenn die Konfigurationen vom Anwender sachgemäß gehandhabt werden. Wenn die Konfiguration ungültig ist, wird EngineeringTargetInvocationException ausgelöst und der Ladevorgang abgebrochen. Die F-aktivierten PLCs werden für den Ladevorgang nicht unterstützt.
- Da die Übersetzung ein Teil des Ladevorgangs ist, ist es empfehlenswert, die Übersetzung vor dem Ladevorgang durchzuführen, um die Übersetzungsergebnisse zu analysieren.
- Openness unterstützt nur die Option Gesamtladen.

Parameter 1: IConfiguration

Der Anwender muss das Objekt IConfiguration als ersten Parameter der Download-Methode angeben. Es dient dazu, eine Verbindung zum angegebenen PLC-Gerät herzustellen. Die Schnittstelle IConfiguration wird von ConfigurationAddress und ConfigurationTargetInterface implementiert. Auf beide Objekte kann über die Instanz ConnectionConfiguration zugegriffen werden. Die Instanz ConnectionConfiguration kann über die Eigenschaft DownloadProvider.Connection: ConnectionConfiguration oder optional über OnlineProvider.Connection: ConnectionConfiguration ermittelt werden.

Die Konfigurierung des Objekts ConnectionConfiguration wird im Abschnitt AUTOHOTSPOT beschrieben.

```
...
DownloadProvider downloadProvider = null;
ConnectionConfiguration configuration = downloadProvider.Configuration;
ConfigurationMode configurationMode = configuration.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = configurationMode.PcInterfaces.Find("Intel(R)
Ethernet Connection I217-LM", 1);
IConfiguration targetConfiguration = pcInterface.TargetInterfaces[0];
...
...
```

Parameter 2 und 3: DownloadConfigurationDelegate

Der Openness-Anwender muss zwei Implementierungen von leeren DownloadConfigurationDelegate(DownloadConfiguration downloadConfiguration) bereitstellen. Der erste Delegate wird für Konfigurationen vor dem Ladevorgang aufgerufen und der zweite wird nach Abschluss des Ladevorgangs aufgerufen. Die Delegates werden für jede Konfiguration aufgerufen, bei der eine Aktion durch den Anwender erforderlich ist. Weitere Informationen zur Handhabung von Rückrufen finden Sie unter Unterstützung von Callbacks (Seite 476). Bestimmte Konfigurationen enthalten nur eine Information, weshalb keine Aktion durch den Anwender erforderlich ist.

Die möglichen Typen von Ladekonfigurationen werden nachstehend aufgeführt.

Konfigurationsname	Beschreibung und Eigenschaften
DownloadConfiguration	<ul style="list-style-type: none"> Basisklasse für alle Konfigurationen. Enthält die einzige Eigenschaft DownloadConfiguration.Message : string (schreibgeschützte Eigenschaft, die die Konfigurationsmeldung enthält)
DownloadSelectionConfiguration	<ul style="list-style-type: none"> Basisklasse für alle auswählbaren Konfigurationen. Enthält keine weiteren Eigenschaften. In allen von dieser Klasse abgeleiteten untergeordneten Klassen muss eine Auswahl bereitgestellt werden.
DownloadCheckConfiguration	<ul style="list-style-type: none"> Basisklasse für alle Konfigurationen, die geprüft oder nicht geprüft sein können. Enthält die einzige Eigenschaft DownloadCheckConfiguration.Checked: bool<string> (Les-/Schreib-Eigenschaft, die angibt, ob die Konfiguration geprüft oder ungeprüft ist)
DownloadPasswordConfiguration	<ul style="list-style-type: none"> Basisklasse für alle Konfigurationen, die ein Passwort für den Ladevorgang benötigen. Enthält eine einzige Methode zum Festlegen des Passworts. DownloadPasswordConfiguration.SetPassword (password: SecureString) : void

Der Datentyp der Konfigurationen wird nachstehend aufgeführt.

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Konfiguration	Datentyp	Beschreibung und Aktion
DownloadSelection-Configuration	StartModules	<p>Set CurrentSelection:StopModulesSelections. Verfügbare Enumerationswerte sind</p> <p>NoAction (No action)</p> <p>StartModule (Stop module)</p> <p>Diese Module werden zum Laden in ein Gerät gestoppt.</p>
	StopModules	<p>Set CurrentSelection:StartModulesSelections. Verfügbare Enumerationswerte:</p> <p>NoAction (No action)</p> <p>StopAll (Stop all)</p> <p>Diese Module werden nach dem Ladevorgang gestartet.</p>
	AllBlocksDownload	<p>Set CurrentSelection:AllBlocksDownloadSelections. Verfügbarer Enumerationswert ist</p> <p>DownloadAllBlocks (Download all blocks to the device)</p> <p>Lädt Software in das Gerät</p>
	OverwriteSystemData	<p>Set CurrentSelection:OverwriteSystemDataSelections. Verfügbare Enumerationswerte sind</p> <p>NoAction (No action)</p> <p>Overwrite (Download to device)</p> <p>Löscht und ersetzt die vorhandenen Systemdaten im Zielspeicherort.</p>
	ConsistentBlocksDownload	<p>Set CurrentSelection:ConsistentBlocksDownloadSelections. Verfügbarer Enumerationswert ist</p> <p>ConsistentDownload (Consistent download)</p> <p>Lädt die Software in das Gerät.</p>
	AlarmTextLibrariesDownload	<p>Set CurrentSelection:AlarmTextLibrariesDownloadSelections. Verfügbare Enumerationswerte sind</p> <p>ConsistentDownload (Consistent download)</p> <p>NoAction (No action)</p> <p>Lädt alle Alarmtexte und Textlistentexte.</p>
	ProtectionLevelChanged	<p>Set CurrentSelection:ProtectionLevelChangedSelections. Verfügbare Enumerationswerte sind</p> <p>NoChange (No change)</p> <p>ContinueDownloading (Continue downloading to the device)</p> <p>Der CPU-Schutz wird auf die nächstniedrigere Stufe gesetzt.</p>
	ActiveTestCanBeAborted	<p>Set CurrentSelection:ActiveTestCanBeAbortedSelections. Verfügbare Enumerationswerte sind</p> <p>NoAction (No action)</p> <p>AcceptAll (Accept all)</p> <p>Aktive Test- und Inbetriebnahmefunktionen werden während des Laudebetriebs des Geräts abgebrochen.</p>
	ResetModule	Set CurrentSelection:ResetModuleSelections Verfügbare Enumerationswerte sind

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Konfiguration	Datentyp	Beschreibung und Aktion
		NoAction (No action) DeleteAll (Delete all) Setzt das Modul zurück.
	LoadIdentificationData	Set CurrentSelection:LoadIdentificationDataSelections. Verfügbare Enumerationswerte sind LoadNothing (Load nothing) LoadData (Load data) LoadSelected (Load selected) Identifikationsdaten in die PROFINET IO-Geräte und ihre Module laden.
	DifferentTargetConfiguration	Set CurrentSelection:DifferentTargetConfigurationSelections. Verfügbare Enumerationswerte sind NoAction (No action) AcceptAll (Accept all) Gibt den Unterschied zwischen konfiguriertem Modul und Zielmodul (online) an
	InitializeMemory	Set CurrentSelection:InitializeMemorySelections. Verfügbare Enumerationswerte: NoAction (No action) AcceptAll (Accept all) Dieser Datentyp dient zum Initialisieren des Speichers.
	ExpandDownload	Set CurrentSelection: ExpandDownloadSelections. Verfügbare Enumerationswerte: NoAction (No action) Download (Download)} Das Laden muss über Ihre Auswahl hinaus erweitert werden.
DownloadCheckConfiguration	CheckBeforeDownload	Eigenschaft Set IsChecked:bool Prüft vor dem Laden in das Gerät.
	UpgradeTargetDevice	Eigenschaft Set IsChecked:bool Prüft die unterschiedlichen Projektversionen im konfigurierten Gerät und im Zielgerät (online).
	OverWriteHMIData	Eigenschaft Set IsChecked:bool Überschreibt die Objekte online.
	FitHMIComponents	Eigenschaft Set IsChecked:bool Komponenten mit einer anderen Version werden auf dem Zielgerät installiert.
	TurnOffSequence	Eigenschaft Set IsChecked:bool Schaltet die Sequenz vor dem Laden aus.
	OverwriteTargetLanguages	Eigenschaft Set IsChecked:bool Zum Unterscheiden der Einstellungen für Projekt und PLC-Programmierung
	DowngradeTargetDevice	Eigenschaft Set IsChecked:bool Zum Erwähnen der verschiedenen Datenformate in Online- und Offline-Projekten.

Konfiguration	Datentyp	Beschreibung und Aktion
DownloadPassword-Configuration	ModuleReadAccessPassword	Methode Set password via SetPassword(password:SecureString). Passwort eingeben, um Lesezugriff auf das Modul zu erhalten.
	ModuleWriteAccessPassword	Methode Set password via SetPassword(password:SecureString) . Passwort eingeben, um Schreibzugriff auf das Modul zu erhalten.
	BlockBindingPassword	Methode Set password via SetPassword(password:SecureString). Methode für die Konfiguration eines bausteingebundenen Passworts.

ACHTUNG

Beachten Sie bitte, dass die Ladekonfigurationen den Konfigurationen in den Dialogen mit der Ladevorschau und den Ladeergebnissen ähneln, wenn Sie mit der Benutzeroberfläche des TIA Portals arbeiten.

⚠️ WARNUNG

Der API-Benutzer ist verantwortlich für die Sicherheitsmaßnahmen zur Handhabung von Passwörtern durch Code.

Nicht verarbeitete Konfiguration, die den Ladevorgang verhindern kann, verursacht eine EngineeringTargetException und bricht den Ladevorgang ab. Bei einer nicht

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

verarbeiteten Ausnahme im Delegate wird eine `EngineeringDelegateInvocationException` ausgelöst.

Beispiel für die Implementierung von `PreDownloadDelegate`:

```
private static void PreConfigureDownload(DownloadConfiguration downloadConfiguration)
{
    StopModules stopModules = downloadConfiguration as StopModules;
    if (stopModules != null)
    {
        stopModules.CurrentSelection = StopModulesSelections.StopAll; // This selection
will set PLC into "Stop" mode
        return;
    }
    AlarmTextLibrariesDownload alarmTextLibraries = downloadConfiguration as
AlarmTextLibrariesDownload;
    if (alarmTextLibraries != null)
    {
        alarmTextLibraries.CurrentSelection =
AlarmTextLibrariesDownloadSelections.ConsistentDownload;
        return;
    }
    BlockBindingPassword blockBindingPassword = downloadConfiguration as
BlockBindingPassword;
    if(blockBindingPassword != null)
    {
        SecureString password = ...; // Get Binding password from a secure location
        blockBindingPassword.SetPassword(password);
        return;
    }
    CheckBeforeDownload checkBeforeDownload = downloadConfiguration as
CheckBeforeDownload;
    if (checkBeforeDownload != null)
    {
        checkBeforeDownload.Checked = true;
        return;
    }
    ConsistentBlocksDownload consistentBlocksDownload = downloadConfiguration as
ConsistentBlocksDownload;
    if (consistentBlocksDownload != null)
    {
        consistentBlocksDownload.CurrentSelection =
ConsistentBlocksDownloadSelections.ConsistentDownload;
        return;
    }
    ModuleWriteAccessPassword moduleWriteAccessPassword = downloadConfiguration as
ModuleWriteAccessPassword;
    if (moduleWriteAccessPassword != null)
    {
        SecureString password = ...; // Get PLC protection level password from a secure
location
        moduleWriteAccessPassword.SetPassword(password);
        return;
    }
    throw new NotSupportedException(); // Exception thrown in the delegate will cancel
download
}
```

Beispiel für die Implementierung von PostDownloadDelegate:

```
private static void PostConfigureDownload(DownloadConfiguration downloadConfiguration)
{
    StartModules startModules = downloadConfiguration as StartModules;
    if (startModules != null)
    {
        startModules.CurrentSelection = StartModulesSelections.StartModule; // Sets PLC in "Run" mode
    }
}
```

Parameter 4: DownloadOptions

Der Anwender muss die Ladeoptionen über DownloadOptions, durch enum gekennzeichnet, angeben. Dieser Parameter bestimmt den Typ des durchzuführenden Ladevorgangs, z. B. Hardware, Software oder Hardware und Software.

```
[Flags]
public enum DownloadOptions
{
    None = 0,      // Download nothing
    Hardware,     // Download hardware only
    Software       // Download software only
}
```

Wenn der Anwender sowohl Software als auch Hardware in das Gerät laden möchte, ist DownloadOptions.Hardware | DownloadOptions.Software als 4. Parameter der Methode Download zu übergeben.

DownloadResult

Das von der Aktion Download zurückgegebene DownloadResult liefert Rückmeldung über den Zustand der geladenen Objekte.

Beispiel für den Aufruf von Download

```
[STAThread]
static void Main()
{
    ...
    DownloadProvider downloadProvider = ...;
    IConfiguration targetConfiguration = ...;
    DownloadConfigurationDelegate preDownloadDelegate = PreConfigureDownload;
    DownloadConfigurationDelegate postDownloadDelegate = PostConfigureDownload;
    DownloadResult result = downloadProvider.Download(targetConfiguration,
preDownloadDelegate, postDownloadDelegate, DownloadOptions.Hardware |
DownloadOptions.Software);
    if (result.State == DownloadResultState.Error)
    {
        // Handle error state
    }
    WriteDownloadResults(result);
    ...
}
private static void PreConfigureDownload(DownloadConfiguration downloadConfiguration)
{
    ...
}
private static void PostConfigureDownload(DownloadConfiguration downloadConfiguration)
{
    ...
}
private void WriteDownloadResults(DownloadResult result)
{
    Console.WriteLine("State:" + result.State);
    Console.WriteLine("Warning Count:" + result.WarningCount);
    Console.WriteLine("Error Count:" + result.ErrorCount);
    RecursivelyWriteMessages(result.Messages);
}
private void RecursivelyWriteMessages(DownloadResultMessageComposition messages, string
indent = "")
{
    indent += "\t";
    foreach (DownloadResultMessage message in messages)
    {
        Console.WriteLine(indent + "DateTime: " + message.DateTime);
        Console.WriteLine(indent + "State: " + message.State);
        Console.WriteLine(indent + "Message: " + message.Message);
        RecursivelyWriteMessages(message.Messages, indent);
    }
}
```

Ändern Sie den folgenden Code, um durch Aufruf der entsprechenden Konfiguration in den PLC zu laden:

```
private static void DownloadNCU(Device ncu, ConfigurationTargetInterface configurationTargetInterface)
{
    DownloadConfigurationDelegate preDownloadDelegate = PreConfigureDownload;
    DownloadConfigurationDelegate postDownloadDelegate = PostConfigureDownload;
    DownloadProvider downloadProvider = null;
    foreach (var item in ncu.DeviceItems[0].DeviceItems)
    {
        downloadProvider = item.GetService<DownloadProvider>();
        if (downloadProvider != null)
        {
            break;
        }
    }
    downloadProvider.Configuration.ApplyConfiguration(configurationTargetInterface);
    IConfiguration targetConfiguration = configurationTargetInterface;
    downloadProvider.Download(targetConfiguration, preDownloadDelegate,
postDownloadDelegate, DownloadOptions.Hardware | DownloadOptions.Software);
}
```

7.19.8.3 PLC starten und stoppen

Voraussetzungen

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Der PLC ist offline.

Verwendung

Bei Interaktionen mit TIA Portal über Openness API kann es notwendig sein, die Betriebsart des PLC zu ändern. TIA Portal Openness bietet die Möglichkeit, den Betriebszustand des PLC entweder in Start oder in Stop zu ändern.

Programmcode

Ändern Sie folgenden Programmcode, um den Betriebszustand des PLC auf STOP einzustellen.

```
public void ConfigurePreDownload(DownloadConfiguration downloadConfiguration)
{
    StopModules stopModules = downloadConfiguration as StopModules;
    if (stopModules != null)
    {
        // Puts PLC in "Stop" mode
        stopModules.CurrentSelection = StopModulesSelections.StopAll;
    }
}
```

Ändern Sie folgenden Programmcode, um den Betriebszustand des PLC auf START einzustellen.

```
public void ConfigurePostDownload(DownloadConfiguration downloadConfiguration)
{
    StartModules startModules = downloadConfiguration as StartModules;
    if (startModules != null)
    {
        // Puts PLC in "Start" mode
        startModules.CurrentSelection = StartModulesSelections.StartModule;
    }
}
```

7.19.8.4 Unterstützung von Callbacks

Voraussetzungen

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Bestimmte API-Methoden erfordern während ihrer Ausführung eine Interaktion mit dem benutzerdefinierten Anwendungscode. Delegates dienen zur Handhabung dieser Callback-Aktionen im benutzerdefinierten Anwendungscode. Sie müssen eine Methode mit einer kompatiblen Signatur implementieren und als Parameter delegate an die Aktion übergeben. Für die Ausführung ruft das TIA Portal die implementierten Methoden auf.

Programmcode

```
// This delegate is declared in Siemens.Engineering.dll
public delegate void
Siemens.Engineering.Download.DownloadConfigurationDelegate(Siemens.Engineering.Download.Co
nfigurations.DownloadConfiguration configuration);
...
```

Beispiel für einen benutzerdefinierten Anwendungscode mit Verwendung und Implementierung von delegate:

```
[STAThread]
static void Main()
{
    ...
    DownloadProvider downloadProvider = ...;
    IConfiguration targetConfiguration = ...;
    DownloadConfigurationDelegate preDownloadDelegate = PreConfigureDownload;
    DownloadConfigurationDelegate postDownloadDelegate = PostConfigureDownload;
    DownloadResult result = downloadProvider.Download(targetConfiguration,
preDownloadDelegate, postDownloadDelegate, DownloadOptions.Hardware |
DownloadOptions.Software);
    ...
}

//This method will be called back by TIA Portal
private static void ConfigurePreDownload(DownloadConfiguration downloadConfiguration)
{
    // Work with the parameter
}

//This method will be called back by TIA Portal
private static void ConfigurePostDownload(DownloadConfiguration downloadConfiguration)
{
    // Work with the parameter
}
```

Hinweis

Das Attribut **STAThread** stellt sicher, dass die Delegates im Haupt-Thread der Ausführung aufgerufen werden.

7.19.8.5 PLC durch Passwort schützen

Voraussetzungen

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Der PLC ist offline.

Verwendung

Bei Interaktionen mit TIA Portal über Openness API kann es notwendig sein, den Schutzgrad des PLC zu ändern. TIA Portal Openness bietet die Möglichkeit, den PLC durch ein Passwort zu schützen. Das Passwort kann sowohl für lesegeschützte als auch für schreibgeschützte PLCs eingestellt werden.

Programmcode

Ändern Sie folgenden Programmcode für lesegeschützte PLCs.

```
public void ConfigurePreDownload(DownloadConfiguration downloadConfiguration)
{
    ModuleReadAccessPassword moduleReadAccessPassword = downloadConfiguration
asModuleReadAccessPassword;
    if (moduleReadAccessPassword != null)
    {
        SecureString password = ...; // Get password from a secure location
        moduleReadAccessPassword.SetPassword(password); // enter the password to gain
full access
    }
}
```

Ändern Sie den folgenden Programmcode für schreibgeschützte PLCs:

```
public void ConfigurePreDownload(DownloadConfiguration downloadConfiguration)
{
    ModuleWriteAccessPassword moduleWriteAccessPassword = downloadConfiguration
asModuleWriteAccessPassword;
    if (moduleWriteAccessPassword != null)
    {
        SecureString password = ...; // Get password from a secure location
        moduleWriteAccessPassword.SetPassword(password); // enter the password to gain full
access
    }
}
```

⚠️ WARNUNG

Der API-Benutzer ist verantwortlich für die Sicherheitsmaßnahmen zur Handhabung von Passwörtern durch Code.

7.19.8.6 Handhabung bausteingegebundener PLC-Passwörter

Voraussetzungen

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Der PLC ist offline.

Verwendung

TIA Portal Openness unterstützt die Datenbindung von Passwörtern für Kundenanwendungen. TIA Portal Openness bietet dem Kunden einen Weg, ein bausteingegebundenes Passwort anzugeben. Zum Beispiel kann ein bausteingegebundenes Passwort auf der Klasse DownloadPasswordConfiguration durch Aufrufen der Methode SetPassword konfiguriert werden.

Hinweis

Wenn Sie den Ladevorgang mit einem Passwort schützen wollen, muss bei jedem Aufruf der Download-Funktion ein Passwort angegeben werden. Das gilt unabhängig davon, ob das Gerät bereits konfiguriert ist. Nach erfolgreicher Eingabe des Passworts für eine gegebene Konfiguration werden alle nachfolgenden Aufrufe von SetPassword ignoriert.

Programmcode

Ändern Sie folgenden Programmcode:

```
public void ConfigurePreDownload(DownloadConfiguration downloadConfiguration)
{
    DownloadPasswordConfiguration downloadPasswordConfiguration = downloadConfiguration as
DownloadPasswordConfiguration;
    if(downloadPasswordConfiguration != null &&
downloadPasswordConfiguration.Message.Contains("block_1"))
    {
        SecureString password = ...; // Get password from a secured location
        downloadPasswordConfiguration.SetPassword(password);
    }
}
```

7.19.9 Funktionen für den Zugriff auf den PLC-Dienst

7.19.9.1 Einstellung der Zugriffsstufen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Mit der TIA Portal Openness API können Sie über die Hardware PlcAccessLevelProvider an der Deviceitem CPU des Geräts auf die Einstellung der Zugriffsstufen zugreifen.

Diese Funktion bietet Aktionen zum Festlegen/Zurücksetzen des Passworts und ein modelliertes Attribut, um die Einstellung der Zugriffsstufen festzulegen/auszulesen.

- Void SetPassword (accessLevelType PlcProtectionAccessLevel, Passwort SecureString)
- Void ResetPassword (accessLevelType PlcProtectionAccessLevel)
- Attribut PlcProtectionAccessLevel

Im Allgemeinen ist die Handhabung über Openness ähnlich wie die in der TIA Portal UI.

- PlcProtectionAccessLevel kann an S7-1200/1500 PLCs immer ausgelesen und festgelegt werden.
- Wenn für die Zugriffsstufe eine andere Einstellung als Vollzugriff festgelegt ist (oder Vollzugriff inkl. Fail-safe), muss ein Passwort für Vollzugriff eingerichtet werden. Ansonsten kommt es zu einem Übersetzungsfehler: "Passwort darf nicht leer sein"

- Es ist nicht zulässig, das gleiche Passwort für verschiedene Zugriffsstufen festzulegen.
- Es können nur Passwörter für höhere Stufen als die festgelegte Zugriffsstufe eingerichtet / zurückgesetzt werden (sofern für PlcProtectionAccessLevel HMIAccess eingestellt ist, kann nur für ReadAccess und FullAccess ein Passwort eingerichtet werden).

Mögliche Ausnahmen:

- EngineeringTargetInvocationException: "**Gleiches Passwort ist bereits für andere Zugriffsstufe vorhanden**" – wird zurückgegeben, wenn der Benutzer versucht, das gleiche Passwort für unterschiedliche Zugriffsstufen festzulegen.
- EngineeringTargetInvocationException: "**Passwort kann für Zugriffsstufe nicht festgelegt werden**" – wird zurückgegeben, wenn der Benutzer versucht, ein Passwort für eine strengere Zugriffsstufe als die ausgewählte festzulegen.
- EngineeringTargetInvocationException: "**Passwort darf nicht leer sein**" – wird zurückgegeben, wenn der Benutzer versucht, ein leeres Passwort festzulegen.
- EngineeringTargetInvocationException: "**Zugriffsstufe ist nicht gültig**" – wird zurückgegeben, wenn der Benutzer versucht, an nicht fehlersicheren PLCs für die Zugriffsstufe FullAccessIncludingFailsafe festzulegen.

Zugriffsstufe

In Openness ist die Zugriffsstufe als Enumeration mit dem Namen PlcProtectionAccessLevel modelliert. Der Name der Enumerationsfelder basiert auf den Einträgen "Access Level" für S7-1200/1500 PLCs.

Name im TIA UI	Enumerations-Eintrag	Wert	Anmerkungen
-	None	0	Für Initialisierung der Enumeration. Dieser Wert darf nicht vom Openness-Benutzer festgelegt werden.
Vollzugriff (kein Schutz)	FullAccess	1	
Lesezugriff	ReadAccess	2	
HMI-Zugriff	HMIAccess	3	
Kein Zugriff	NoAccess	4	
Vollzugriff inkl. Fail-safe (kein Schutz)	FullAccessIncludingFailsafe	5	Nur bei fehlersicheren PLCs verfügbar.

Bei PLCs ist ein EOM-Attribut (modelliertes Attribut) mit dem Namen PlcProtectionAccessLevel vom Typ PlcProtectionAccessLevel-Enumeration verfügbar, um die Zugriffsstufe festzulegen/abzurufen.

Sie können das folgende Codebeispiel für PlcProtectionAccessLevel für "Kein Zugriff" verwenden:

```
DeviceItem S71500PLC = ...;
PlcAccessLevelProvider myPlcAccessLevelProvider =
S71500PLC.GetService<PlcAccessLevelProvider>();
myPlcAccessLevelProvider.PlcProtectionAccessLevel = PlcProtectionAccessLevel.NoAccess;
```

Passwort

Das Passwort ist aus Sicherheitsgründen als SecureString implementiert. Die Passwörter für die verschiedenen Zugriffsstufen können nur mit der entsprechenden Aktion festgelegt/ zurückgesetzt werden. Das Auslesen der Passwörter wird nicht unterstützt.

```
myPlcAccessLevelProvider.SetPassword(PlcProtectionAccessLevel.ReadAccess,  
someSecureString);  
myPlcAccessLevelProvider.ResetPassword(PlcProtectionAccessLevel.ReadAccess);
```

Siehe auch

Verbindung zum TIA Portal aufbauen (Seite 79)

Projekt öffnen (Seite 111)

7.19.9.2 Zugriff auf die OPC UA Server-Schnittstelle

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Sie können Serverschnittstellenobjekte erstellen und XML-Dateien importieren, die die OPC UA-Serverschnittstelle für den PLC definieren.

Mit der TIA Portal Openness API können Sie über den OpcUaProvider -Dienst auf die folgenden Funktionalitäten zugreifen:

- Hinzufügen eines OPC UA-Serverschnittstellenobjekts
- Enumerieren über die Serverschnittstelle des PLC
- Importieren einer Serverschnittstellendatei
- Exportieren einer vorhandenen Serverschnittstelle nach XML
- Aktivieren/Deaktivieren der Serverschnittstelle
- Löschen einer Serverschnittstelle

Programmcode

Sie können über einen OpcProvider-Dienst, den Sie über PlcSoftware erwerben können, mit OPC UA-Serverschnittstellen interagieren.

```
PlcSoftware software = ....;
OpcUaProvider provider = software.GetService<OpcUaProvider>();
if (provider != null)
{
    ....
}
```

Um die Navigation zu den OPC UA-Serverschnittstellen zu ermöglichen, ändern Sie folgenden Programmcode:

```
PlcSoftware software = ....;
OpcUaProvider provider = software.GetService<OpcUaProvider>();
if (provider != null)
{
    OpcUaCommunicationGroup commGroup = provider.CommunicationGroup;
    ServerInterfaceGroup serverInterfaceGroup = commFolder.ServerInterfaceGroup;
    ServerInterfaceComposition serverInterfaces = serverInterfaceGroup.ServerInterfaces;
}
```

Um die Elemente in der Sammlung ServiceInterfaceComposition zu enumerieren, ändern Sie folgenden Programmcode:

```
ServerInterfaceComposition serverInterfaces = ....;
foreach (ServerInterface serverInterface in serverInterfaces)
{
    ....;
}
```

Um ein OPC UA-Serverschnittstellenobjekt hinzuzufügen, ändern Sie folgenden Programmcode:

```
ServerInterfaceComposition serverInterfaces = ....;
if (serverInterfaces != null)
{
    ServerInterface serverInterface = serverInterfaces.Create(name);
}
```

Um die Eigenschaften eines OPC UA-Serverschnittstellenobjekts abzurufen und festzulegen, ändern Sie folgenden Programmcode:

```
ServerInterfaceComposition serverInterfaces = ....;
if (serverInterfaces != null)
{
    ServerInterface serverInterface = serverInterfaces.Create("New Interface");
    serverInterface.Author = "James Cornett";
}
```

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Um den Inhalt des Serverschnittstellenobjekts basierend auf einer XML-Datei festzulegen, ändern Sie folgenden Programmcode:

```
ServerInterfaceComposition serverInterfaces = ....;
if (serverInterfaces != null)
{
    ServerInterface serverInterface = serverInterfaces.Create(name);
    if (serverInterface != null)
    {
        serverInterface.Import(new FileInfo(importFilePath));
    }
}
```

Um den Inhalt des Serverschnittstellenobjekts in eine XML-Datei zu schreiben, ändern Sie folgenden Programmcode:

```
String exportFilePath = Path.Combine(Directory.GetCurrentDirectory(),
"ExportInterface.xml");
ServerInterface serverInterface = ...;
serverInterface.Export(new FileInfo(exportFilePath));
```

Um das Objekt aus der ServerInterfaceComposition zu entfernen, in der es enthalten ist, und die Zuordnung des Objekts aufzuheben, ändern Sie folgenden Programmcode:

```
ServerInterface serverInterface = ...;
serverInterface.Delete();
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.19.9.3 Auf Software-Prüfsumme zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Sie haben mit Ihrer TIA Portal Openness-Anwendung ein Projekt geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Mit der Openness API können Sie auf die Software-Prüfsumme einer PLC-Station zugreifen.

Programmcode

Für den Zugriff rufen Sie die Instanz von PlcSoftware ab und laden eine Dienstinstanz von PlcChecksumProvider.

```
PlcSoftware plc = ...;
//get PlcSoftware instance
PlcChecksumProvider checksumProvider =
plc.GetService<PlcChecksumProvider>();
```

Falls der PLC die Prüfsummenberechnung nicht unterstützt, gibt der GetService -Aufruf null zurück.

An der PlcChecksumProvider-Instanz kann die Software-Prüfsumme wie folgt erreicht werden:

```
string softwareChecksum = checksumProvider.Software;
```

Das Software-Attribut ist schreibgeschützt und es gibt null zurück, wenn das Programm nicht übersetzt ist.

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.19.9.4 PC-Schnittstelle zuordnen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Für die Schnittstellenzuordnung müssen Sie den PclInterfaceAssignment-Dienst von Siemens.HW.Features auf der Geräteelementschnittstelle abrufen.

Wenn die PC-Station eine Version <= 2.0 hat, geben die Methoden dieses Diensts null zurück und die Schnittstellenzuordnung ist nicht möglich bei Schnittstellen, die nur der PC-Station

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

oder nur Windows oder keinem von beiden zugeordnet sind. Bei Schnittstellen, die einer SW-CPU zugeordnet sind, ist dies wieder möglich.

Hinweis

Beim CAx-Austausch ist die Unterstützung der PC-Station sehr begrenzt. Es werden nur PC-Stationen mit Version 1.0 und dem Standardwert für InterfaceAssignment unterstützt. Das liegt daran, dass diese beiden Werte sich auf die Gerätestruktur auswirken (d. h. was eingebaut ist, was wo gesteckt werden kann). Da jedoch die Parameter nicht zur Datei AutomationML gehören, werden die Werte beim Import nicht gesetzt. Folglich ist die Struktur des IPC nicht passend und die anderen Module können nicht gesteckt werden.

Programmcode: Schnittstellenzuordnung

Für die Schnittstellenzuordnung ändern und verwenden Sie folgenden Programmcode:

```
DeviceItem swCpu = ...;
DeviceItem myInterface = ...;
PcInterfaceAssignment provider = myInterface.GetService<PcInterfaceAssignment>();
//user has to give device item sw cpu in case he/she wants to assign to it. (in the future,
there could be multi sw-cpu cases)
provider.AssignInterface(PcInterfaceAssignmentMode.SoftwarePlc, swCpu);
...
provider.AssignInterface(PcInterfaceAssignmentMode.PcStation);
...
provider.AssignInterface(PcInterfaceAssignmentMode.None);
...
```

Um Informationen über den aktuellen Zuordnungsmodus einer Schnittstelle abzurufen, ändern und verwenden Sie folgenden Programmcode:

```
PcInterfaceAssignment provider = myInterface.GetService<PcInterfaceAssignment>();
PcInterfaceAssignmentMode mode = provider.PcInterfaceAssignmentMode;
switch(mode)
{
    case PcInterfaceAssignmentMode.None: Console.WriteLine("Assigned to none or windows-
only.");
    break;
    case PcInterfaceAssignmentMode.PcStation: Console.WriteLine("Assigned to pc-station.");
    break;
    case PcInterfaceAssignmentMode.SoftwarePlc: Console.WriteLine("Assigned to SoftwarePlc: "
+ provider.SoftwarePlc.Name);
    break;
}
```

Programmcode: Konfiguration von Hardware-Ressource und IPC-Erweiterung

Mit dem folgenden Code können Sie auch die richtige Eigenschaft 'IPC-Erweiterung und Hardware-Ressource' über den PclInterfaceAssignment-Dienst auswählen:

```
DeviceItem myInterface = ...;
PcInterfaceAssignment provider = myInterface.GetService<PcInterfaceAssignment>();
//this method returns a subset of IpcExpansion values which are available for the IPC that
the interface is plugged in. IEnumerable<string> ipcExpansionChoices =
provider.GetAvailableIPCExpansions ();
string myChoice;
foreach(var choice in ipcExpansionChoices)
{
//user must select the desired value depending on his/her configuration
if (choice.Contains("3yxx1"))
{
    myChoice = choice;
    break;
}
}
provider.IpcExpansion = myChoice;
//after assigning IpcExpansion value, user may assign HardwareResource value using the
following enumeration. provider.HardwareResource = HardwareResource.X101;
//OR
//myInterface.SetAttribute("HardwareResource", HardwareResource.X101);
//myInterface.SetAttribute("IpcExpansion", mychoice);
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.19.10 Hardware, Software und Dateien in PLC-Gerät laden

Voraussetzungen

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe AUTOHOTSPOT
- Sie haben mit Ihrer TIA Portal Openness-Anwendung ein Projekt geöffnet.
Siehe AUTOHOTSPOT

Verwendung

Der Openness-Anwender kann über StationUploadProvider (auf das von einem project aus zugegriffen wird) eine Station in ein Projekt laden. Das Laden in eine DeviceGroup wird nicht unterstützt. Wenn ein project zur Durchführung eines Ladevorgangs verwendet wird, gibt der Dienst bei Aufruf von GetService eine Instanz von StationUploadProvider zurück. Andernfalls wird null zurückgegeben.

Programmcode: StationUploadProvider-Dienst von einem Projekt abrufen

```
Project myProject = ...;
StationUploadProvider uploadProviderForProject =
myProject.GetService<StationUploadProvider>();
if (uploadProviderForProject != null)
{
    ...
}
```

Parameter der Upload-Methode

Um das Laden in ein PLC-Gerät durchzuführen, ruft der Anwender die Upload-Methode StationUpload von StationUploadProvider. auf. Die Parameter dieser Upload-Methode sind ConfigurationAddress und UploadConfigurationDelegate . Da mit der Methode StationUpload Software, Hardware und Dateien geladen werden, sind die UploadOptions optional.

Parametername	Typ	Beschreibung
configurationAddress	Siemens.Engineering.Connection.ConfigurationAddress	Adresse des zu ladenden Geräts
uploadConfigurationDelegate	Siemens.Engineering.Upload.UploadConfigurationDelegate	Delegate, der aufgerufen wird, um die Konfiguration vor dem Laden zu prüfen
uploadOptions	Siemens.Engineering.Upload.UploadOptions	Upload-Optionen

Parameter 1: ConfigurationAddress

Der Anwender muss das Objekt ConfigurationAddress für die Upload-Methode angeben. Das Adressobjekt dient dazu, eine Verbindung zum angegebenen PLC-Gerät herzustellen, in das geladen werden soll. Das Objekt ConfigurationAddress muss in der ConnectionConfiguration des StationUploadProvider. erstellt werden. Die Configuration enthält eine Liste unterstützter Modes. Einer der Modes muss zur Verwendung beim Laden ausgewählt werden. Der ausgewählte ConfigurationMode enthält eine Liste aller lokalen PciInterfaces, die den ausgewählten Mode unterstützen. Eine dieser Schnittstellen muss ausgewählt werden. Die gewünschte Adresse kann in der Address -Sammlung der ausgewählten ConfigurationPciInterface erstellt werden.

Ändern Sie zum Erstellen eines Adressobjekts den folgenden Code:

```
...
StationUploadProvider uploadProvider = null;
...
ConnectionConfiguration configuration = uploadProvider.Configuration;
ConfigurationMode configurationMode = configuration.Modes.Find("PN/IE");
ConfigurationPcInterface pcInterface = configurationMode.PcInterfaces.Find("Intel(R)
Ethernet Connection I217-LM", 1);
//Create an address. This "ConfigurationAddress" is used as parameter for upload."
ConfigurationAddress uploadAddress = pcInterface.Addresses.Create("192.68.0.1");
...
```

Upload des Projekts

Der Anwender kann den Stationsupload durch Aufrufen der Aktion StationUpload starten.

Die folgenden Parameter sind obligatorisch:

- ConfigurationAddress: Die Adresse des Geräts, in das geladen werden soll
- UploadConfigurationDelegate: Der Rückruf zur Behandlung von Lade-Inhibits

```
...
StationUploadProvider uploadProvider = null;
Device uploadedObject = null;
...
UploadConfigurationDelegate preUploadDelegate = PreConfigureUpload;
UploadResult result = uploadProvider.StationUpload(uploadAddress, preUploadDelegate);
// The uploaded device
uploadedObject = resultUploadedStation;
if (uploadedObject == null)
{
...
}
internal void PreConfigureUpload(UploadConfiguration uploadConfiguration)
{
...
}
```

Parameter2: UploadConfigurationDelegate

Der Openness-Anwender muss eine Implementierung von void UploadConfigurationDelegate (UploadConfiguration uploadConfiguration) bereitstellen. Der Delegate wird für Konfigurationen vor dem Laden aufgerufen. Der Delegate wird für jede Konfiguration aufgerufen, bei der eine Aktion durch den Anwender erforderlich ist. Weitere Informationen zur Handhabung von Rückrufen finden Sie unter AUTOHOTSPOT. Bestimmte Konfigurationen enthalten nur eine Information, weshalb keine Aktion durch den Anwender erforderlich ist.

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Die möglichen Typen von Ladekonfigurationen werden nachstehend aufgeführt:

Konfigurationsname	Beschreibung und Eigenschaften
UploadConfiguration	<ul style="list-style-type: none"> Basisklasse für alle Konfigurationen. Sie enthält Informationen im Message-Attribut Enthält die einzige Eigenschaft UploadConfiguration.Message : string (schreibgeschützte Eigenschaft, die die Konfigurationsmeldung enthält)
UploadPasswordConfiguration	<ul style="list-style-type: none"> Abgeleitet von UploadConfiguration Basisklasse für alle Konfigurationen, die ein Passwort für den Ladevorgang benötigen. Enthält eine einzige Methode zum Festlegen des Passworts. UploadPasswordConfiguration.SetPassword (password: SecureString) : void - Passwort festlegen
UploadSelectionConfiguration	<ul style="list-style-type: none"> Abgeleitete Klasse von UploadConfiguration Enthält keine weiteren Eigenschaften.

Der Datentyp der Konfigurationen wird nachstehend aufgeführt.

Konfiguration	Datentyp	Beschreibung und Aktion
UploadPasswordConfiguration	ModuleReadAccessPassword	Methode Set password via SetPassword(password:SecureString). Passwort eingeben, um Lesezugriff auf das Modul zu erhalten.
	PasswordReadAccess	Methode Set password via SetPassword(password:SecureString). Passwort für den Softwareupload in klassischen PLCs eingeben, um Lesezugriff auf das Modul zu erhalten.
UploadSelectionConfiguration	UploadMissingProducts	Set CurrentSelection:UploadMissingProductsSelections Verfügbare Enumerationswerte: TryUpload (Consistent upload) NoAction (No action) Eine Auswahl für den Upload angeben.

Unterstützung eines fehlersicheren Passworts ist nicht notwendig. Für den Lesezugriff beim Laden in einen F-PLC wird kein Passwort benötigt.

Eine nicht behandelte Konfiguration, die den Ladevorgang verhindern kann, verursacht eine EngineeringTargetInvocationException und bricht den Ladevorgang ab.

Bei einer nicht behandelten Ausnahme im Delegate wird eine EngineeringDelegateInvocationException ausgelöst.

Beispiel für die Implementierung von PreUploadDelegate:

```

private static void PreConfigureUpload(UploadConfiguration UploadConfiguration)
{
    ModuleReadAccessPassword moduleReadAccessPassword = UploadConfiguration as
ModuleReadAccessPassword;
    if (moduleReadAccessPassword != null)
    {
        string passWD = "passWD";
        var password = new SecureString();
        foreach (var c in passWD)
            password.AppendChar(c);

        moduleReadAccessPassword.SetPassword(password);
        return;
    }

    ModuleWriteAccessPassword moduleWriteAccessPassword = UploadConfiguration as
ModuleWriteAccessPassword;
    if (moduleWriteAccessPassword != null)
    {
        string passWD = "passWD";
        var password = new SecureString();
        foreach (var c in passWD)
            password.AppendChar(c);

        moduleWriteAccessPassword.SetPassword(password);
        return;
    }
    ...

    throw new NotSupportedException(); // Exception thrown in the delegate will cancel upload
}

```

Parameter3: UploadOptions

Die Upload-Optionen kann der Anwender nicht angeben. Die Upload-Optionen lauten: "Hardware", "Software", "Hardware und Software" sowie "Hardware, Software und Dateien".

UploadResult

Das von der Upload-Aktion zurückgegebene Ergebnis UploadResult liefert Rückmeldung über den Zustand der geladenen Objekte.

- UploadResult.Message: UploadResultMessageComposition - Zusammensetzung von UploadResultMessage

Die folgenden Attribute werden unterstützt:

Attribute	Beschreibung
ErrorCount	int-Wert der Fehler beim Upload
State	UploadResultState mit den möglichen Werten: Success, Information, Warning und Error

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Attribute	Beschreibung
UploadedStation	Eine Device-Instanz der geladenen Station
WarningCount	Anzahl der Warnungen beim Laden als int

Die UploadResultMessage enthält:

- UploadResultMessage.Messages : UploadResultMessageComposition - Zusammensetzung von UploadResultMessage

Die folgenden Attribute werden unterstützt:

Attribute	Beschreibung
DateTime	System.DateTime der erstellten Meldung.
ErrorCount	Ein int-Zähler für Fehler.
State	UploadResultState mit den möglichen Werten: Success, Information, Warning und Error
WarningCount	Anzahl der Warnungen beim Laden als int

Beispiel für den Aufruf von Upload

```
internal bool UploadPLC()
{
    ...
    UploadResult result = uploadProvider.StationUpload(uploadAddress, preUploadDelegate);
    ...
    PrintAllMessages(result.Messages, 0);
    ...
}

internal void PrintAllMessages(UploadResultMessageComposition messages, int level)
{
    if (messages == null)
        return;

    if (level == 0)
        Console.WriteLine("\n");
    foreach (UploadResultMessage message in messages)
    {
        string messageOut = message.Message.PadLeft(message.Message.Length + level, '\t') + "\n";
        Console.WriteLine(messageOut);

        if ((message.Messages != null) && (message.Messages.Count > 0))
            PrintAllMessages(message.Messages, level+1);
    }
}
```

7.19.11 PLC-Software vergleichen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe AUTOHOTSPOT
- Sie haben mit Ihrer TIA Portal Openness-Anwendung ein Projekt geöffnet.
Siehe AUTOHOTSPOT

Verwendung

Sie haben die folgenden Möglichkeiten, um die Abweichung zwischen der Software zweier Geräte zu ermitteln:

- Vergleich der Software zweier konfigurierter PLCs
- Vergleich der Software eines PLC und der Projektbibliothek
- Vergleich der Software eines PLC und der globalen Bibliothek
- Vergleich der Software eines PLC und der Masterkopie eines PLC
- Vergleich der Software eines konfigurierten PLC mit der Software eines verbundenen PLC im Status „online“

Signatur

Verwenden Sie die Methode `CompareTo` oder `CompareToOnline` für den Vergleich.

```
public CompareResult CompareTo (ISoftwareCompareTarget compareTarget)
```

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

```
public CompareResult CompareToOnline ()
```

Rückgabewert/Parameter	Funktion
CompareResult compareResult	Gibt das Vergleichsergebnis zurück: <ul style="list-style-type: none"> FolderContentsDifferent: Inhalt der verglichenen Ordner unterscheidet sich. FolderContentsIdentical: Inhalt der verglichenen Ordner ist identisch. ObjectsDifferent: Inhalt der verglichenen Objekte unterscheidet sich. ObjectsIdentical: Inhalt der verglichenen Objekte ist identisch. LeftMissing: Das Objekt ist nicht in dem Objekt enthalten, von dem aus der Vergleich gestartet wurde. RightMissing: Das Objekt ist nicht in dem Objekt enthalten, das verglichen wird. CompareIrrelevant: Der Vergleich dieser 2 Objekte ist irrelevant. FolderContainsDifferencesOwnStateDifferent: Der Inhalt des Ordners weist einen oder zwei Unterschiede auf, der eigene Zustand des Ordners weicht ab. FolderContentEqualOwnStateDifferent: Der Ordnerinhalt ist identisch, der eigene Zustand des Ordners weicht ab.
ISoftwareCompareTarget compareTarget	Liste vergleichbarer Objekte.

Programmcode

Ändern Sie den folgenden Programmcode, um das Vergleichsergebnis auszugeben:

```
private static void WriteResult(CompareResultElement compareResultElement, string indent)
{
    Console.WriteLine("{0} <{1}> <{2}> <{3}> <{4}> ",
        indent,
        compareResultElement.LeftName,
        compareResultElement.ComparisonResult,
        compareResultElement.RightName,
        compareResultElement.DetailedInformation);
    WriteResult(compareResultElement.Elements, indent);
}
private static void WriteResult (IEnumerable<CompareResultElement> compareResultElements,
string indent)
{
    indent += " ";
    foreach (CompareResultElement compareResultElement in compareResultElements)
    {
        WriteResult(compareResultElement, indent);
    }
}
```

Ändern Sie den folgenden Programmcode, um die Software von Geräten zu vergleichen:

```
private static void CompareTwoOfflinePlcs(PlcSoftware plcSoftware0, PlcSoftware
plcSoftware1)
{
    if (plcSoftware0 != null && plcSoftware1 != null)
    {
        CompareResult compareResult = plcSoftware0.CompareTo(plcSoftware1);
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

Ändern Sie den folgenden Programmcode, um die Software eines PLC mit der Projektbibliothek zu vergleichen:

```
private static void ComparePlcToProjectLibrary(Project project, PlcSoftware plcSoftware)
{
    if (project != null && plcSoftware != null)
    {
        CompareResult compareResult = plcSoftware.CompareTo(project.ProjectLibrary);
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Ändern Sie den folgenden Programmcode, um die Software eines PLC mit der globalen Bibliothek zu vergleichen:

```
private static void ComparePlcToGlobalLibrary(PlcSoftware plcSoftware, GlobalLibrary globalLibrary)
{
    if (plcSoftware != null && globalLibrary != null)
    {
        CompareResult compareResult = plcSoftware.CompareTo(globalLibrary);
        WriteResult(compareResult.RootElement, String.Empty);
    }
}
```

Ändern Sie den folgenden Programmcode, um die Software eines PLC mit einer Masterkopie zu vergleichen:

```
private static void ComparePlcToMasterCopy(Project project, PlcSoftware plcSoftware)
{
    if (project != null && plcSoftware != null)
    {
        CompareResult compareResult =
plcSoftware.CompareTo(project.ProjectLibrary.MasterCopyFolder.MasterCopies[0]);
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

Ändern Sie den folgenden Programmcode, um die Software eines PLC mit der Software eines verbundenen PLC zu vergleichen:

```
private static void ComparePlcToOnlinePlc(PlcSoftware plcSoftware)
{
    if (plcSoftware != null)
    {
        CompareResult compareResult = plcSoftware.CompareToOnline();
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

7.19.12 PLC-Hardware vergleichen

Voraussetzungen

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen
- Sie haben mit Ihrer TIA Portal Openness-Anwendung ein Projekt geöffnet.
Siehe Öffnen eines Projekts

Anwendung

Mithilfe von TIA Portal Openness können Sie die Hardware zweier PLC-Geräte vergleichen.

Signatur

Verwenden Sie die Methode CompareTo für den Vergleich der beiden Hardwareobjekte.

CompareResult CompareTo (IHardwareCompareTarget compareTarget);

Rückgabewert/-parameter	Funktion
CompareResult compare result	Das Vergleichsergebnis zurückgeben: <ul style="list-style-type: none"> • FolderContainsDifferencesOwnStateDifferent: Der Inhalt des Ordners weist einen oder zwei Unterschiede auf, der eigene Zustand des Ordners weicht ab. • FolderContentEqualOwnStateDifferent: Der Ordnerinhalt ist identisch, der eigene Zustand des Ordners weicht ab.
IHardwareCompareTarget compareTarget	Das Vergleichsziel, wegen dessen der Hardwarevergleich durchgeführt wird. Darf nicht null sein.

Wenn der Parameter compareTarget null ist und ein Hardwarevergleich versucht wird, wird die Siemens.Enginnering.EngineeringTargetInvocationExceptions ausgelöst.

Programm

Ändern Sie den folgenden Programmcode, um das Vergleichsergebnis auszugeben:

```
...
CompareResult compareResult = plc_1.CompareTo(plc_2);
CompareResultState resultState = compareResult.RootElement.ComparisonResult;
if (resultState == CompareResultState.FolderContainsDifferencesOwnStateDifferent)
{
    // Folder contents have one or more differences, folder's own state is different:
    // May occur if the plc has a different subordinate element, e.g., a local module, and
    // the plc itself is different, e.g., in a parameter
}
else if (resultState == CompareResultState.FolderContentEqualOwnStateDifferent)
{
    // Folder content is the same, folder's own state is different:
    // May occur if a folder-style module, e.g., FM 351, has equal subordinate elements but
    // the module itself is different, e.g., in a parameter
}
else if (resultState == CompareResultState.FolderContentsIdentical)
{
    ...
}
```

7.19.13 Online-Verbindung zur PLC aufbauen oder trennen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe AUTOHOTSPOT
- Ein Projekt ist geöffnet.
Siehe AUTOHOTSPOT
- Alle Geräte werden enumeriert.
Siehe AUTOHOTSPOT.

Verwendung

Sie können die Online-Verbindung zu einer PLC herstellen oder eine bestehende Online-Verbindung trennen.

Programmcode

Um die Online-Verbindung zu einer PLC herzustellen oder zu trennen, ändern Sie folgenden Programmcode:

```
public static void SetOnlineConnection(DeviceItem deviceItem)
{
    OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
    if (onlineProvider == null) { return; }
    // Go online
    if (onlineProvider.Configuration.IsConfigured)
    {
        onlineProvider.GoOnline();
    }
    // Go offline
    onlineProvider.GoOffline();
}
```

Sie können auch die Online-Verbindungen zu allen verfügbaren PLCs in einem Projekt herstellen oder trennen.

```
public static void SetOnlineConnectionForAllPLCs(Project project)
{
    foreach (Device device in project.Devices)
    {
        foreach (DeviceItem deviceItem in device.DeviceItems)
        {
            OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
            if (onlineProvider != null)
            {
                // Establish online connection to PLC:
                onlineProvider.GoOnline();

                // ...
                // Disconnect online connection to PLC:
                onlineProvider.GoOffline();
            }
        }
    }
}
```

7.19.14 Projektsprache der PLC zuordnen

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen

Anwendung

Sie können mithilfe von TIA Portal Openness dem Webserver Projektsprachen zuordnen und Sprachen einer S7-1500 PLC anzeigen. Hierzu ist ein dynamisches Attribut vom Typ StructuredData notwendig.

Für den Zugriff auf die Einstellungen in mehreren Sprachen wurde das neue dynamische Attribut MultilingualSupport vom Typ TableData hinzugefügt. Dieses Attribut kann in Zeilen aufgeteilt werden. In den einzelnen Zeilen kann die zugeordnete Projektsprache jeweils mit dem dynamischen Attribut ProjectLanguage festgelegt oder gelesen werden.

Programmcode

Ändern Sie folgenden Programmcode, um die Projektsprache der PLC zuzuordnen:

```
//To Set the Project Language
LanguageSettings languageSettings = project.LanguageSettings;
LanguageAssociation activeLanguages = languageSettings.ActiveLanguages;
DeviceItem Plc = ...;
Tabledata multilingualSupportTable = Plc.GetAttribute("MultilingualSupport");
StructuredDataComposition structuredDataComp = multilingualSupportTable.Rows; // This will
return a collection of Structured Data.
StrcturedData firstRow= structuredDataComp[0]; //First row
Language activeGermanLanguage = activeLanguages.Find(CultureInfo.GetCultureInfo("de-DE"));
firstRow.SetAttribute("ProjectLanguage", activeGermanLanguage.Culture);
//To Get the Languages of device displayed
var assignedToGerman = firstRow.GetAttribute("DisplayLanguage");
```

7.19.15 Beobachtungs- und Forcetabellen für Webserver- und PLC-Display zuordnen

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen

Anwendung

Mit TIA Portal Openness können Sie bereits erstellte Beobachtungs- und Forcetabellen dem Webserver zuordnen. Bei der Beobachtungstabelle und der Forcetabelle handelt es sich um Software, die im Softwarecontainer gefunden, erstellt und gelöscht werden kann. Sie können mit TIA Portal Openness Beobachtungs- und Forcetabellen exportieren/importieren. Zum Export/Import von Beobachtungs- und Forcetabellen siehe Beobachtungs- und Forcetabelle exportieren/importieren

Für die Zuordnung von Beobachtungs- und Forcetabellen zum Webserver wird der Dienst WatchAndForceTableAccessManager am DeviceItem der PLC verwendet.

Dieser Dienst enthält zwei Navigatoren für Beobachtungs- und Forcetabellen.

- WatchtableAccessRules {WatchtableAccessRuleComposition}
- ForcetableAccessRules {ForcetableAccessRuleComposition}

Der Navigator WatchTableAccessRules stellt die Zusammensetzung WatchTableAccessRuleComposition zur Verfügung, die Objekte vom Typ WatchTableAccessRule enthält. Die Zusammensetzung ist standardmäßig leer. Für die WatchTableAccessRule -Objekte sind die Aktionen "Finden" und "Erstellen" definiert.

AccessRule für WatchTable & ForceTable

- **PlcWatchTable** watchtable {get;} gibt die Beobachtungstabelle des Softwareobjekts zurück. Um eine zugeordnete Beobachtungstabelle auszutauschen, muss der Benutzer die Zuordnung der aktuellen Tabelle entfernen (mit WatchTableAccessRule.Delete()) und eine neue Tabelle hinzufügen (mit WatchTableAccessRule.Create()).
- **WatchAndForceTableAccess** access {get; set;} gibt die Webserver-Zugriffsstufe (d. h. Lesen oder Lesen/Schreiben) zurück bzw. legt sie fest

Name in der TIA UI	Wert	enum-Beschreibung in Openness
-	0	None
Lesen	1	Read
Lesen/Schreiben	2	Write

In folgenden Fällen werden Fehler ausgelöst:

- Wenn der Webserver nicht aktiviert wurde (WebserverActivate == FALSE), wird eine EngineeringTargetInvocationException ausgelöst, deren Fehlerdetails besagen, dass eine WatchTableAccessRule nur hinzugefügt werden kann, wenn der Webserver aktiviert ist.
- Wenn der Benutzer versucht, eine Beobachtungstabelle mit WatchAndForceTableAccess.None hinzuzufügen, wird eine ConfigOpenessUserException ausgelöst, deren Fehlerdetails besagen, dass WatchAndForceTableAccess.None als Zugriffsstufe nicht zulässig ist

Programmcode: Beobachtungstabelle zuordnen und deren Zuordnung aufheben

Ändern Sie folgenden Programmcode, um eine Beobachtungstabelle auf dem Webserver zu suchen:

```
WatchAndForceTableAccessManager mngr =
deviceItem.GetService<WatchAndForceTableAccessManager>();
WatchTableAccessRuleComposition watchTableCmp = mngr.WatchTableAccessRules;
WatchTableAccessRule accessRule1 = watchTableCmp.Find(watchTable1);
```

Hinweis

Es wird null zurückgegeben, wenn die betreffende Beobachtungstabelle nicht mit einer WatchTableAccessRule auf dem Webserver verknüpft ist

Ändern Sie folgenden Programmcode, um auf dem Webserver eine neue Beobachtungstabelle mit Lesezugriff zu erstellen:

```
WatchAndForceTableAccessManager mngr =
deviceItem.GetService<WatchAndForceTableAccessManager>();
WatchTableAccessRuleComposition watchTableCmp = mngr.WatchTableAccessRules;
watchTableCmp.Create(watchTable1, WatchAndForceTableAccess.Read);
```

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Ändern Sie folgenden Programmcode, um eine vorhandene WatchTableAccessRule zu löschen und die Zuordnung der Beobachtungstabelle zum Webserver aufzuheben:

```
WatchTableAccessRule whatchtable= watchTableCmp.Find(watchTable1);  
whatchtable.Delete();
```

In folgenden Fällen werden Fehler ausgelöst:

- Wenn der Webserver nicht aktiviert wurde (WebserverActivate == FALSE), wird eine EngineeringTargetInvocationException ausgelöst, deren Fehlerdetails besagen, dass eine WatchTableAccessRule nur hinzugefügt werden kann, wenn der Webserver aktiviert ist
- Wenn eine WatchtableAccessRule zu dieser Beobachtungstabelle bereits vorhanden ist, wird eine ConfigOpenessUserException ausgelöst, deren Fehlerdetails besagen, dass die Beobachtungstabelle bereits vorhanden ist.
- Wenn Sie versuchen, eine Beobachtungstabelle mit WatchAndForceTableAccess.None hinzuzufügen, wird eine ConfigOpenessUserException ausgelöst, deren Fehlerdetails besagen, dass WatchAndForceTableAccess.None als Zugriffsstufe nicht zulässig ist

Programmcode: Dem Webserver eine Forcetabelle zuordnen

Ändern Sie folgenden Programmcode, um eine Forcetabelle auf dem Webserver zu suchen:

```
WatchAndForceTableAccessManager mngr =  
deviceItem.GetService<WatchAndForceTableAccessManager>();  
ForceTableAccessRuleComposition forceTableCmp = mngr.ForceTableAccessRules;  
ForceTableAccess forceTable = forceTableCmp.Find(forceTable1);
```

Ändern Sie folgenden Programmcode, um auf dem Webserver eine PLC-Forcetabelle mit Lesezugriff zu erstellen:

```
ForceTableAccessRuleComposition forceTableCmp = mngr.ForceTableAccessRules;  
forceTableCmp.Create(forceTable1, WatchAndForceTableAccess.Read);
```

Beobachtungs- und Forcetabellen am PLC-Display

Sie können denselben Openess-Dienst WatchAndForceTableAccessManager verwenden, der am Submodul Display (DeviceItem) des PLC DeviceItem verfügbar ist. Für die Zuordnung von Beobachtungs- und Forcetabellen am PLC-Display wird dieselbe Webserver-Funktion wie oben beschrieben verwendet. Im Unterschied zum Webserver kann das Display nicht deaktiviert werden. Damit ist eine ähnliche Validierung wie bei WebserverActive hier nicht möglich.

7.19.16 Zugriff auf Webserver und OPC UA-Benutzerverwaltung

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen

Anwendung

Sie können mit TIA Portal Openness auf den Webserver und das OPC UA-Submodul von PLCs zugreifen. Es ist möglich, einen Benutzer bis zur Höchstzahl am Webserver und am OPC UA-Submodul von PLCs hinzuzufügen. Dieser Benutzer hat einen Benutzernamen und ein Passwort.

Benutzerverwaltung am Webserver

Sie können die folgenden Vorgänge wie z. B. das Hinzufügen eines Benutzers, das Löschen eines Benutzers und das Festlegen des Passworts am Webserver für alle unterstützten Geräte nur dann durchführen, wenn der Webserver am Modul aktiviert ist.

Wenn der Webserver nicht aktiviert ist, wird durch das Hinzufügen und Löschen von Benutzern und das Festlegen des Passworts eine EngineeringTargetInvocationException ausgelöst. Die Lesevorgänge sind trotzdem verfügbar.

Der Dienst WebServerUserManagement ist für die PLC-Instanz nur der unterstützten Geräte verfügbar. Für alle nicht unterstützten Geräte lautet der Dienst null.

```
WebServerUserManagement
{
  Navigators WebServerUsers
  {
    WebServerUserComposition
  }
}
```

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Der Dienst stellt einen Navigator namens WebServerUsers zur Verfügung, mit dessen Hilfe eine WebServerUserComposition bezogen werden kann, die die Instanzen von WebServerUser verwaltet.

```
WebServerUserComposition
{
    WebServerUser Find(string username);
    void Create(string username, WebserverUserPermissions permissions, SecureString password);
}
WebServerUser
{
    string UserName{get;}
    WebserverUserPermissions Permissions {get;set;}
    void Delete();
    void SetPassword(SecureString password);
}
```

Programmcode: Aktionen an der WebServerUserComposition

Ändern Sie das folgende Programm, um einen Webserver-Benutzer zu suchen:

```
WebServerUserComposition webServerUserComposition = WebServerUserManagement.WebServerUsers;
WebServerUser user1 = webServerUserComposition.Find("user1");
```

Ändern Sie folgenden Programmcode, um einen neuen Webserver-Benutzer mit Berechtigungen für Webserver-Benutzer zu erstellen:

```
WebServerUserComposition webServerUserComposition = WebServerUserManagement.WebServerUsers;
WebServerUser user1 = webServerUserComposition.Create("user1",
    WebserverUserPermissions.ReadTagStatus, someSecureString);
```

Ändern Sie folgenden Programmcode, um den Webserver-Benutzer zu löschen:

```
WebServerUserComposition webServerUserComposition = WebServerUserManagement.WebServerUsers;
WebServerUser user1 = webServerUserComposition.Find("user1");
user1.Delete();
```

Ändern Sie folgenden Programmcode, um das aktuelle Passwort des Benutzers durch das neue Passwort zu ersetzen:

```
WebServerUserComposition webServerUserComposition = WebServerUserManagement.WebServerUsers;
WebServerUser user1 = webServerUserComposition.Find("user1");
user1.SetPassword(someSecureString);
```

Benutzerverwaltung am OPC UA-Server

Sie können die folgenden Vorgänge wie z. B. das Hinzufügen eines Benutzers, das Löschen eines Benutzers und das Festlegen des Passworts am OPC UA-Server nur dann durchführen, wenn der OPC UA-Server aktiviert ist und der Benutzername und die Passwort-Authentifizierung aktiviert sind.

Wenn diese Voraussetzungen nicht erfüllt sind, wird durch den Vorgang eine `EngineeringTargetInvocationException` ausgelöst, deren Fehlerdetails besagen, dass die Authentifizierung aktiviert werden muss.

Der Dienst `OpcUaUserManagement` ist nur am OPC UA-Submodul der PLC verfügbar. Der Dienst ist für alle unterstützten Geräte verfügbar, bei denen das OPC UA-Submodul für das Geräteelement verfügbar ist. Andernfalls wird der Dienst mit dem Wert null zurückgegeben.

```
OpcUaUserManagement
{
    Navigators OpcUaUsers
    {
        OpcUaUserComposition
    }
}
```

Der Dienst stellt einen Navigator namens `OpcUaUsers` zur Verfügung, mit dessen Hilfe eine `OpcUaUserComposition` bezogen werden kann. Die Zusammensetzung verwaltet die Instanzen von `OpcUaUser`:

```
OpcUaUserComposition
{
    OpcUaUser Find(string username);
    void Create(string username, SecureString password);
}
OpcUaUser
{
    string UserName{get;}
    void Delete();
    void SetPassword(SecureString password);
}
```

Programmcode: Aktionen an der `OpcUaUserComposition`

Ändern Sie folgenden Programmcode, um einen `OpcUaUser` zu suchen:

```
OpcUaUserComposition opcUaUserComposition = opcUaUserManagement.OpcUaUsers;
OpcUaUser user1 = opcUaUserComposition.Find("user1");
```

Ändern Sie folgenden Programmcode, um einen neuen `OpcUaUser` zu erstellen:

```
OpcUaUserComposition opcUaUserComposition = opcUaUserManagement.OpcUaUsers;
opcUaUserComposition.Create("user1", someSecureString);
```

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Ändern Sie folgenden Programmcode, um den OpcUaUser zu löschen:

```
OpcUaUserComposition opcUaUserComposition = opcUaUserManagement.OpcUaUsers;
OpcUaUser user = opcUaUserComposition.Find("user1");
user.Delete();
```

Ändern und verwenden Sie folgenden Programmcode, um das aktuelle Passwort durch ein bereitgestelltes zu ersetzen:

```
OpcUaUserComposition opcUaUserComposition = opcUaUserManagement.OpcUaUsers;
OpcUaUser user = opcUaUserComposition.Find("user1");
user.SetPassword(someSecureString);
```

7.19.17 Zugriff auf Domänen-Einstellungen

Voraussetzungen

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts

Einleitung

Sie können mit TIA Portal Openness auf die vollständige Domänenverwaltung zugreifen. Die beiden folgenden neuen Dienste werden der Klasse Subnet hinzugefügt, damit auf die Domänenverwaltung über TIA Portal Openness zugegriffen werden kann:

- Dienst MrpDomainOwner
- Dienst SyncDomainOwner

Programmcode: Auf MrpDomainOwner zugreifen

Der Dienst MrpDomain liefert dem Navigator MrpDomains, was die MrpDomainComposition enthält. Die Zusammensetzung enthält Objekte vom Typ MrpDomain.

```
Subnet subnet = ...;
MrpDomainOwner mrpDomainOwner = subnet.GetService<MrpDomainOwner>();
MrpDomainComposition mrpDomainComposition = mrpDomainOwner.MrpDomains;
```

Ändern Sie folgenden Programmcode, um eine neue Domäne namens newMrpDomain zu erstellen:

```
int countOfMrpDomains = mrpDomainComposition.Count;
MrpDomain someMrpDomain = mrpDomainComposition.ElementAt(3);
MrpDomain firstMrpDomain = mrpDomainComposition.First();
MrpDomain byName = mrpDomainComposition.Find("DomainName");
// create a new domain
MrpDomain newMrpDomain = mrpDomainOwner.MrpDomains.Create("newMrpDomain");
```

Ändern Sie folgenden Programmcode, um Attributwerte einer MRP-Domäne zu lesen/zuschreiben:

```
NetworkInterface toBeAdded = ...;
newMrpDomain.SetAttribute("IsDefault", true);
newMrpDomain.SetAttribute("ManagerOutsideOfProjectActive", false);
var participants = firstMrpDomain.DomainParticipants;
int count = participants.Count;
participants.Add(toBeAdded);
foreach (NetworkInterface networkIf in participants)
{
    // do something at the interface
}
newMrpDomain.Delete();
```

Programmcode: Auf SyncDomainOwner zugreifen

Der Dienst stellt die SyncDomains des Navigators bereit, die die SyncDomainComposition enthalten. Diese Zusammensetzung enthält Objekte vom Typ SyncDomain.

```
Subnet subnet = ...;
SyncDomainOwner syncDomainOwner = subnet.GetService<SyncDomainOwner>();
SyncDomainComposition syncDomainComposition = syncDomainOwner.SyncDomains;
```

Ändern Sie folgenden Programmcode, um eine neue SyncDomain mit einem bestimmten Namen zu erstellen

```
int countOfSyncDomains = syncDomainComposition.Count;
SyncDomain someSyncDomain = syncDomainComposition.ElementAt(3);
SyncDomain firstSyncDomain = syncDomainComposition.First();
SyncDomain byName = syncDomainComposition.Find("DomainName");
// create a new domain
SyncDomain newSyncDomain = syncDomainOwner.SyncDomains.Create("newSyncDomain");
```

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Ändern Sie folgenden Programmcode, um den Attributwert von SyncDomain zu lesen/zuschreiben:

```
NetworkInterface toBeAdded = ...;
string convertedName = newSyncDomain.ConvertedName;
newSyncDomain.SetAttribute("IsDefault", true);
newSyncDomain.SetAttribute("HighPerformanceActive", true);
newSyncDomain.SetAttribute("FastForwardingActive", true);
var participants = firstSyncDomain.DomainParticipants;
int count = participants.Count;
participants.Add(toBeAdded);
foreach (NetworkInterface networkIf in participants)
{
// do something at the interface
}
newSyncDomain.Delete();
```

7.19.18 Display-Passwort festlegen

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen

Anwendung

Sie können mit TIA Portal Openness das Passwort zum Schutz des Displays festlegen. Sie können mit der Methode SetPassword(SecureString password) das Passwort zum Schutz des Displays festlegen, wobei der Parameter für das Passwort ein SecureString sein muss. Das Bestätigungsdatum wird in Openness nicht benötigt und ist daher nicht verfügbar.

Die Funktion ist verfügbar, wenn die folgenden Voraussetzungen erfüllt sind:

- Die PLC verfügt über eine Instanz "Display".
- Das "Display" unterstützt die Funktion zum Schutz durch ein "Passwort". Es gibt Fälle, z. B. eine Software-PLC, in denen "Display" verfügbar, aber kein Passwortschutz verfügbar ist.

Programmcode

Ändern Sie folgenden Programmcode, um das Geräteelement abzurufen, das Inhaber der Dienstinstanz ist. Hierzu kann die Eigenschaft OwnedBy verwendet werden.

```
DisplayProtection displayProtection = plcDisplay.GetService<DisplayProtection>();
var displayDeviceItem = displayProtection.OwnedBy;
```

Ändern Sie folgenden Programmcode, um das Passwort zum Schutz des Displays festzulegen:

```
displayProtection.SetPassword(someSecuredString);
//Sets the string someSecuredString as the CPU display protection password.
```

Hinweis

Software-Controller verfügen über ein Display-Submodul, unterstützen jedoch keinen Displayschutz. Deshalb ist in TIA Portal Openness die Funktion SetPassword am Display-Submodul von Software-Controllern nicht verfügbar.

7.19.19 Zugänglichkeit

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts

Einleitung

Sie können mit TIA Portal Openness die IP-Zugänglichkeit unterstützen. In TIA Portal Openness wird an der Geräteelement-CPU ein neues dynamisches Attribut PlcAccessCommunicationModule verfügbar. Das Attribut gehört zum Typ Objekt. Dadurch können Sie auf das CP-Objekt zugreifen, das Sie für die IP-Zugänglichkeit auswählen möchten, und das Objekt dem PlcAccessCommunicationModule zuordnen.

Wenn bereits ein CP ausgewählt wurde, können Sie das CP-Objekt abrufen, indem Sie den Wert von PlcAccessCommunicationModule abrufen.

Programmcode

```
DeviceItem Plc = ...;
object communicationModule = Plc.GetAttribute("PlcAccessCommunicationModule");
// cP contains the object reference of a plugged CP
Plc.SetAttribute("PlcAccessCommunicationModule", cP);
```

7.19.20 Modulbeschreibung aktualisieren

Voraussetzung

- TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen
- Ein Projekt ist geöffnet.
Siehe AUTOHOTSPOT

Anwendung

In TIA Portal Openness wird der Dienst ModuleDescriptionUpdater am Geräteelement dazu verwendet, die aktuelle Modulbeschreibung auf die neueste Version des Moduls zu aktualisieren. Um das Geräteelement abzurufen, das Inhaber der Dieninstanz ist, können Sie die Eigenschaft OwnedBy verwenden.

Mithilfe des Attributs CanUpdate können Sie anzeigen, ob eine neue ConfigObject-Version für dieses Deviceitem verfügbar ist.

Attribut	Datentyp	Beschreibung
CanUpdate	bool	TRUE: Eine neue Version ist verfügbar FALSE: Es gibt keine neue Version

Sie können mit der Aktion UpdateModuleDescription () die ConfigObject -Version eines Deviceitem aktualisieren.

Aktion	Rückgabewert	Beschreibung
UpdateModuleDescription	True	Das Deviceitem ist aktuell: <ul style="list-style-type: none"> • Das Deviceitem wurde erfolgreich aktualisiert • Das Deviceitem war bereits aktuell
	False	Das Deviceitem ist nicht aktuell: <ul style="list-style-type: none"> • Das Deviceitem konnte nicht aktualisiert werden

Programmcode

```
DeviceItem deviceItem = ...;
var descriptionUpdater = deviceItem.GetService<ModuleDescriptionUpdater>();
if (descriptionUpdater != null)
{
    if (descriptionUpdater.CanUpdate) //e.g. is update module version possible
    {
        bool result = descriptionUpdater.UpdateModuleDescription();
        .
        .
    }
}
```

7.19.21 Zertifikat verwalten

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts

Anwendung

Sie können mit TIA Portal Openness Zertifikate verwalten. Hierzu gehört z. B. das Erstellen und Löschen von Zertifikaten, der Export und Import von Zertifikaten, das Zuordnen von Zertifikaten und Aufheben der Zuordnung sowie das Abrufen der Zertifikats-ID.

Local Certificate Manager

Für Gerätezertifikate wird der Dienst LocalCertificateManager an der PLC des DeviceItem bereitgestellt. Der LocalCertificateManager verfügt über einen lokalen Speicher namens LocalCertificateStore für die Zertifikate. Darin werden die Zertifikate für die bestimmte PLC gespeichert. Um das Geräteelement abzurufen, das Inhaber der Dieninstanz ist, können Sie die Eigenschaft OwnedBy verwenden.

```
DeviceItem Plc= ...;
// Get local certificate manager
var localCertificateManager = Plc.GetService<LocalCertificateManager>();
// Disable global certificate manager
localCertificateManager.EnableGlobalCertificatesStore = false;
// Get local certificate store
var localCertificateStore = localCertificateManager.LocalCertificateStore;
```

Zertifikateverwaltung

Ändern Sie folgenden Programmcode, um ein Zertifikat zu erstellen:

```
/ Create templates
var templateTls = localCertificateStore.GetCertificateTemplate(CertificateUsage.Tls);
var templateWebserver =
localCertificateStore.GetCertificateTemplate(CertificateUsage.WebServer);
var templateOpcUaServer =
localCertificateStore.GetCertificateTemplate(CertificateUsage.OpcUaServer); //
...
//Template handling and configuration is handled later
// Create certificates
var certificateTls = localCertificateStore.Certificates.Create(templateTls);
var certificateWeb = localCertificateStore.Certificates.Create(templateWebserver);
var certificateOpcUa = localCertificateStore.Certificates.Create(templateOpcUaServer);
```

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Ändern Sie folgenden Programmcode, um ein Zertifikat zu löschen und zu exportieren:

```
var exportPath = ...;
// renew a certificate
Tls.Delete();
certificateTls = localCertificateStore.Certificates.Create(templateTlsNew);
certificateWeb.Export(new FileInfo(exportPath), CertificateExportFormat.Cer);
```

Ändern Sie folgenden Programmcode, um mit dynamischen Attributen dem Webserver und OPC UA-Server Zertifikate zuzuordnen:

```
DeviceItem opcUaSubmodule = ...;
//Find assigned certificates
var foundWebCertificate = Plc.GetAttribute("WebserverCertificate");
var foundOpcUaCertificate = opcUaSubmodule.GetAttribute("OpcUaServerCertificate");
//Assign certificates
certificatesPlc.SetAttribute("WebserverCertificate", certificateWeb);
opcUaSubmodule.SetAttribute("OpcUaServerCertificate", certificateOpcUa);
//Unassign certificates
certificatesPlc.SetAttribute("WebserverCertificate", null);
opcUaSubmodule.SetAttribute("OpcUaServerCertificate", null);
```

Ändern Sie folgenden Programmcode, um Zertifikate in den lokalen Zertifikatsspeicher für eine PLC zu importieren:

```
var certificateWithoutPwd = ...;
var certificateWithPwd = ...;
var password = new SecureString();
// ...
// Import certificates
// Without password
var importedCertificate1 = certificates.Import(new FileInfo(certificateWithoutPwd));
//With password
var importedCertificate2 = certificates.Import(new FileInfo(certificateWithPwd), password);
```

Ändern Sie folgenden Programmcode, um die Zertifikats-ID abzurufen:

```
var certificateId = importedCertificate2.Id;
```

Ändern Sie folgenden Programmcode, um anzugeben, ob ein Zertifikat über einen privaten Schlüssel verfügt. Hierzu kann die boolesche Eigenschaft HasPrivateKey am Zertifikatsobjekt verwendet werden.

```
if(importedCertificate2.HasPrivateKey)
{
//Do something
}
```

Zertifikatsvorlage

Die Vorlagen dienen als Grundlage für die Erstellung von Zertifikaten. Neue Zertifikate können im LocalCertificateStore mit der Aktion GetCertificateTemplate(CertificateUsage) erstellt werden. Hierbei wird CertificateUsage mit den folgenden möglichen Werten enumeriert:

- None (nicht unterstützt)
- Tls
- WebServer
- OpcUaServer
- OpcUaClient
- OpcUaClientServer

```
// Create template
var certTemplate = localCertificateStore.GetCertificateTemplate(CertificateUsage.Tls);
```

In einer Zertifikatsvorlage können alle Eigenschaften eines Zertifikats festgelegt werden. Auf die Zertifikatsvorlage besteht Lese- und Schreibzugriff:

- Signatur: Der verwendete Signatur-Algorithmus gehört zum Typ SignatureAlgorithm, der mit den folgenden möglichen Werten enumeriert wird:
 - None (nicht unterstützt)
 - Sha1RSA
 - Sha256RSA

```
certTemplate.Signature = SignatureAlgorithm.Sha1RSA;
```

- SubjectAlternativeNames: Dies ist eine Zusammensetzung, die Objekte vom Typ SubjectAlternativeName (IList vom Typ <SubjectAlternativeName>) enthält. Ein neuer SAN kann mit der Aktion Create(SubjectAlternativeNameType, Zeichenkette) erstellt werden. Die übergebenen Parameter sind der Type (vom Typ SubjectAlternativeNameType) und der Value (vom Typ String).
- SubjectAlternativeNameType
 - None (nicht unterstützt)
 - Dns
 - Email
 - IP
 - Uri

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

```
var san1 = certTemplate.SubjectAlternativeNames.Create(SubjectAlternativeNameType.Dns,
"127.0.0.1");
var san2 = certTemplate.SubjectAlternativeNames.Create(SubjectAlternativeNameType.IP,
"192.168.178.1");
var san3 = certTemplate.SubjectAlternativeNames.Create(SubjectAlternativeNameType.Email,
"test@prov.com");
var san4 = certTemplate.SubjectAlternativeNames.Create(SubjectAlternativeNameType.Uri,
"something");
san3.Delete();
```

- SubjectCommonName: string

```
string subjectCommonName = certTemplate.SubjectCommonName;
certTemplate.SubjectCommonName = "exampleSubjectCommonName";
```

Verwendung des Typs CertificateUsage

```
var usage = certTemplate.Usage;
certTemplate.Usage = CertificateUsage.OpcUaClientServer;
```

ValidFrom: Typ DateTime

ValidUntil: Typ DateTime

```
var validFrom = certTemplate.ValidFrom;
var validUntilDateTime = new DateTime(2080, 10, 10);
certTemplate.ValidUntil = validUntilDateTime;
```

7.19.22 Bausteine

7.19.22.1 Gruppe "Programmbausteine" abfragen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Eine PLC ist im Projekt ermittelt.

Programmcode

Ändern Sie den folgenden Programmcode, um die Gruppe "Programmbausteine" abzufragen:

```
private static void GetBlockGroupOfPLC(PlcSoftware plcsoftware)
//Retrieves the system group of a block
{
    PlcBlockSystemGroup blockGroup = plcsoftware.BlockGroup;
}
```

7.19.22.2 Systemgruppe für Systembausteine abfragen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode:

Ändern Sie den folgenden Programmcode, um die für Systembausteine vom System erzeugte Gruppe zu ermitteln:

```
PlcSoftware plcSoftware = ...
foreach (PlcSystemBlockGroup systemGroup in plcSoftware.BlockGroup.SystemBlockGroups)
{
    foreach (PlcSystemBlockGroup group in systemGroup.Groups)
    {
        PlcBlockComposition pbComposition = group.Blocks;
        foreach (PlcBlock block in pbComposition)
        {
            //Fügen Sie Ihren Code hier hinzu
        }
    }
}
```

7.19.22.3 Systemuntergruppen enumerieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode: Alle Systemuntergruppen enumerieren

Ändern Sie den folgenden Programmcode, um die Systemuntergruppen aller Systembausteine zu enumerieren:

```
//Retrieves the system generated group for system blocks
private static void GetSystemgroupForSystemblocks(PlcSoftware plcSoftware)
{
    PlcSystemBlockGroupComposition systemBlockGroups =
plcSoftware.BlockGroup.SystemBlockGroups;
    if (systemBlockGroups.Count != 0)
    {
        PlcSystemBlockGroup sbSystemGroup = systemBlockGroups[0];
        foreach (PlcSystemBlockGroup group in sbSystemGroup.Groups)
        {
            EnumerateSystemBlockGroups(group);
        }
    }
}
private static void EnumerateSystemBlockGroups(PlcSystemBlockGroup systemBlockGroup)
{
    foreach (PlcSystemBlockGroup group in systemBlockGroup.Groups)
    {
        // recursion EnumerateSystemBlockGroups(group);
    }
}
```

Programmcode: Auf eine bestimmte Untergruppe zugreifen

Ändern Sie den folgenden Programmcode, um auf eine bestimmte Untergruppe zuzugreifen:

```
private static void AccessSbGroup(PlcSystemBlockGroup systemBlockGroup)
{
    PlcSystemBlockGroup group1 = systemBlockGroup.Groups.Find("User group XYZ");
    PlcSystemBlockGroup group2 = group1.Groups.Find("User group ZYX");
}
```

Siehe auch

[Externe Datei hinzufügen \(Seite 530\)](#)

7.19.22.4 Benutzerdefinierte Bausteingroupen enumerieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Eine PLC ist im Projekt ermittelt.

Verwendung

Enthaltene Untergruppen werden beim Enumerieren rekursiv berücksichtigt.

Programmcode: Alle Gruppen enumerieren

Ändern Sie den folgenden Programmcode, um die benutzerdefinierten Bausteingroupen zu enumerieren:

```
//Enumerates all block user groups including sub groups
private static void EnumerateAllBlockGroupsAndSubgroups(PlcSoftware plcsoftware)
{
    foreach (PlcBlockUserGroup blockUserGroup in plcsoftware.BlockGroup.Groups)
    {
        EnumerateBlockUserGroups(blockUserGroup);
    }
}

private static void EnumerateBlockUserGroups(PlcBlockUserGroup blockUserGroup)
{
    foreach (PlcBlockUserGroup subBlockUserGroup in blockUserGroup.Groups)
    {
        EnumerateBlockUserGroups(subBlockUserGroup);
        // recursion
    }
}
```

Programmcode: Auf eine Gruppe zugreifen

Ändern Sie den folgenden Programmcode, um auf eine ausgewählte benutzerdefinierte Bausteingroupe zuzugreifen:

```
//Gives individual access to a specific block user group
private static void AccessBlockusergroup(PlcSoftware plcsoftware)
{
    PlcBlockUserGroupComposition userGroupComposition = plcsoftware.BlockGroup.Groups;
    PlcBlockUserGroup plcBlockUserGroup = userGroupComposition.Find("MyUserfolder");
}
```

7.19.22.5 Alle Bausteine enumerieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Eine PLC ist im Projekt ermittelt.

Verwendung

Der gezielte Zugriff auf einen Programmbaustein ist möglich, wenn dessen Name bekannt ist.

Programmcode: Alle Bausteine enumerieren

Ändern Sie den folgenden Programmcode, um die Bausteine aller Bausteingruppen zu enumerieren:

```
private static void EnumerateAllBlocks(PlcSoftware plcsoftware)
//Enumerates all blocks
{
    foreach (PlcBlock block in plcsoftware.BlockGroup.Blocks)
    {
        // Do something...
    }
}
```

Programmcode: Auf einen bestimmten Baustein zugreifen

Um auf einen bestimmten Baustein zuzugreifen, ändern Sie folgenden Programmcode:

```
private static void AccessASingleBlock(PlcSoftware plcsoftware)
//Gives individual access to a block
{
    // The parameter specifies the name of the block
    PlcBlock block = plcsoftware.BlockGroup.Blocks.Find("MyBlock");
}
```

7.19.22.6 Informationen eines Bausteins/Anwenderdatentyps abfragen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Die TIA Portal Openness API unterstützt die Abfrage der folgenden Informationen für Programm und Datenbausteine und für Anwenderdatentypen:

- Zeitstempel im UTC-Zeitformat
Über den Zeitstempel erfahren Sie Folgendes:
 - Wann der Baustein zuletzt übersetzt wurde
 - Wann der Baustein zuletzt geändert wurde
- Attribut „Consistency“
Das Attribut „Consistency“ ist in folgenden Fällen auf „True“ gesetzt:
 - Der Baustein wurde erfolgreich übersetzt.
 - Der Baustein wurde seit der Übersetzung nicht geändert.
 - An externen Objekten wurden keine Änderungen vorgenommen, die eine Neuübersetzung erforderlich machen würden.
- Verwendete Programmiersprache (nur Programm und Datenbausteine)
- Bausteinnummer
- Bausteinname
- Baustinautor
- Bausteinfamilie
- Bausteintitel
- Bausteinversion

Weitere Informationen hierzu siehe AUTOHOTSPOT.

Programmcode

Um die oben aufgeführten Informationen abzufragen, ändern Sie folgenden Programmcode:

```
private static void GetPlcBlockInformation(PlcSoftware plcSoftware)
{
    PlcBlock plcBlock = plcSoftware.BlockGroup.Blocks.Find("MyBlock");
    // Read information
    DateTime compileDate = plcBlock.CompileDate;
    DateTime modifiedDate = plcBlock.ModifiedDate;
    bool isConsistent = plcBlock.IsConsistent;
    int blockNumber = plcBlock.Number;
    string blockName = plcBlock.Name;
    ProgrammingLanguage programmingLanguage = plcBlock.ProgrammingLanguage;
    string blockAuthor = plcBlock.HeaderAuthor;
    string blockFamily = plcBlock.HeaderFamily;
    string blockTitle = plcBlock.HeaderName;
    System.Version blockVersion = plcBlock.HeaderVersion;
}
```

Siehe auch

[Import von Projektierungsdaten \(Seite 780\)](#)

7.19.22.7 Einen Baustein schützen und Schutz aufheben

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Die PLC ist nicht online.

Verwendung

Über die Klasse PlcBlockProtectionProvider und den Dienst PlcBlockProtectionProvider können Sie einen Baustein mit einem Passwort schützen und den Passwortschutz wieder aufheben. Der Dienst PlcBlockProtectionProvider ist bei Bausteinen verfügbar, die folgende Voraussetzungen erfüllen:

- Der Baustein kann einen Know-how-Schutz erhalten
- Der Baustein ist ein Code-Baustein oder eine globale DB
- Der Baustein ist im aktuellen PLC unterstützt oder editierbar
- Der Baustein befindet sich nicht in einem schreibgeschützten Kontext
- Der Baustein ist nicht Know-how-geschützt
- Der Baustein ist nicht online

- Der Baustein ist keine CPU-DB
- Der Baustein nutzt keine klassische Verschlüsselungssprache, ProDiag oder ProDiag-OB
- Der Baustein ist kein verschlüsselter importierter klassischer Baustein

Falls der Baustein nicht alle Bedingungen erfüllt, wird von der Methode GetService() eine Nullreferenz ausgegeben.

Programmcode: Vorgänge in Bezug auf Know-how-Schutz ausführen

Ändern Sie folgenden Programmcode:

```
PlcBlock block = ...;
PlcBlockProtectionProvider protectionProvider =
block.GetService<PlcBlockProtectionProvider>();
if (protectionProvider != null)
{
    ... // perform know-how protection related operations here
}
```

Einen Baustein schützen

Legen Sie mit der Methode Protect() ein Passwort fest, um den Programmierbaustein zu schützen.

```
void Protect(SecureString password)
```

Ein Fehler tritt auf,

- wenn Sie einen bereits geschützten Baustein zu schützen versuchen: Eine EngineeringTargetInvocationException wird mit einer Meldung ausgelöst, die besagt, dass Sie kein bereits geschütztes Objekt schützen können.
- wenn Sie einen Baustein mit einem leeren String als Passwort zu schützen versuchen: Eine EngineeringTargetInvocationException wird mit einer Meldung ausgelöst, die besagt, dass kein Passwort angegeben wurde.
- wenn Sie einen fehlersicheren Baustein zu schützen versuchen, während das Failsafe-Programm passwortgeschützt ist: Eine EngineeringTargetInvocationException wird ausgelöst.
- wenn Sie einen fehlersicheren Baustein zu schützen versuchen, während dieser Baustein nicht aufgerufen ist: Eine EngineeringTargetInvocationException wird ausgelöst.

Ungeschützter Baustein

Entfernen Sie das Passwort, mit dem der Programmierbaustein geschützt ist, mithilfe der Methode Unprotect().

```
void Unprotect(SecureString password)
```

Ein Fehler tritt auf,

- wenn Sie den Schutz eines bereits ungeschützten Bausteins aufzuheben versuchen: Eine EngineeringTargetInvocationException wird mit einer Meldung ausgelöst, die besagt, dass Sie nicht den Schutz eines ungeschützten Objekts aufheben können.
- wenn Sie den Schutz eines Bausteins mit dem falschen Passwort aufzuheben versuchen: Eine EngineeringTargetInvocationException wird mit einer Meldung ausgelöst, die besagt, dass das Passwort nicht akzeptiert wurde.
- wenn Sie einen Baustein mit einem leeren String als Passwort zu schützen versuchen: Eine EngineeringTargetInvocationException wird mit einer Meldung ausgelöst, die besagt, dass kein Passwort angegeben wurde.

Nach ungültigen Zeichen suchen

Da Sie einen Baustein unter Verwendung der Protect()-Methode mit beliebigen Zeichen schützen können, einschließlich Rückschritt, Tab usw., ist es unter Umständen nicht möglich, den Schutz im TIA Portal aufzuheben. Da Passwörter als SecureString übermittelt werden, müssen Sie selbst überprüfen, ob das Passwort ungültige Zeichen enthält. Mit der GetInvalidPasswordCharacters()-Methode können Sie eine Liste ungültiger Zeichen abrufen.

```
SecureString CreatePasswordString(ProtectionProvider protectionProvider, IEnumerable<char>
contentCharacters)
{
    IList<char> invalidCharacters = protectionProvider.GetInvalidPasswordCharacters();
    SecureString password = new SecureString();
    foreach(char ch in contentCharacters)
    {
        if (!invalidCharacters.Contains(ch))
        {
            password.AppendChar(ch);
        }
        else
        {
            // at least one of the content characters is not valid
            // signal an error - e.g. throw an exception
            ...
        }
    }
    return password;
}
```

Ein Fehler tritt auf,

- wenn Sie den Schutz eines bereits ungeschützten Bausteins aufzuheben versuchen: Eine EngineeringTargetInvocationException wird mit einer Meldung ausgelöst, die besagt, dass Sie nicht den Schutz eines ungeschützten Objekts aufheben können.
- wenn Sie den Schutz eines Bausteins mit dem falschen Passwort aufzuheben versuchen: Eine EngineeringTargetInvocationException wird mit einer Meldung ausgelöst, die besagt, dass das Passwort nicht akzeptiert wurde.
- wenn Sie einen Baustein mit einem leeren String als Passwort zu schützen versuchen: Eine EngineeringTargetInvocationException wird mit einer Meldung ausgelöst, die besagt, dass kein Passwort angegeben wurde.

7.19.22.8 Baustein löschen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Die PLC ist nicht online.

Programmcode

Ändern Sie den folgenden Programmcode, um einen Baustein zu löschen:

```
//Runs through block group and deletes blocks
private static void DeleteBlocks(PlcSoftware plcsoftware)
{
    PlcBlockSystemGroup group = plcsoftware.BlockGroup;
    // or BlockUserGroup group = ...;
    for (int i = group.Blocks.Count - 1; i >= 0; i--)
    {
        PlcBlock block = group.Blocks[i];
        if (block != null)
        {
            block.Delete();
        }
    }
}
```

Siehe auch

Import von Projektierungsdaten (Seite 780)

7.19.22.9 Gruppe für Bausteine erzeugen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode

Ändern Sie folgenden Programmcode, um eine Gruppe für Bausteine zu erzeugen:

```
private static void CreateBlockGroup(PlcSoftware plcsoftware)
//Creates a block group
{
    PlcBlockSystemGroup systemGroup = plcsoftware.BlockGroup;
    PlcBlockUserGroupComposition groupComposition = systemGroup.Groups;
    PlcBlockUserGroup myCreatedGroup = groupComposition.Create("MySubGroupName");
}
```

Siehe auch

[Import von Projektierungsdaten \(Seite 780\)](#)

7.19.22.10 Gruppe für Bausteine löschen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
[Siehe Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
[Siehe Projekt öffnen \(Seite 111\)](#)
- Die PLC ist nicht online.

Programmcode

Um eine Gruppe für Bausteine zu löschen, ändern Sie folgenden Programmcode:

```
// Deletes user groups from PlcBlockSystemGroup or PlcBlockUserGroup
private static void DeleteBlockFolder(PlcSoftware plcSoftware)
{
    PlcBlockUserGroup group = plcSoftware.BlockGroup.Groups.Find("myGroup");
    //PlcBlockSystemGroup group = plcSoftware.BlockGroup;
    PlcBlockUserGroupComposition subgroups = group.Groups;
    PlcBlockUserGroup subgroup = subgroups.Find("myUserGroup");
    if (subgroup != null)
    {
        subgroup.Delete();
    }
}
```

Siehe auch

[Import von Projektierungsdaten \(Seite 780\)](#)

7.19.22.11 Auf Attribute sämtlicher Bausteine zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Sie können die für alle Bausteine geltenden Attribute mit der Methode SetAttribute() und SetAttributes() festlegen.

SetAttributes() kann für alle Objekttypen verwendet werden, bei denen ein schreibbares Openness-Attribut verfügbar ist. Bei einem Eingabeelement verhält es sich wie SetAttribute.

Parameterprüfung:

- Anfängliche Prüfungen werden durchgeführt, bevor das erste Element von der globalen PE-Infrastruktur verarbeitet wird: Die Eingabeelemente werden auf identische und gültige Attributnamen sowie auf Übereinstimmung der Datentypen geprüft. Ein Attribut kann von einem SetAttributes nicht zweimal festgelegt werden.
- Eine sequenzielle Prüfung erfolgt während der Ausführung von SetAttributes. Die Reihenfolge der Verarbeitung entspricht der Reihenfolge der Elemente in der Eingabeliste.

SetAttributes wird in nur einer Transaktion verarbeitet. Wenn ein Attribut der Eingabeliste nicht festgelegt werden kann, bleiben die Werte der Attribute unverändert. SetAttributes legt die Attribute in der gleichen Reihenfolge fest, wie sie in der Parameterliste vorhanden sind. Die Parameter können voneinander abhängig oder unabhängig sein. Wenn ein Parameter von einem vorherigen Parameter in der Liste abhängig ist, wird dieser Parameter geprüft und entsprechend dem aktuellen Wert des anderen Attributs ausgewertet (nicht entsprechend dem ursprünglichen Wert zum Zeitpunkt des Aufrufs). Wenn SetAttributes nicht ausgeführt werden kann, werden Einzelheiten zum Fehler angegeben. Die Fehlermeldung enthält den Namen des ersten Attributs, das nicht festgelegt werden konnte, und dessen Wert sowie den Grund dafür.

Die folgenden Programmcodebeispiele basieren auf den zwei Attributen AutoNumber und Number, die SetAttribute() verwenden (siehe Bausteine exportieren (Seite 898) für alle geltenden Attribute von Bausteinen).

Programmcode: SetAttribute

```
...
PlcBlockGroup blockFolder = YourUtilities.GetFolder();
var block = blockFolder.Blocks.Find("Block_1");
if ((bool)block.GetAttribute("AutoNumber")==true)
{
    block.SetAttribute("AutoNumber",false);
}
block.SetAttribute("Number",2);
...
```

Programmcode: SetAttributes

```
PlcBlock block = SelectBlock("MC-Servo");
if (block != null)
{
    IList<KeyValuePair<string, object>> list = new List<KeyValuePair<string, object>>()
    {
        new KeyValuePair<string, object>("DataExchangeMode", OBDDataExchangeMode.Synchronous), new
        KeyValuePair<string, object>("SynchronousApplicationCycleTime", (float)69)
    };
    try
    {
        block.SetAttributes(list);
    }
    catch (EngineeringException e)
    {
        Console.WriteLine("Exception: " + e.Message);
    }
}
```

7.19.22.12 Einen ProDiag-FB anlegen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Der Openness-Anwender kann mit der Create-Aktion der PLCBlock-Zusammensetzung mit den folgenden Parametern einen ProDiag-FB anlegen.

1. Name
2. Auto-Nummerierungsmerker
3. Nummer
 - Wenn "Auto-Nummerierungsmerker" wahr ist, wird die angegebene Bausteinnummer festgelegt, sofern sie frei ist. Andernfalls wird ein neuer Baustein generiert.
 - Wenn "Auto-Nummerierungsmerker" falsch ist, wird für die angegebene Bausteinnummer der Baustein festgelegt.
4. Programmiersprache
 - Wenn der Anwender die Create-Aktion mit der Programmiersprache ProDiag aufruft, wird ein neuer FB ohne IDB angelegt.
 - Wenn der Anwender die Create-Aktion mit IDB von ProDiag aufruft, dann wird IDB von ProDiag angelegt.
 - In jedem anderen nicht unterstützten Fall wird eine wiederherstellbare Ausnahme ausgelöst.

Programmcode: Einen ProDiag-FB anlegen

```
...
PlcBlockGroup blockFolder = plc.BlockGroup;
PlcBlockComposition blockComposition = blockFolder.Blocks;
if (blockComposition != null)
{
  string fbName = "ProDiag_Block";
  bool isAutoNumber = true;
  int number = 1;
  var progLang = ProgrammingLanguage.ProDiag;
  FB block = blockComposition.CreateFB(fbName, isAutoNumber, number, progLang);
  string iDBName="ProDiag_IDB";
  string instanceOfName = fbName;
  InstanceDB iDbBlock = blockComposition.CreateInstanceDB(iDBName, isAutoNumber, number,
  instanceOfName);
}
...
}
```

Siehe auch

[Auf Überwachungen und Eigenschaften des ProDiag-FB zugreifen \(Seite 528\)](#)

7.19.22.13 Auf Überwachungen und Eigenschaften des ProDiag-FB zugreifen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Auf Überwachungen des User-FB zugreifen

Der Openess-Anwender kann mit dem folgenden Code-Snippet auf die Überwachungen des FB zugreifen. Jeder FB hat die Liste der Überwachungen einschließlich Classic und Plus PLCs.

Programmcode: Auf Überwachungen des ProDiag-FB zugreifen

```
...
PlcBlock iDB = plc.BlockGroup.Blocks.Find("FB_Block_DB");
string fbName = iDB.GetAttribute("InstanceOfName").ToString();
FB fb = (FB)plc.BlockGroup.Blocks.Find(fbName);
if (fb.Supervisions.Count > 0) Console.WriteLine("Contains supervisions");
else
Console.WriteLine("Does not contains supervisions");
...
...
```

Auf die Attribute des FB-Bausteins zugreifen

Der Openess-Anwender kann AssignedProDiagFB an InstanceDB über das Attribut AssignedProDiagFB festlegen (siehe Bausteine exportieren (Seite 898)). Der Anwender kann mit der Methode GetAttribute(), GetAttributes() und SetAttribute() auf die Attribute zugreifen. Der Anwender kann die Methode SetAttributes() nicht dazu verwenden, die Attribute für mehr als ein Attribut festzulegen. TIA Portal Openess löst eine Ausnahme bei Verwenden der Methode SetAttributes() aus.

Wenn das Attribut (im angegebenen Baustein) nicht unterstützt wird, wird eine wiederherstellbare Benutzerausnahme ausgelöst. Ist kein zugeordneter ProDiag-Baustein festgelegt, gibt GetAttribute() eine leere Zeichenkette zurück.

Programmcode: Zugeordneten ProDiag-FB und IDB abrufen und festlegen

```
...
PlcBlockGroup blockFolder = plc.BlockGroup;
PlcBlock instanceDB = blockFolder.Blocks.Find("IDB");
PlcBlock plcProdiag = blockFolder.Blocks.Find("block_Prodiag");
instanceDB.SetAttribute("AssignedProDiagFB", plcProdiag.Name);
var assignedProDiagFB = instanceDB.GetAttribute("AssignedProDiagFB");
...

```

Siehe auch

[Einen ProDiag-FB anlegen \(Seite 526\)](#)

7.19.22.14 ProDiag-FB-Bausteine und -Attribute lesen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Sie haben über eine TIA Portal Openness-Anwendung ein Projekt geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Sie können TIA Portal Openness verwenden, um die Version des ProDiag-Funktionsbausteins und weitere Attributwerte in Verbindung mit ProDiag zu lesen. Mit GetAttribute () und GetAttributes () können Sie die vorhandenen sprachspezifischen Attribute von ProDiag-FBs lesen.

Attribute

Die folgenden Attribute werden von ProDiag-FBs in Openness unterstützt:

Attribute	Typ
ProDiagVersion	Version
InitialValueAcquisition	Bool
UseCentralTimeStamp	Bool

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.19.22.15 Externe Datei hinzufügen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Sie haben über eine TIA Portal Openess-Anwendung ein Projekt geöffnet:
Siehe Projekt öffnen (Seite 111)

Verwendung

Sie können einem PLC eine externe Datei hinzufügen. Diese externe Datei wird im Dateisystem unter dem definierten Pfad gespeichert.

Folgende Formate werden unterstützt:

- AWL
- SCL
- DB
- UDT

Hinweis

Der Zugriff auf Gruppen im Ordner „Externe Quelldateien“ wird nicht unterstützt.

Es wird eine Ausnahme ausgelöst, wenn Sie eine andere Dateierweiterung als *.AWL, *.SCL, *.DB oder *.UDT angeben.

Programmcode

Ändern Sie den folgenden Programmcode, um im Ordner "Externe Quelldateien" eine externe Datei aus einem Baustein zu erzeugen:

```
private static void CreateBlockFromFile(PlcSoftware plcSoftware)
// Creates a block from a AWL, SCL, DB or UDT file
{
    PlcExternalSource externalSource =
plcSoftware.ExternalSourceGroup.ExternalSources.CreateFromFile("SomeBlockNameHere", "SomePa
thHere");
}
```

7.19.22.16 Quelle aus Baustein generieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Die PLC ist nicht online.

Verwendung

Die Schnittstelle TIA Portal Openness API unterstützt die Generierung von Quellen in UTF-8 aus AWL- oder SCL-Bausteinen, Datenbausteinen und PLC-Typen (Anwenderdatentypen). Um eine Quelldatei eines Bausteins zu generieren, rufen Sie die Methode `GenerateSource` auf der Instanz `PlcExternalSourceSystemGroup` auf.

Der Umfang der generierten Quelldatei ist von der Generierungsoption dieser Funktion abhängig:

- `GenerateOptions.None`
Quelle nur aus bereitgestellten Bausteinen generieren.
- `GenerateOptions.WithDependencies`
Quelle einschließlich aller abhängigen Objekte generieren.

Die Schnittstelle `Siemens.Engineering.SW.ExternalSources.IGenerateSource` zeigt an, dass eine Quelle generiert werden kann.

Für Bausteine werden nur die Programmiersprachen AWL und SCL unterstützt. In den folgenden Fällen werden Ausnahmen ausgelöst:

- Die Programmiersprache ist nicht AWL oder SCL
- Eine Datei gleichen Namens ist bereits am Zielspeicherort vorhanden.

Für Anwenderdatentypen wird nur die Dateierweiterung `".udt"` unterstützt. In den folgenden Fällen werden Ausnahmen ausgelöst:

- Die Dateierweiterung für DBs ist nicht `".db"`.
- Die Dateierweiterung für AWL-Bausteine ist nicht `".awl"`.
- Die Dateierweiterung für SCL-Bausteine ist nicht `".scl"`.

Programmcode

Um Quelldateien aus Bausteinen und Typen zu erzeugen, ändern Sie folgenden Programmcode:

```
//method declaration
...
PlcExternalSourceSystemGroup.GenerateSource

(IEnumerable<Siemens.Engineering.SW.ExternalSources.IGenerateSource>
plcBlocks, FileInfo sourceFile, GenerateOptions generateOptions);
...
//examples
...
var blocks = new List<PlcBlock>(){block1};
var fileInfo = new FileInfo(@"C:\temp\SomePathHere.scl");

PlcExternalSourceSystemGroup systemGroup = ...;

systemGroup.GenerateSource(blocks, fileInfo, GenerateOptions.WithDependencies);

// exports all blocks and with all their dependencies(e.g. called blocks, used DBs or UDTs)
// as ASCII text into the provided source file.
...
or
...
var types = new List<PlcType>(){udt1};
var fileInfo = new FileInfo(@"C:\temp\SomePathHere.udt");

PlcExternalSourceSystemGroup systemGroup = ...;

systemGroup.GenerateSource(types, fileInfo, GenerateOptions.WithDependencies );

// exports all data types and their used data types into the provided source file.
...
```

Siehe auch

[Import von Projektierungsdaten \(Seite 780\)](#)

7.19.22.17 Bausteine aus Quelle erzeugen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Die PLC ist nicht online.

Verwendung

In der Gruppe "Externe Quelldateien" können Sie Bausteine aus allen externen Dateien erzeugen. Es werden nur externe Dateien mit dem Format ASCII unterstützt.

Hinweis

Der Zugriff auf Gruppen im Ordner „Externe Quelldateien“ wird nicht unterstützt.

Vorhandene Bausteine werden überschrieben.

Wenn beim Aufruf ein Fehler auftritt, wird eine Exception ausgelöst. Die ersten 256 Zeichen jeder Fehlermeldung sind in der Benachrichtigung der Exception enthalten. Das Projekt wird auf den Verarbeitungszustand vor der Ausführung der Methode `GenerateBlocksFromSource` zurückgesetzt.

Programmcode

Um in der Gruppe "Externe Quelldateien" aus allen externen Dateien Bausteine zu erzeugen, ändern Sie folgenden Programmcode:

```
// Creates a block from an external source file
PlcSoftware plcSoftware = ...;
foreach (PlcExternalSource plcExternalSource in
plcSoftware.ExternalSourceGroup.ExternalSources)
{
    plcExternalSource.GenerateBlocksFromSource();
}
```

7.19.22.18 Generierung aus einer Quelle mit bekanntem Quellformat

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Der PLC ist nicht online.

Anwendung

Der Generator kann grundsätzlich ein Objekt (Baustein/UDT) erzeugen, doch ein internes Teil verursacht, dass der erzeugte Baustein/UDT nicht übersetzbare ist.

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Typische fehlende oder ungültige Teile, die verursachen, dass der Baustein/Typ nicht übersetzbare ist:

- Verweis auf einen nicht vorhandenen Typ
- Verwendung eines fehlenden Symbols

Bei Generierungsfehlern in der aktuellen Implementierung von 'GenerateBlocksFromSource' void GenerateBlocksFromSource():

- Wird eine wiederherstellbare Ausnahme zurückgegeben (aufgrund einer fehlgeschlagenen Generierung)
- Werden die Fehlermeldung(en) der Generierung in die Meldung der Ausnahme eingeschlossen
- Wird der generierte Baustein/UDT gelöscht

Aufgrund der Langzeitstabilität von Openness ist eine neue Überladung von GenerateBlocksFromSource verfügbar:

IList<IEngineeringObject> GenerateBlocksFromSource(GenerateBlockOptions option)
Wenn dieses neue 'GenerateBlocksFromSource' mit GenerateBlockOptions.KeepOnError aufgerufen wird,

- Wird keine wiederherstellbare Ausnahme zurückgegeben (unabhängig von den Generierungsergebnissen)
- Werden keine Informationen über die Generierungsergebnisse der erzeugten Bausteine geliefert
- Wird eine IList der IEngineeringObjects der generierten Bausteine zurückgegeben (mit oder ohne Generierungsfehler)

Wenn dieses neue 'GenerateBlocksFromSource' mit GenerateBlockOptions.None aufgerufen wurde und die Generierung nicht fehlerfrei war,

- Wird eine wiederherstellbare Ausnahme zurückgegeben (aufgrund einer fehlgeschlagenen Generierung)
- Werden die Fehlermeldung(en) der Generierung in die Meldung der Ausnahme eingeschlossen
- Wird der generierte Baustein/UDT gelöscht
 - Dies ist das gleiche Verhalten wie bei der aktuellen Implementierung.

Bei fehlerfreier Generierung

- Wird keine Ausnahme zurückgegeben
- Werden keine Informationen über die Generierungsergebnisse der erzeugten Bausteine geliefert
- Wird eine IList der IEngineeringObjects der generierten Bausteine zurückgegeben
 - Dies ist das gleiche Verhalten wie beim Aufruf mit GenerateBlockOptions.KeepOnError.

Programmcode

```

try
{
    IList<IEngineeringObject> generatedObjects =
externalSource.GenerateBlocksFromSource(GenerateBlockOptions.KeepOnError);
    foreach (IEngineeringObject engineeringObject in generatedObjects)
{
    CompilerResult compilerResult = null;
    string objectName = null;
    if (engineeringObject is PlcBlock)
    {
        // handle case for PlcBlock
        // e.g. retrieve the compiler result
        PlcBlock block = (PlcBlock)engineeringObject;
        objectName = block.Name;
        compilerResult = block.Compile();
    }
    else if (engineeringObject is PlcType)
    {
        // handle case for PlcType
        // e.g. retrieve the compiler result
        PlcType plcType = (PlcType)engineeringObject;
        objectName = plcType.Name;
        compilerResult = plcType.Compile();
    }
    // handle the compiler result
    if (compilerResult != null)
    {
        if (compilerResult.State == CompilerResultState.Error)
        {
            Console.WriteLine("Object '{0}' could not be compiled successfully!", objectName);
            Console.WriteLine("Number of compiler errors: {0}", compilerResult.ErrorCount);
            foreach (CompilerResultMessage compilerResultMessage in compilerResult.Messages)
            {
                Console.WriteLine(compilerResultMessage.Description);
            }
        }
        else
        {
            Console.WriteLine("Object '{0}' could be compiled successfully.", objectName);
        }
    }
}
catch (RecoverableException exception)
{
    // handle recoverable exception
    Console.WriteLine(exception.Message);
}

```

Siehe auch

Verbindung zum TIA Portal aufbauen (Seite 79)

Projekt öffnen (Seite 111)

7.19.22.19 Anwenderdatentyp löschen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Die PLC ist nicht online.

Programmcode

Ändern Sie den folgenden Programmcode, um einen Benutzertyp zu löschen:

```
private static void DeleteUserDataType(PlcSoftware plcSoftware)
{
    PlcTypeSystemGroup typeGroup = plcSoftware.TypeGroup;
    PlcTypeComposition dataTypes = typeGroup.Types;
    PlcType dataType = dataTypes.Find("DataTypeName");
    if (dataType != null)
    {
        dataType.Delete();
    }
}
```

Siehe auch

Import von Projektierungsdaten (Seite 780)

7.19.22.20 Externe Datei löschen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Sie haben über eine TIA Portal Openess-Anwendung ein Projekt geöffnet:
Siehe Projekt öffnen (Seite 111)
- Die PLC ist nicht online.

Programmcode

Um in der Gruppe "Externe Quelldateien" eine externe Datei zu löschen, ändern Sie folgenden Programmcode:

Hinweis

Der Zugriff auf Gruppen in der Gruppe "Externe Quelldateien" wird nicht unterstützt.

```
// Deletes an external source file
private static void DeleteExternalSource(PlcSoftware plcSoftware)
{
    PlcExternalSource externalSource =
plcSoftware.ExternalSourceGroup.ExternalSources.Find("myExternalsource");
    externalSource.Delete();
}
```

7.19.22.21 Bausteineditor starten

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Eine Instanz des TIA Portals ist mit Benutzeroberfläche geöffnet.

Programmcode

Um den zugehörigen Editor für eine Objektreferenz des Typs `PlcBlock` in der Instanz des TIA Portals zu starten, ändern Sie den folgenden Programmcode:

```
//Opens a block in a block editor
private static void StartBlockEditor(PlcSoftware plcSoftware)
{
    PlcBlock plcBlock = plcSoftware.BlockGroup.Blocks.Find("MyBlock");
    plcBlock.ShowInEditor();
}
```

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Ändern Sie folgenden Programmcode, um den zugehörigen Editor für eine Objektreferenz des Typs `PlcType` in der Instanz des TIA Portals zu öffnen:

```
//Opens a udt in udt editor
private static void StartPlcTypEditor(PlcSoftware plcSoftware)
{
    PlcTypeComposition types = plcSoftware.TypeGroup.Types;
    PlcType udt = types.Find("my_udt");
    udt.ShowInEditor();
}
```

Siehe auch

[Import von Projektierungsdaten \(Seite 780\)](#)

7.19.22.22 Bausteine mit Hilfe von Fingerabdrücken ändern

Voraussetzungen

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Sie haben mit Ihrer Anwendung TIA Portal Openness ein Projekt geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Über TIA Portal Openness können Sie Änderungen in Bausteinen oder UDTs erkennen. Hierfür können Sie die Fingerabdrücke des Objekts miteinander vergleichen. Eine Fingerabdruck-Instanz enthält eine `FingerprintId`, die die Art des Fingerabdrucks festlegt, sowie den `fingerprint`-Wert als String. Alle bereitgestellten `fingerprints` berücksichtigen nur Benutzereingaben, kein Übersetzungsergebnis oder andere Änderungen, die das System vornimmt.

Die Enumeration `FingerprintId` listet alle Arten von Fingerabdrücken auf, die in Openness unterstützt werden:

Wert	Beschreibung
Code	Berücksichtigt alle Änderungen am Code im Hauptteil des Bausteins. Das Übersetzungsergebnis wird nicht berücksichtigt.
Interface	Berücksichtigt alle Änderungen an der Schnittstelle eines Bausteins, einschließlich der Startwerte eines DB.
Properties	Berücksichtigt Änderungen an den Eigenschaften eines Bausteins, z. B. Name, Nummer.

Wert	Beschreibung
Comments	Berücksichtigt Änderungen an den Kommentaren eines Bausteins. Bei OBs ändert sich der Fingerabdruck auch, wenn die Liste verfügbarer Sprachen in den Spracheinstellungen des Projekts geändert wird.
LibraryType	Vorhanden, wenn ein Baustein mit einem Bibliothekstyp verschaltet ist.
Texts	Bei V15 SP1 ist dieser Fingerabdruck nur für Graph-Bausteine vorhanden.
Alarms	Vorhanden, wenn ein Baustein mit der Meldungsfunktion arbeitet.
Supervision	Vorhanden, wenn ein Baustein eine Überwachungsfunktion enthält.
TechnologyObject	Nur für Technologieobjekt-DBs vorhanden.
Events	Nur für OBs vorhanden.
TextualInterface	Vorhanden, wenn der Baustein eine Textschnittstelle besitzt.

Sie benötigen den FingerprintProvider , um die Fingerabdrücke eines Objekts aufzurufen. Dieser Dienst ist für Bausteine (FB, FC, OB, DB) und UDTs, jedoch nicht für VariablenTabellen verfügbar. Der FingerprintProvider berechnet alle verfügbaren Fingerabdrücke eines Objekts und gibt diese bei jedem Aufruf von GetFingerprints zurück. Um korrekte Fingerabdrücke zu erhalten, muss der Baustein oder UDT vor dem Aufrufen der Fingerabdrücke konsistent sein. Sonst wird eine RecoverableException gemeldet. Ist ein Fingerabdruck nach seiner Berechnung ungültig, wird eine RecoverableException gemeldet.

Programmcode

Um die Fingerabdruckinstanz abzurufen, ändern Sie folgenden Programmcode:

```
PlcBlock block = ...;
FingerprintProvider provider = block.GetService<FingerprintProvider>();
IList<Fingerprint> fingerprints = provider.GetFingerprints();
foreach(var fingerprint in fingerprints)
{
    string fpValue = fingerprint.Value;
    FingerprintId fpId = fingerprint.Id;
}
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.19.22.23 Bausteine für benutzerdefinierte Seiten generieren/löschen

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Sie können mit TIA Portal Openness benutzerdefinierte Seiten innerhalb des Webservers von PLCs erstellen: Die Daten für diese benutzerdefinierten Seiten werden in speziellen, systemgenerierten Bausteinen der PLC gespeichert. Sie können diese Bausteine auch manuell bearbeiten und löschen. Wenn Sie jedoch Daten oder Eigenschaften in diesen Datenbausteinen ändern, funktioniert der Webserver anschließend nicht mehr ordnungsgemäß.

In TIA Portal Openness wird der neue Dienst WebserverUserDefinedPages an der PLC des DeviceItem hinzugefügt. Sie können mithilfe dieses Dienstes die Datenbausteine für benutzerdefinierte Seiten generieren: `List<PLCBlock> GenerateBlocks(Arguments)`. Beim Aufruf dieser Funktion werden die entsprechenden Datenbausteine generiert und eine Liste der generierten Datenbausteine wird an den Benutzer zurückgegeben.

Es gibt zwei mögliche Überladungen dieser Funktion:

- `List<PLCBlock> WebserverUserDefinedPages.GenerateBlocks(WebDBGenerateOptions GenerateOptions);`
- `List<PLCBlock> WebserverUserDefinedPages.GenerateBlocks(System.IO.DirectoryInfo htmlDirectory, System.IO.FileInfo defaultHTMLPage, string applicationName, WebDBGenerateOptionsGenerateOptions);`

Die Parameterwerte für das HTML-Verzeichnis, die HTML-Seite und den Anwendungsnamen werden nicht in den Projektdaten gespeichert. Somit werden die Werte der entsprechenden dynamischen Attribute nicht geändert.

Parameter	Datentyp	Typ	Beschreibung
GenerateOptions	WebDBGenerateOptions (enumeration)	Mandatory	<p>Mögliche Werte:</p> <ul style="list-style-type: none"> • None - Wenn Datenbausteine für benutzerdefinierte Seiten bereits vorher erstellt wurden, wird keine Aktion ausgeführt und eine Ausnahme ausgelöst • Override - Wenn Datenbausteine für benutzerdefinierte Seiten bereits vorher erstellt wurden, werden sie gelöscht und die neuen Datenbausteine generiert
htmlDirectory	System.IO.DirectoryInfo	Optional	Gibt das HTML-Verzeichnis unabhängig vom Wert des dynamischen Attributs WebserverHTMLDirectory an.
defaultHTMLPage	System.IO.FileInfo	Optional	Gibt die standardmäßige HTML-Seite unabhängig vom Wert des dynamischen Attributs WebserverDefaultHTMLPage an.
applicationName	string	Optional	Gibt den Anwendungsnamen unabhängig vom Wert des dynamischen Attributs WebserverApplicationName an.

GenerateBlocks() löst eine wiederherstellbare Ausnahme aus:

- Wenn der Webserver deaktiviert ist (WebserverActive == False) - "Webserver muss aktiviert werden"
- Wenn WebserverHTMLDirectory leer oder der Pfad ungültig ist - "HTML-Verzeichnis hat ungültigen Pfad"
- Wenn WebserverDefaultHTMLPage leer oder der Pfad ungültig ist - "Default-HTML-Verzeichnis hat ungültigen Pfad"
- Wenn das WebserverHTMLDirectory zu lang ist - "HTML-Verzeichnis hat zu viele Daten"
- Der Anwendungsname ist ungültig - "Der Applikationsname ist ungültig"
- Die dynamischen Inhalte sind ungültig - "Dynamischer Inhalt ist ungültig"
- Die Nummer des Steuerdatenbausteins ist ungültig - "Die Nummer des Steuer-DBs ist ungültig"

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

- Die Startnummer des Datenbausteins ist ungültig - "Die Startnummer des DBs ist ungültig"
- Bausteine sind bereits vorhanden: Wenn der Benutzer versucht, einen Baustein mit einem Namen zu generieren, der bereits vorhanden ist - "Löschen Sie die bestehenden Bausteine, bevor Sie neue Bausteine generieren" (wird nur ausgelöst, wenn WebDBGenerateOptions.None verwendet wurde)

Es gibt keine spezielle Funktion zum Löschen aller Bausteine, die mit benutzerdefinierten Seiten verbunden sind. Sie müssen manuell die Bausteine löschen, die von der Aktion "Generieren" an die Seiten zurück übergeben werden, oder zu den entsprechenden Datenbausteinen navigieren, wie es in TIA Portal Openness bereits möglich ist.

Programmcode

Ändern Sie den folgenden Programmcode, um einen Baustein für eine benutzerdefinierte Seite zu generieren:

```
DeviceItem deviceItem = ...;
var WebserverUserDefinedPagesService = deviceItem.GetService<WebserverUserDefinedPages>();
List<PLCBlock> blocks =
WebserverUserDefinedPagesService.GenerateBlocks(WebDBGenerateOptions.None);
List<PLCBlock> blocks = WebserverUserDefinedPagesService.GenerateBlocks(htmlDirectory,
defaultHTMLPage, applicationName, WebDBGenerateOptions.Override);
```

Siehe auch

- Verbindung zum TIA Portal aufbauen (Seite 79)
- Projekt öffnen (Seite 111)

7.19.23 Technologieobjekte

7.19.23.1 Übersicht über Funktionen für Technologieobjekte

TIA Portal Openness unterstützt eine Auswahl von Technologieobjektfunktionen für definierte Aufgaben, die Sie mittels der Public API außerhalb des TIA Portal aufrufen können.

Sie erhalten die Codekomponenten, die für jede Aufgabe angepasst werden müssen.

Funktionen

Die folgenden Funktionen sind für Technologieobjekte verfügbar:

- Zusammensetzung von Technologieobjekten abfragen (Seite 546)
- Technologieobjekt erstellen (Seite 546)
- Technologieobjekt löschen (Seite 547)
- Technologieobjekt übersetzen (Seite 548)
- Technologieobjekte enumerieren (Seite 549)

- Technologieobjekt finden (Seite 550)
- Parameter eines Technologieobjekts enumerieren (Seite 551)
- Parameter eines Technologieobjekts finden (Seite 551)
- Parameter eines Technologieobjekts lesen (Seite 552)
- Parameter eines Technologieobjekts schreiben (Seite 553)

Siehe auch

Standard-Bibliotheken (Seite 45)

Einsatzmöglichkeiten (Seite 38)

7.19.23.2 Übersicht über Technologieobjekte und Versionen

Technologieobjekte

Die folgende Tabelle zeigt die in der Public API verfügbaren Technologieobjekte.

CPU	FW	Technologieobjekt	Version des Technologieobjekts
S7-1200	≥ V4.2	TO_PositioningAxis	V6.0
		TO_CommandTable	
		PID_Compact	V2.3
		PID_3Step	
		PID_Temp	V1.1

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

CPU	FW	Technologieobjekt	Version des Technologieobjekts
S7-1500	< V2.0	High_Speed_Counter	V3.0
		SSI_Absolute_Encoder	V2.0
	\geq V2.0	TO_SpeedAxis	\geq V3.0
		TO_PositioningAxis	
		TO_ExternalEncoder	
		TO_SynchronousAxis	
		TO_OutputCam	
		TO_CamTrack	
		TO_MeasuringInput	
		TO_Cam (S7-1500T) ¹⁾	
		TO_Kinematics (S7-1500T)	V4.0
		High_Speed_Counter	\geq V3.0
		SSI_Absolute_Encoder	\geq V2.0
		PID_Compact	\geq V2.3
		PID_3Step	V2.3
		PID_Temp	V1.1
		CONT_C	
		CONT_S	
		TCONT_CP	
		TCONT_S	
S7-300/400	Jede	CONT_C	V1.1
		CONT_S	
		TCONT_CP	
		TCONT_S	
		TUN_EC ²⁾	
		TUN_ES ²⁾	
		PID_CP ²⁾	V2.0
		PID_ES ²⁾	
		AXIS_REF	

1) Das Technologieobjekt unterstützt die folgenden Openness-Funktionen nicht: Schreiben von Parametern.

2) Das Technologieobjekt unterstützt die folgenden Openness-Funktionen nicht: Enumerieren von Parametern, Suchen von Parametern, Lesen von Parametern, Schreiben von Parametern.

Hinweis

S7-1500 Motion Control

Die Technologieobjekte TO_OutputCam, TO_CamTrack und TO_MeasuringInput der S7-1500 werden getrennt behandelt.

Weitere Informationen finden Sie im Abschnitt "S7-1500 Motion Control (Seite 562)".

7.19.23.3 Überblick der Datentypen

Die Datentypen der Parameter von Technologieobjekten im TIA Portal sind C#-Datentypen in der Public API zugeordnet.

Datentypen

Die folgende Tabelle zeigt die Zuordnung der Datentypen:

Format	Datentyp im TIA Portal	Datentyp in C#
Binärzahlen	Bool	bool
	BBool	bool
	Byte	byte
	Word	ushort
	DWord	uint
	LWord	ulong
Ganzzahlen	SInt	sbyte
	Int	short
	DInt	int
	LInt	long
	USInt	byte
	UIInt	ushort
Gleitpunktzahlen	UDInt	uint
	ULInt	ulong
	Real	float
	LReal	double
	Time	double
	Char	char
Zeichenfolgen	WChar	char
	String	string
	WString	string
Hardware-Datentypen	HW_*	ushort
	Block_*	ushort

* Platzhalter für Erweiterung des Gerätetyps im TIA Portal-Projekt

7.19.23.4 Zusammensetzung von Technologieobjekten abfragen

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Eine PLC ist im Projekt ermittelt.
Siehe PLC-Target und HMI-Target abfragen (Seite 185)

Programmcode

Um alle Technologieobjekte einer PLC zu erhalten, ändern Sie den folgenden Programmcode:

```
// Retrieves all technology objects of a PLC
private static void GetTechnologicalObjectsOfPLC(PlcSoftware plcSoftware)
{
    TechnologicalInstanceDBGroup technologicalObjectGroup =
plcSoftware.TecnologicalObjectGroup;
    TechnologicalInstanceDBComposition technologicalObjects =
technologicalObjectGroup.TecnologicalObjects;
}
```

Siehe auch

Standard-Bibliotheken (Seite 45)

7.19.23.5 Technologieobjekt erstellen

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Eine PLC ist im Projekt ermittelt.
Siehe PLC-Target und HMI-Target abfragen (Seite 185)

Verwendung

Nur Technologieobjekte, die im Abschnitt Übersicht über Technologieobjekte und Versionen (Seite 543) aufgeführt sind, können erstellt werden. Bei nicht unterstützten Technologieobjekten oder ungültigen Parametern wird eine Ausnahme gemeldet.

Hinweis**S7-1500 Motion Control**

Die Technologieobjekte TO_OutputCam, TO_CamTrack und TO_MeasuringInput der S7-1500 werden getrennt behandelt.

Weitere Informationen finden Sie im Abschnitt "S7-1500 Motion Control (Seite 562)".

Programmcode

Um ein Technologieobjekt zu erstellen und es einer vorhandenen PLC hinzuzufügen, ändern Sie den folgenden Programmcode:

```
// Create a technology object and add to technology object composition
private static void CreateTechnologicalObject(PlcSoftware plcSoftware)
{
    TechnologicalInstanceDBComposition technologicalObjects =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects;

    string nameOfTO = "PID_Compact_1"; // How the technology object should be named
    string typeOfTO = "PID_Compact"; // How the technology object type is called, e.g. in
    // "Add new technology object"-dialog
    Version versionOfTO = new Version("2.3"); // Version of technology object
    TechnologicalInstanceDB technologicalObject = technologicalObjects.Create(nameOfTO,
typeOfTO, versionOfTO);
}
```

Mögliche Werte und Kombinationen aus Name, Typ und Version des Technologieobjekts sind im Abschnitt Übersicht über Technologieobjekte und Versionen (Seite 543) zu finden.

Siehe auch

[Standard-Bibliotheken \(Seite 45\)](#)

7.19.23.6 Technologieobjekt löschen**Voraussetzung**

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Eine PLC ist im Projekt ermittelt.
Siehe PLC-Target und HMI-Target abfragen (Seite 185)
- Das Technologieobjekt ist vorhanden.
Siehe Technologieobjekt finden (Seite 550)

Programmcode

Um ein Technologieobjekt zu löschen, ändern Sie den folgenden Programmcode:

```
// Delete a technology object from DB composition and from PLC
private static void DeleteTechnologicalObject(TechnologicalInstanceDB technologicalObject)
{
    technologicalObject.Delete();
}
```

Siehe auch

[Standard-Bibliotheken \(Seite 45\)](#)

7.19.23.7 Technologieobjekt übersetzen

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
[Siehe Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
[Siehe Projekt öffnen \(Seite 111\)](#)
- Eine PLC ist im Projekt ermittelt.
[Siehe PLC-Target und HMI-Target abfragen \(Seite 185\)](#)
- Das Technologieobjekt ist vorhanden.
[Siehe Technologieobjekt erstellen \(Seite 546\)](#)

Programmcode: Übersetzen eines Technologieobjekts

Um ein Technologieobjekt zu übersetzen, ändern Sie den folgenden Programmcode:

```
// Compile a single technology object
private static void CompileSingleTechnologicalObject(TechnologicalInstanceDB
technologicalObject)
{
    ICompilable singleCompile = technologicalObject.GetService<ICompilable>();
    CompilerResult compileResult = singleCompile.Compile();
}
```

Programmcode: Übersetzen der Technologieobjektgruppe

Um die Technologieobjektgruppe zu übersetzen, ändern Sie den folgenden Programmcode:

```
// Compile technology object group
private static void CompileTechnologicalObjectGroup(PlcSoftware plcSoftware)
{
    TechnologicalInstanceDBGрупп technologicalObjectGroup =
plcSoftware.TechnologicalObjectGroup;
    ICompilable groupCompile = technologicalObjectGroup.GetService<ICompilable>();
    CompilerResult compileResult = groupCompile.Compile();
}
```

Übersetzungsergebnisse

Die Übersetzungsergebnisse von Technologieobjekten werden rekursiv gespeichert.

Ein Beispiel für die rekursive Auswertung von Übersetzungsergebnissen finden Sie in Abschnitt "Projekt übersetzen (Seite 135)".

Siehe auch

Standard-Bibliotheken (Seite 45)

7.19.23.8 Technologieobjekte enumerieren

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Eine PLC ist im Projekt ermittelt.
Siehe PLC-Target und HMI-Target abfragen (Seite 185)

Programmcode

Um Technologieobjekte zu enumerieren, ändern Sie den folgenden Programmcode:

```
// Enumerate all technology objects
private static void EnumerateTechnologicalObjects(PlcSoftware plcSoftware)
{
    TechnologicalInstanceDBComposition technologicalObjects =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects;
    foreach (TechnologicalInstanceDB technologicalObject in technologicalObjects)
    {
        // Do something ...
    }
}
```

Siehe auch

[Standard-Bibliotheken \(Seite 45\)](#)

7.19.23.9 Technologieobjekt finden

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
[Siehe Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
[Siehe Projekt öffnen \(Seite 111\)](#)
- Eine PLC ist im Projekt ermittelt.
[Siehe PLC-Target und HMI-Target abfragen \(Seite 185\)](#)

Programmcode

Um ein spezifisches Technologieobjekt zu finden, ändern Sie den folgenden Programmcode:

```
// Find a specific technology object by its name
private static void FindTechnologicalObject(PlcSoftware plcSoftware)
{
    TechnologicalInstanceDBComposition technologicalObjects =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects;
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject = technologicalObjects.Find(nameOfTO);
}
```

Siehe auch

[Standard-Bibliotheken \(Seite 45\)](#)

7.19.23.10 Parameter eines Technologieobjekts enumerieren

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Eine PLC ist im Projekt ermittelt.
Siehe PLC-Target und HMI-Target abfragen (Seite 185)
- Ein Technologieobjekt ist vorhanden.
Siehe Technologieobjekt erstellen (Seite 546) oder Parameter eines Technologieobjekts finden (Seite 551)
- Das Technologieobjekt (Seite 543) unterstützt diese Funktion.

Programmcode

Um Parameter eines bestimmten Technologieobjekts zu enumerieren, ändern Sie den folgenden Programmcode:

```
// Enumerate parameters of a technology object
private static void EnumerateParameters(PlcSoftware plcSoftware)
{
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find(nameOfTO);

    foreach (TechnologicalParameter parameter in technologicalObject.Parameters)
    {
        // Do something ...
    }
}
```

Siehe auch

[Standard-Bibliotheken \(Seite 45\)](#)

[Technologieobjekt finden \(Seite 550\)](#)

7.19.23.11 Parameter eines Technologieobjekts finden

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

- Eine PLC ist im Projekt ermittelt.
Siehe PLC-Target und HMI-Target abfragen (Seite 185)
- Ein Technologieobjekt liegt vor.
Siehe Technologieobjekt erstellen (Seite 546)
- Das Technologieobjekt (Seite 543) unterstützt diese Funktion.

Programmcode

Um Parameter eines bestimmten Technologieobjekts zu finden, ändern Sie den folgenden Programmcode:

```
// Find parameters of a technology object
private static void FindParameterOfTechnologicalObject(PlcSoftware plcSoftware)
{
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find(nameOfTO);

    string nameOfParameter = "Config.InputUpperLimit";
    TechnologicalParameter parameter =
technologicalObject.Parameters.Find(nameOfParameter);
}
```

Parameter verschiedener Technologieobjekte

Parameter von SIMATIC S7-1200 Motion Control (Seite 554)

Parameter von SIMATIC S7-1500 Motion Control (Seite 562)

Parameter der PID-Regelung (Seite 581)

Zählparameter (Seite 582)

Parameter von Easy Motion Control (Seite 582)

Siehe auch

Standard-Bibliotheken (Seite 45)

Technologieobjekt finden (Seite 550)

7.19.23.12 Parameter eines Technologieobjekts lesen

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

- Eine PLC ist im Projekt ermittelt.
Siehe PLC-Target und HMI-Target abfragen (Seite 185)
- Ein Technologieobjekt liegt vor.
Siehe Technologieobjekt erstellen (Seite 546)
- Das Technologieobjekt (Seite 543) unterstützt diese Funktion.

Programmcode

Um Parameter eines bestimmten Technologieobjekts zu lesen, ändern Sie den folgenden Programmcode:

```
// Read parameters of a technology object
private static void ReadParameterOfTechnologicalObject(PlcSoftware plcSoftware)
{
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find(nameOfTO);

    string nameOfParameter = "Config.InputUpperLimit";
    TechnologicalParameter parameter =
technologicalObject.Parameters.Find(nameOfParameter);

    // Read from parameter
    string name = parameter.Name;
    object value = parameter.Value;
}
```

Siehe auch

- Standard-Bibliotheken (Seite 45)
- Technologieobjekt finden (Seite 550)

7.19.23.13 Parameter eines Technologieobjekts schreiben

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Eine PLC ist im Projekt ermittelt.
Siehe PLC-Target und HMI-Target abfragen (Seite 185)
- Ein Technologieobjekt liegt vor.
Siehe Technologieobjekt erstellen (Seite 546)
- Das Technologieobjekt (Seite 543) unterstützt diese Funktion.

Ausnahme

Eine EngineeringException wird gemeldet, wenn:

- Sie einen neuen Wert für einen Parameter festlegen, der keinen Schreibzugriff bietet.
- Ein neuer Wert für einen Parameter einen nicht unterstützten Typ hat.

Programmcode

Um Parameter eines spezifischen Technologieobjekts zu schreiben, ändern Sie den folgenden Programmcode:

```
// Write parameters of a technology object
private static void WriteParameterOfTechnologicalObject(PlcSoftware plcSoftware)
{
    string nameOfTO = "PID_Compact_1";
    TechnologicalInstanceDB technologicalObject =
plcSoftware.TechnologicalObjectGroup.TechnologicalObjects.Find(nameOfTO);

    string nameOfParameter = "Config.InputUpperLimit";
    TechnologicalParameter parameter =
technologicalObject.Parameters.Find(nameOfParameter);

    // Write to parameter if the value is writable
    object value = 3.0;
    parameter.Value = value;
}
```

Parameter der verschiedenen Technologieobjekte

[Parameter von SIMATIC S7-1200 Motion Control \(Seite 554\)](#)

[Parameter von SIMATIC S7-1500 Motion Control \(Seite 562\)](#)

[Parameter der PID-Regelung \(Seite 581\)](#)

[Zählparameter \(Seite 582\)](#)

[Parameter von Easy Motion Control \(Seite 582\)](#)

Siehe auch

[Standard-Bibliotheken \(Seite 45\)](#)

[Technologieobjekt finden \(Seite 550\)](#)

7.19.23.14 S7-1200 Motion Control

Version der Openness Engineering Library ändern

Wenn Sie "Openness\PublicAPI\V14 SP1\Siemens.Engineering.dll" mit dem TIA Portal V15 benutzen, dann funktioniert Ihre aktuelle Openness-Anwendung weiterhin.

Wenn Sie mit dem TIA Portal V15 zu "Openness\PublicAPI\V15\Siemens.Engineering.dll" wechseln, dann müssen Sie alle Zugriffe auf Array-Variablen für S7-1200 Motion Control anpassen.

Die betroffenen Arrays für TO_PositioningAxis sind in der folgenden Tabelle aufgelistet:

Zugriff in Openness < V15	Zugriff in Openness ≥ V15
_Sensor.Sensor[1].<alle Variablen>	_Sensor[1].<alle Variablen>
ControlPanel.Input.Command.Command[1].<alle Variablen>	ControlPanel.Input.Command[1].<alle Variablen>
ControlPanel.Output.Command.Command[1].<alle Variablen>	ControlPanel.Output.Command[1].<alle Variablen>
Internal.Internal[n].<alle Variablen>	Internal[n].<alle Variablen>
Sensor.Sensor[1].<alle Variablen>	Sensor[1].<alle Variablen>
StatusSensor.StatusSensor[1].<alle Variablen>	StatusSensor[1].<alle Variablen>

Die betroffenen Arrays für TO_CommandTable sind in der folgenden Tabelle aufgelistet:

Zugriff in Openness < V15	Zugriff in Openness ≥ V15
Command.Command[n].<alle Variablen>	Command[n].<alle Variablen>

PROFIdrives mit der Hardware-Adresse verschalten

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Eine S7-1200 CPU ist im Projekt angelegt.
- Ein PROFIdrive ist in dem Projekt verfügbar und mit der S7-1200 CPU verschaltet.
- Das Technologieobjekt liegt vor.
Siehe Technologieobjekt erstellen (Seite 546).

Programmcode

Ändern Sie den folgenden Programmcode, um einen PROFIdrive durch eine Hardware-Adresse mit „TO_PositioningAxis“ zu verschalten.

```
//An instance of the technology object axis is already available in the program before
private static void ConnectingDrive(TechnologicalInstanceDB technologicalObject)
{
    //Set axis to PROFIdrive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 1;

    //Set axis to drive mode
    technologicalObject.Parameters.Find("_Actor.Interface.DataConnection").Value = 0;

    //Set connection to adress of drive. The output will be set automatically.
    technologicalObject.Parameters.Find("_Actor.Interface.ProfiDriveIn").Value = "%I68.0";
    technologicalObject.Parameters.Find("Sensor[1].Interface.Number").Value = 1;
    // 1 = Encoder1, 2 = Encoder2;
}
```

Geber für PROFIdrives mit der Hardware-Adresse verschalten

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Eine S7-1200 CPU ist im Projekt angelegt.
- Ein PROFIdrive ist in dem Projekt verfügbar und mit der S7-1200 CPU verschaltet.
- Das Technologieobjekt liegt vor.
Siehe Technologieobjekt erstellen (Seite 546).

Programmcode

Ändern Sie den folgenden Programmcode, um einen Geber mit der Hardware-Adresse mit dem "TO_PositioningAxis" zu verschalten:

```
//An instance of the technology object axis is already available in the program before
private static void ConnectingEncoder(TechnologicalInstanceDB technologicalObject)
{
    //Set axis to PROFIdrive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 1;

    //Set the encoder mode
    technologicalObject.Parameters.Find("_Sensor[1].Interface.EncoderConnection").Value =
7;

    //Set axis to use PROFINET encoder
    technologicalObject.Parameters.Find("_Sensor[1].Interface.DataConnection").Value = 0;

    //Set connection to adress of drive. The output will be set automatically.
    technologicalObject.Parameters.Find("_Sensor[1].Interface.ProfiDriveIn").Value =
"%I68.0";
    technologicalObject.Parameters.Find("Sensor[1].Interface.Number").Value = 1;
    // 1 = Encoder1, 2 = Encoder2;
}
```

Analogantriebe mit der Hardware-Adresse verschalten

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Eine S7-1200 CPU ist im Projekt angelegt.
- Ein Analogantrieb ist in dem Projekt verfügbar und mit der S7-1200 CPU verschaltet.
- Das Technologieobjekt liegt vor.
Siehe Technologieobjekt erstellen (Seite 546).

Programmcode

Ändern Sie den folgenden Programmcode, um einen Analogantrieb mit der Hardware-Adresse mit dem "TO_PositioningAxis" zu verschalten:

```
//An instance of the technology object axis is already available in the program before
private static void ConnectingEncoder(TechnologicalInstanceDB technologicalObject)
{
    //Set axis to analog drive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 0;

    //Set axis to drive mode
    technologicalObject.Parameters.Find("_Actor.Interface.DataConnection").Value = 0;

    //Set connection to analog adress of drive
    technologicalObject.Parameters.Find("_Actor.Interface.Analog").Value = "%QW64";
}
```

Geber für Analogantriebe mit der Hardware-Adresse verschalten

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Eine S7-1200 CPU ist im Projekt angelegt.
- Ein Analogantrieb ist in dem Projekt verfügbar und mit der S7-1200 CPU verschaltet.
- Das Technologieobjekt liegt vor.
Siehe Technologieobjekt erstellen (Seite 546).

Programmcode

Ändern Sie den folgenden Programmcode, um einen Geber mit der Hardware-Adresse mit dem "TO_PositioningAxis" zu verschalten:

```
//An instance of the technology object axis is already available in the program before
//Connecting by High Speed Counter mode
private static void ConnectingEncoder(TechnologicalInstanceDB technologicalObject)
{
    //Set axis to analog drive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 0;

    //Set encoder for high-speed counter mode
    technologicalObject.Parameters.Find("_Sensor[1].Interface.EncoderConnection").Value =
4;
    technologicalObject.Parameters.Find("_Sensor[1].Interface.HSC.Name").Value = "HSC_1";
}

//An instance of the technology object axis is already available in the program before
//Connecting by PROFINET/PROFIBUS telegram
private static void ConnectingEncoder(TechnologicalInstanceDB
technologicalObject)
{
    //Set axis to analog drive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 0;
    //Set encoder for PROFINET/PROFIBUS mode
    technologicalObject.Parameters.Find("_Sensor[1].Interface.EncoderConnection").Value =
7;
    technologicalObject.Parameters.Find("_Sensor[1].Interface.DataConnection").Value =
"Encoder";
    technologicalObject.Parameters.Find("_Sensor[1].Interface.ProfiDriveIn").Value =
"%I68.0";
    technologicalObject.Parameters.Find("Sensor[1].Interface.Number").Value = 1;
    // 1 = Encoder1, 2 = Encoder2;
}
```

Antriebe mit einem Datenbaustein verschalten

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Eine S7-1200 CPU ist im Projekt angelegt.

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

- Ein Datenbaustein ist in dem Projekt verfügbar und auf "Nicht optimiert" eingestellt
Bei einem PROFIdrive-Achstyp enthält der Datenbaustein eine Variable dieses Typs, z. B. PD_TEL3.
Bei einem Analogantrieb enthält der Datenbaustein eine Variable des Datentyps Wort.
- Das Technologieobjekt liegt vor.
Siehe Technologieobjekt erstellen (Seite 546).

Programmcode

Ändern Sie den folgenden Programmcode, um einen PROFIdrive mit einem Datenbaustein mit dem "TO_PositioningAxis" zu verschalten.

```
//An instance of the technology object axis is already available in the program before
private static void ConfigureDrivewithDataBlock(TechnologicalInstanceDB
technologicalObject)
{
    //Set axis to PROFIdrive mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 1;

    //Set axis to data block mode
    technologicalObject.Parameters.Find("_Actor.Interface.DataConnection").Value = 1;

    //Set the tag in the data block
    technologicalObject.Parameters.Find("_Actor.Interface.DataBlock").Value =
"Data_block_1.Member_of_type_PD_TEL3";
}
```

Programmcode

Ändern Sie den folgenden Programmcode, um einen Analogantrieb mit einem Datenbaustein mit dem "TO_PositioningAxis" zu verschalten:

```
//An instance of the technology object axis is already available in the program before
//Connecting an analog drive with data block.
private static void ConfigureDrivewithDataBlock(TechnologicalInstanceDB
technologicalObject)
{
    //Set axis to analog mode
    technologicalObject.Parameters.Find("Actor.Type").Value = 0;

    //Set the tag in the data block
    technologicalObject.Parameters.Find("_Actor.Interface.Analog").Value =
>Data_block_1.Static_1";
}
```

Geber mit einem Datenbaustein verschalten

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Eine S7-1200 CPU ist im Projekt angelegt.
- Ein Datenbaustein ist in dem Projekt verfügbar und auf "Nicht optimiert" eingestellt
Bei PROFIdrive enthält der Datenbaustein eine Variable dieses Typs, z. B. PD_TEL3.
- Das Technologieobjekt liegt vor.
Siehe Technologieobjekt erstellen (Seite 546).

Programmcode

Ändern Sie den folgenden Programmcode, um einen Geber mit einem Datenbaustein zu verschalten:

```
//An instance of the technology object axis is already available in the program before
private static void ConfigureEncoderWithDataBlock(TechnologicalInstanceDB
technologicalObject)
{
    //Set axis to PROFIdrive mode depending by axis type. 1 = PROFIdrive, 0 = Analog Drive.
    technologicalObject.Parameters.Find("Actor.Type").Value = 1;

    //Set the encoder mode
    technologicalObject.Parameters.Find("_Sensor[1].Interface.EncoderConnection").Value =
7;

    //Set axis to data block mode
    technologicalObject.Parameters.Find("_Sensor[1].Interface.DataConnection").Value = 1;

    //Set the tag in the data block. For PD_TEL3 and PD_TEL4 "Encoder1" or "Encoder2".
    technologicalObject.Parameters.Find("_Sensor[1].Interface.DataBlock").Value =
>Data_block_1.Member_of_Type_PD_TEL3";
}
```

Parameter für TO_PositioningAxis und TO_CommandTable

Eine Liste mit allen verfügbaren Variablen finden Sie im Funktionshandbuch SIMATIC STEP 7 S7-1200 Motion Control im Internet (<https://support.industry.siemens.com/cs/ww/de/view/109754206>).

Hinweis

Im TIA Portal finden Sie in der Parametersicht der Technologieobjektkonfiguration die Spalte "Name in Openness".

7.19.23.15 S7-1500 Motion Control

TO_OutputCam, TO_CamTrack und TO_MeasuringInput erstellen und finden

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Eine S7-1500 CPU ist im Projekt angelegt.
- Ein Technologieobjekt des Typs TO_PositioningAxis, TO_SynchronousAxis oder TO_ExternalEncoder ist im Projekt angelegt.

Anwendung

Die Technologieobjekte Nocken, Nockenspur und Messtaster sind mit einem der Technologieobjekte Positionierachse, Gleichlaufachse oder Externer Geber verschaltet. Um auf die Technologieobjekte Nocken, Nockenspur oder Messtaster zuzugreifen, verwenden Sie den Service OutputCamMeasuringInputContainer.

Programmcode: Technologieobjekte Nocken, Nockenspur und Messtaster erstellen und finden

Ändern Sie den folgenden Programmcode, um das Technologieobjekt Nocken, Nockenspur oder Messtaster zu erstellen oder zu finden.

```
/*An instance of the technology object under which the TO_OutputCam, TO_CamTrack or
TO_MeasuringInput should be created is already available in the program before*/
private static void CreateFind_OutputcamCamtrackMeasuringinput(TechnologicalInstanceDB
technologyObject)
{
    //Retrieve service OutputCamMeasuringInputContainer
    OutputCamMeasuringInputContainer container =
    technologyObject.GetService<OutputCamMeasuringInputContainer>();
    //Get access to TO_OutputCam / TO_CamTrack container
    TechnologicalInstanceDBComposition outputcamCamtrackContainer = container.OutputCams;

    //Find technology object TO_OutputCam or TO_CamTrack
    TechnologicalInstanceDB outputCam = outputcamCamtrackContainer.Find("OutputCamName");
    TechnologicalInstanceDB camTrack = outputcamCamtrackContainer.Find("CamTrackName");

    //Create new technology object TO_OutputCam or TO_CamTrack
    TechnologicalInstanceDB newOutputCam =
    outputcamCamtrackContainer.Create("NewOutputCamName", "TO_OutputCam",
    new Version(3, 0));
    TechnologicalInstanceDB newCamTrack =
    outputcamCamtrackContainer.Create("NewCamTrackName", "TO_CamTrack", new Version(3, 0));

    //Get access to TO_MeasuringInput container
    TechnologicalInstanceDBComposition measuringInputContainer = container.MeasuringInputs;

    //Find technology object TO_MeasuringInput
    TechnologicalInstanceDB measuringInput =
    measuringInputContainer.Find("MeasuringInputName");

    //Create new technology object TO_MeasuringInput
    TechnologicalInstanceDB newMeasuringInput =
    measuringInputContainer.Create("NewMeasuringInput", "TO_MeasuringInput",
    new Version(3, 0));
}
```

Parameter von S7-1500 Motion Control

Die meisten Parameter von S7-1500 Motion Control Technologieobjekten sind direkt auf Datenbausteinvariablen abgebildet, doch es gibt auch einige zusätzliche Parameter, die nicht direkt auf Datenbausteine abgebildet sind. In Openness haben die direkt abgebildeten Parameter die gleiche Reihenfolge wie in der "Datennavigation" in der Parametersicht des Technologieobjekts. Nach den direkt abgebildeten Parametern folgen die zusätzlichen Parameter in der Reihenfolge der Tabelle.

Parameter, die direkt auf Technologieobjekt-Datenbausteinvariablen abgebildet sind:

Sie haben Zugriff auf alle Technologieobjekt-Datenbausteinvariablen, die im Allgemeinen beschrieben sind, außer:

- Schreibgeschützte Variablen
- Variablen des Datentyps VREF
- Variablen mit der Struktur „InternalToTrace“
- Variablen mit der Struktur „ControlPanel“

Sie können zusätzliche Informationen über die direkt abgebildeten Parameter in folgenden Anhängen finden:

- Funktionshandbuch SIMATIC S7-1500 Motion Control:
<https://support.industry.siemens.com/cs/ww/en/view/109749262> (<https://support.industry.siemens.com/cs/ww/de/view/109749262>)
- Funktionshandbuch SIMATIC S7-1500T Motion Control:
<https://support.industry.siemens.com/cs/ww/en/view/109749263> (<https://support.industry.siemens.com/cs/ww/de/view/109749263>)
- Funktionshandbuch SIMATIC S7-1500T Kinematics Functions:
<https://support.industry.siemens.com/cs/ww/en/view/109749264> (<https://support.industry.siemens.com/cs/ww/de/view/109749264>)

Einige Technologieparameter, die Datenbausteinvariablen abbilden, müssen in der PublicAPI schreibbar gemacht werden. Die zulässigen Werte sind die gleichen wie die Werte für die zugrundeliegenden Datenbausteinvariablen. Die betroffenen Parameter sind in den folgenden Tabellen aufgelistet:

Name in Openness	Datentyp	TO_SpeedAxis	TO_Positionin-gAxis	TO_Synchron-ousAxis	TO_ExternalEn-coder
Actor.Type	int	X	X	X	-
Actor.Interface.EnableDri-veOutput	bool	X	X	X	-
Actor.Interface.DriveRea-dyInput	bool	X	X	X	-
Actor.DataAdaptionOffline	bool	X	X	X	-
VirtualAxis.Mode	uint	X	X	X	-
Sensor[n].DataAdaptionOffli-ne ¹⁾	bool	-	X	X	-
Sensor[n].Existen ¹⁾	bool	-	X	X	-
Sensor[n].Interface.Number ¹⁾	uint	-	X	X	-
Sensor[n].Type ¹⁾	int	-	X	X	-
Sensor.DataAdaptionOffline	bool	-	-	-	X
Sensor.Interface.Number	uint	-	-	-	X
Sensor.Type	int	-	-	-	X

Name in Openness	Datentyp	TO_OutputCam	TO_MeasuringInput	TO_Kinematics ²⁾
Interface.LogicOperation	int	X	-	-
Parameter.MeasuringInput-Type	int	-	X	-
Kinematics.TypeOfKinematics	int	-	-	X
MotionQueue.MaxNumberOfCommands	int	-	-	X

1) S7-1500 CPU: n=1; S7-1500T CPU: 1≤n≤4

2) S7-1500T CPU

Parameter, die nicht direkt auf Technologieobjekt-Datenbausteinvariablen abgebildet sind:

Für S7-1500 Motion Control Technologieobjekte sind folgende zusätzliche Parameter verfügbar, die nicht direkt auf Datenbausteinvariablen abgebildet sind:

Name in Openness	Name in Funktionssicht	Möglicher Wert	Datentyp in Openness	TO_SpeedAxis	TO_PositioningAxis TO_SynchronousAxis	TO_Externa-IEncoder
_Properties.Motion-Type	Achstyp bzw. „Technische Einheit der Position“	0: Linear 1: Rotierend	int	-	X	X
_Units.LengthUnit	Positionseinheiten	Siehe Variable Units.LengthUnit ³⁾	uint	-	X	X
_Units.VelocityUnit	Geschwindigkeitseinheiten	Siehe Variable Units.VelocityUnit ³⁾	uint	X	X	X
_Units.TorqueUnit	Drehmomentseinheiten	Siehe Variable Units.TorqueUnit ³⁾	uint	X	X	-
_Units.ForceUnit	Krafteinheiten	Siehe Variable Units.ForceUnit ³⁾	uint	-	X	-
_Actor.Interface.Telegram	Antriebstelegramm	Telegramm Nummer ⁴⁾	uint	X	X	-
_Actor.Interface.EnableDriveOutputAddress	Adresse für Ausgang Antriebsfreigabe	PublicAPI-Objekt	SW.Tags.PlcTag	X	X	-
_Actor.Interface.DriveReadyInputAddress	Adresse für Eingang Antriebsfreigabe	PublicAPI-Objekt	SW.Tags.PlcTag	X	X	-
_Sensor[n].Interface.Telegram ⁵⁾	Gebertelegramm	Telegramm Nummer ⁴⁾	uint	-	X	-

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Name in Openness	Name in Funktionssicht	Möglicher Wert	Datentyp in Openness	TO_SpeedAxis	TO_PositioningAxis TO_SynchronousAxis	TO_Externa- lEncoder
_Sensor[n].ActiveHoming.DigitalInputAddress ⁵⁾	Digitaleingang	PublicAPI-Objekt	SW.Tags.PlcTag	-	X	-
_Sensor[n].PassiveHoming.DigitalInputAddress ⁵⁾	Digitaleingang	PublicAPI-Objekt	SW.Tags.PlcTag	-	X	-
_PositionLimits_HW.MinSwitchAddress	Adresse negativer Hardware-Endschalter	PublicAPI-Objekt	SW.Tags.PlcTag	-	X	-
_PositionLimits_HW.MaxSwitchAddress	Adresse positiver Hardware-Endschalter	PublicAPI-Objekt	SW.Tags.PlcTag	-	X	-
_Sensor.Interface.Telegram	Gebertelegramm	Telegramm Nummer ⁴⁾	uint	-	-	X
_Sensor.PassiveHoming.DigitalInputAddress	Digitaleingang	PublicAPI-Objekt	SW.Tags.PlcTag	-	-	X

Für die Technologieobjekte Nocken, Nockenspur und Messtaster ist der folgende zusätzliche Parameter verfügbar:

Name in Openness	Name in Funktionssicht	Möglicher Wert	Datentyp
_AssociatedObject	Zugeordnete Achse oder Externer Geber	PublicAPI-Objekt	SW.TechnologicalObjects.TechnologicalInstanceDB

Beim Technologieobjekt Kinematik sind die folgenden zusätzlichen Parameter verfügbar (S7-1500T):

Name in Openness	Name in Funktionssicht	Möglicher Wert	Datentyp
_KinematicsAxis[1...4]	Achse 1 - 3, Ausrichtungssachse	Achse, die mit TO_Kinematics-Objekten verschaltet werden kann	SW.TechnologicalObjects.TechnologicalInstanceDB
_Units.LengthUnit	Maßeinheit > Position	Siehe Variable Units.LengthUnit ³⁾	uint
_Units.LengthVelocityUnit	Maßeinheit > Geschwindigkeit	Siehe Variable Units.LengthVelocityUnit ³⁾	uint
_Units.AngleUnit	Maßeinheit > Winkel	Siehe Variable Units.AngleUnit ³⁾	uint
_Units.AngleVelocityUnit	Maßeinheit > Winkelgeschwindigkeit	Siehe Variable Units.AngleVelocityUnit ³⁾	uint

3) Mögliche Werte sind im Funktionshandbuch S7-1500 Motion Control im Kapitel Variablen Units (TO) beschrieben

4) Mögliche Werte sind im Funktionshandbuch S7-1500 Motion Control im Kapitel PROFIdrive Telegramme beschrieben

5) S7-1500 CPU: n=1; S7-1500T CPU: 1≤n≤4

Programmcode: Direkt abgebildete Datenbausteineinvariablen

Ändern Sie den folgenden Programmcode, um auf die direkt abgebildeten Parameter zuzugreifen:

```
//An instance of the technology object is already available in the program before
private static void ReadWriteDataBlockTag(TechnologicalInstanceDB technologyObject)
{
    //Read value from data block tag "ReferenceSpeed"
    double value =
        (double)technologyObject.Parameters.Find("Actor.DriveParameter.ReferenceSpeed").Value;

    //Write data block tag "ReferenceSpeed"
    technologyObject.Parameters.Find("Actor.DriveParameter.ReferenceSpeed").Value = 3000.0;
}
```

Programmcode: Weitere Parameter

Ändern Sie den folgenden Programmcode, um auf die zusätzlichen Parameter zuzugreifen:

```
//An instance of the technology object is already available in the program before
private static void ReadWriteAdditionalParameter(TechnologicalInstanceDB technologyObject)
{
    //Read additional parameter "_Properties.MotionType"
    uint value = (uint)technologyObject.Parameters.Find("_Properties.MotionType").Value;

    //Write additional parameter "_Properties.MotionType"
    technologyObject.Parameters.Find("_Properties.MotionType").Value = 1;
}
```

Weitere Informationen

Weitere Informationen finden Sie hier:

- Funktionshandbuch SIMATIC S7-1500 Motion Control:
<https://support.industry.siemens.com/cs/ww/en/view/109749262> (<https://support.industry.siemens.com/cs/ww/de/view/109749262>)
- Funktionshandbuch SIMATIC S7-1500T Motion Control:
<https://support.industry.siemens.com/cs/ww/en/view/109749263> (<https://support.industry.siemens.com/cs/ww/de/view/109749263>)
- Funktionshandbuch SIMATIC S7-1500T Kinematics Functions:
<https://support.industry.siemens.com/cs/ww/en/view/109749264> (<https://support.industry.siemens.com/cs/ww/de/view/109749264>)

Antriebe verschalten

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Eine S7-1500 CPU ist im Projekt angelegt.
- Ein Technologieobjekt des Typs TO_SpeedAxis, TO_PositioningAxis oder TO_SynchronousAxis ist im Projekt angelegt.
- Ein Antrieb ist im Projekt angelegt.

Anwendung

Um eine Achse mit einem Antrieb zu verschalten, ist es erforderlich, mehrere Werte zusammen in einem einzigen Aufruf anzugeben. Der Typ der Public API AxisEncoderHardwareConnectionInterface bietet die folgenden Methoden, die zur Verschaltung oder Trennung der Aktor- oder Sensorschnittstellen verwendet werden können:

Methode	Beschreibung
void Connect(HW.DeviceItem moduleInOut)	Verschaltet mit Eingangs- und Ausgangsadressen an einem Modul
void Connect(HW.DeviceItem moduleIn, HW.DeviceItem moduleOut)	Verschaltet mit Eingangs- und Ausgangsadressen an getrennten Modulen
void Connect(HW.DeviceItem moduleIn, HW.DeviceItem moduleOut, ConnectOption connectOption)	Verschaltet mit Eingangs- und Ausgangsadressen an getrennten Modulen unter Angabe einer zusätzlichen ConnectOption
void Connect(HW.Channel channel)	Verschaltet mit einem Kanal
void Connect(int addressIn, int addressOut, ConnectOption connectOption)	Verschaltet Bit-Adressen direkt
void Connect(string pathToDBMember)	Verschaltet mit einer DatenbausteinvARIABLEN
void Connect(SW.Tags.PlcTag outputTag)	Verschaltet mit einer CPU-VARIABLEN
void Disconnect()	Trennt eine existierende Verschaltung

Hinweis

Automatische Verschaltung

Bitte beachten Sie, dass hier das gleiche Verhalten wie bei der Anwenderschnittstelle angewendet wird. Immer wenn die Aktorschnittstelle über eine der folgenden Verschaltungsmethoden verschaltet wird und das Telegramm ein Sensor teil oder Telegramm 750 enthält, werden diese Teile automatisch verschaltet.

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Sie können die folgenden schreibgeschützten Attribute verwenden, um festzustellen, wie das Technologieobjekt verschaltet wird. Die entsprechenden Verschaltungswerte werden nur eingestellt, wenn eine Verschaltung dieser spezifischen Art besteht.

Attribut	Datentyp	Beschreibung
IsConnected	bool	TRUE: Schnittstelle ist verschaltet FALSE: Schnittstelle ist nicht verschaltet
InputOutputModule	HW.Deviceltem	Verschaltetes Modul, das Eingangs- und Ausgangsadressen enthält
InputModule	HW.Deviceltem	Verschaltetes Modul, das Eingangsadressen enthält Der Wert wird auch im Falle einer bestehenden Verschaltung zu einem Modul, das Eingangs- und Ausangadsressen enthält, eingestellt.
OutputModule	HW.Deviceltem	Verschaltetes Modul, das Ausangadsressen enthält Der Wert wird auch im Falle einer bestehenden Verschaltung zu einem Modul, das Eingangs- und Ausangadsressen enthält, eingestellt.
InputAddress	int	Logische EingangAdresse des verschalteten Objekts; zum Beispiel 256.
OutputAddress	int	Logische AusgangAdresse des verschalteten Objekts; zum Beispiel 256.
ConnectOption	ConnectOption	Wert der bei der Verbindungsherstellung eingestellten ConnectOption: <ul style="list-style-type: none">• Standardeinstellung Es können nur Module ausgewählt werden, die als gültige Verbindungspartner erkannt werden.• AllowAllModules Entspricht dem Markieren von "Zeige alle Module" in der Anwenderschnittstelle.
Channel	HW.Channel	Verschalteter Kanal
PathToDBMember	string	Verschaltete Technologieobjekt-Datenbausteinvariable
OutputTag	SW.Tags.PlcTag	Verschaltete CPU-Variable (analoge Verschaltung)
SensorIndexInActor-Telegram	int	Verschalteter Sensorsteil in Aktortelegramm Das Attribut ist nur für Sensorschnittstellen relevant. 0: Geber ist nicht verschaltet 1: Geber ist mit der ersten Sensorschnittstelle im Telegramm verschaltet 2: Geber ist mit der zweiten Sensorschnittstelle im Telegramm verschaltet Der Wert der Aktorschnittstelle ist stets 0.

Hinweis**Zugriff auf die Sensorschnittstelle**

Um auf die Sensorschnittstelle zuzugreifen, können Sie SensorInterface[m] mit $0 \leq m \leq 3$ verwenden.

Programmcode: void Connect(HW.DeviceItem moduleInOut)

Ändern Sie den folgenden Programmcode, um ein gemischtes Modul, das Eingangs- und Ausgangsadressen enthält, zu verschalten.

```
//An instance of technology object and device item is already available in the program  
before  
private static void UseServiceAxisHardwareConnectionProvider(TechnologicalInstanceDB  
technologyObject, DeviceItem devItem)  
{  
    //Retrieve service AxisHardwareConnectionProvider  
    AxisHardwareConnectionProvider connectionProvider =  
    technologyObject.GetService<AxisHardwareConnectionProvider>();  
  
    //Connect ActorInterface with DeviceItem  
    connectionProvider.ActorInterface.Connect(devItem);  
  
    //Connect first SensorInterface with DeviceItem  
    connectionProvider.SensorInterface[0].Connect(devItem);  
  
    //Check ConnectionState of ActorInterface  
    bool actorInterfaceConnectionState = connectionProvider.ActorInterface.IsConnected;  
  
    //Check ConnectionState of first SensorInterface  
    bool sensorInterfaceConnectionState =  
    connectionProvider.SensorInterface[0].IsConnected;  
}
```

Telegramm 750 verschalten

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Eine S7-1500 CPU ist im Projekt angelegt.
- Ein Technologieobjekt des Typs TO_SpeedAxis, TO_PositioningAxis oder
TO_SynchronousAxis V4.0 ist im Projekt angelegt.
- Ein Antrieb, der Telegramm 750 unterstützt, ist im Projekt angelegt.

Anwendung

Wenn Telegramm 750 nach Verschalten des Antriebs und der Achse hinzugefügt wurde, ist es nötig, Telegramm 750 getrennt zu verschalten. EnableTorqueData ist automatisch auf TRUE gesetzt. Der Typ der Public API TorqueHardwareConnectionInterface bietet die folgenden Methoden, die zur Verschaltung oder Trennung von Telegramm 750 verwendet werden können:

Methode	Beschreibung
void Connect(HW.Deviceltem moduleInOut)	Verschaltet mit Eingangs- und Ausgangsadressen an einem Modul
void Connect(HW.Deviceltem moduleIn, HW.Deviceltem moduleOut)	Verschaltet mit Eingangs- und Ausgangsadressen an getrennten Modulen
void Connect(HW.Deviceltem moduleIn, HW.Deviceltem moduleOut, ConnectOption connectOption)	Verschaltet mit Eingangs- und Ausgangsadressen an getrennten Modulen unter Angabe einer zusätzlichen ConnectOption
void Connect(int addressIn, int addressOut, ConnectOption connectOption)	Verschaltet Bit-Adressen direkt
void Connect(string pathToDBMember)	Verschaltet mit einer Datenbausteinvariablen
void Disconnect()	Trennt eine existierende Verschaltung

Die TorqueHardwareConnectionInterface kann über die Eigenschaft TorqueInterface am Typ AxisHardwareConnectionProvider abgerufen werden. Wird die Verschaltung mit Telegramm 750 nicht unterstützt, ist der Wert der Eigenschaft "null".

Ist der Antrieb über Datenbausteinvariablen verschaltet, können Sie Telegramm 750 nicht mittels Modul verschalten. Sie können die folgenden schreibgeschützten Attribute verwenden, um festzustellen, wie das Technologieobjekt verschaltet wird. Die entsprechenden Verschaltungswerte werden nur eingestellt, wenn eine Verschaltung dieser spezifischen Art besteht:

Attribut	Datentyp	Beschreibung
IsConnected	bool	TRUE: Schnittstelle ist verschaltet FALSE: Schnittstelle ist nicht verschaltet
InputOutput-Module	HW.Deviceltem	Verschaltetes Modul, das Eingangs- und Ausgangsadressen enthält
InputModule	HW.Deviceltem	Verschaltetes Modul, das Eingangsadressen enthält. Der Wert wird auch im Fall einer bestehenden Verschaltung zu einem Modul, das Eingangs- und Ausgangsadressen enthält, eingestellt.
OutputModu- le	HW.Deviceltem	Verschaltetes Modul, das Ausgangsadressen enthält. Der Wert wird auch im Fall einer bestehenden Verschaltung zu einem Modul, das Eingangs- und Ausgangsadressen enthält, eingestellt.
InputAddress	int	Logische EingangAdresse des verschalteten Objekts, beispielsweise 256

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Attribut	Datentyp	Beschreibung
OutputAddress	int	Logische Ausgangsadresse des verschalteten Objekts, beispielsweise 256
ConnectOption	ConnectOption	Wert der bei der Verschaltungsherstellung eingestellten ConnectOption: <ul style="list-style-type: none"> • Standard Es können nur Module ausgewählt werden, die als gültige Verschaltungspartner erkannt werden. • AllowAllModules Entspricht dem Auswählen von "Zeige alle Module" in der Anwenderschnittstelle.
PathToDB-Member	string	Verschaltete Technologieobjekt-Datenbausteinvariable

Programmcode: Telegramm 750 verschalten

Ändern Sie den folgenden Programmcode, um ein gemischtes Modul, das Eingangs- und Ausgangsadressen enthält, zu verschalten:

```
//An instance of technology object and device item is already available in the program
before
private static void ConnectTorqueInterface(TechnologicalInstanceDB technologyObject,
DeviceItem devItem)
{
    //Retrieve service AxisHardwareConnectionProvider
    AxisHardwareConnectionProvider connectionProvider =
    technologyObject.GetService<AxisHardwareConnectionProvider>();
    //Connect TorqueInterface with DeviceItem
    connectionProvider.TorqueInterface.Connect(devItem);
}
```

Geber verschalten

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Eine S7-1500 CPU ist im Projekt angelegt.
- Ein Technologieobjekt des Typs TO_ExternalEncoder ist im Projekt angelegt.
- Im Projekt wird ein Objekt bestimmt, welches PROFIdrive Telegramm 81 oder 83 bereitstellt.

Anwendung

Um ein Technologieobjekt Externer Geber mit der Geber-Hardware zu verschalten, ist es erforderlich, mehrere Werte zusammen in einem einzigen Aufruf anzugeben. Der Typ der Public API AxisEncoderHardwareConnectionInterface bietet die folgenden Methoden, die zur Verschaltung oder Trennung der Sensorschnittstelle verwendet werden können.

Methode	Beschreibung
void Connect(HW.DeviceItem moduleInOut)	Verschaltet mit Eingangs- und Ausgangsadressen an einem Modul
void Connect(HW.DeviceItem moduleIn, HW.DeviceItem moduleOut)	Verschaltet mit Eingangs- und Ausgangsadressen an getrennten Modulen
void Connect(HW.DeviceItem moduleIn, HW.DeviceItem moduleOut, ConnectOption connectOption)	Verschaltet mit Eingangs- und Ausgangsadressen an getrennten Modulen unter Angabe einer zusätzlichen ConnectOption
void Connect(HW.Channel channel)	Verschaltet mit einem Kanal
void Connect(int addressIn, int addressOut, ConnectOption connectOption)	Verschaltet Bit-Adressen direkt
void Connect(string pathToDBMember)	Verschaltet mit einer Datenbausteinvariablen
void Connect(SW.Tags.PlcTag outputTag)	Nicht relevant für das Verschalten von Gebern
void Disconnect()	Trennt eine existierende Verschaltung

Sie können die folgenden schreibgeschützten Attribute verwenden, um festzustellen, wie das Technologieobjekt verschaltet wird. Die entsprechenden Verschaltungswerte werden nur eingestellt, wenn eine Verschaltung dieser spezifischen Art besteht.

Attribut	Datentyp	Beschreibung
IsConnected	bool	TRUE: Schnittstelle ist verschaltet FALSE: Schnittstelle ist nicht verschaltet
InputOutputModule	HW.DeviceItem	Verschaltetes Modul, das Eingangs- und Ausgangsadressen enthält
InputModule	HW.DeviceItem	Verschaltetes Modul, das Eingangsadressen enthält Der Wert wird auch im Falle einer bestehenden Verschaltung zu einem Modul, das Eingangs- und Ausgangsadressen enthält, eingestellt.
OutputModule	HW.DeviceItem	Verschaltetes Modul, das Ausgangsadressen enthält Der Wert wird auch im Falle einer bestehenden Verschaltung zu einem Modul, das Eingangs- und Ausgangsadressen enthält, eingestellt.
InputAddress	int	Die logische EingangAdresse des verschalteten Objekts ist beispielsweise 256.
OutputAddress	int	Die logische AusgangAdresse des verschalteten Objekts ist beispielsweise 256.
ConnectOption	ConnectOption	Wert der bei der Verbindungsherstellung eingestellten ConnectOption: <ul style="list-style-type: none">• Default Es können nur Module ausgewählt werden, die als gültige Verbindungspartner erkannt werden.• AllowAllModules Entspricht dem Markieren von "Zeige alle Module" in der Anwenderschnittstelle.
Channel	HW.Channel	Verschalteter Kanal
PathToDBMember	string	Verschaltete Datenbausteinvariable

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Attribut	Datentyp	Beschreibung
OutputTag	SW.Tags.PlcTag	Nicht relevant für das Verschalten von Gebern
SensorIndexInActor-Telegram	int	Verschaltetes Sensorsortelegramm Das Attribut ist nur für Sensorschnittstellen relevant. 0: Geber ist nicht verschaltet 1: Geber ist mit der ersten Sensorschnittstelle im Telegramm verschaltet 2: Geber ist mit der zweiten Sensorschnittstelle im Telegramm verschaltet Der Wert der Aktorschnittstelle ist stets 0.

Programmcode: Geber verschalten

Ändern Sie den folgenden Programmcode, um ein Technologieobjekt Externer Geber zu verschalten:

```
//An instance of technology object and device item is already available in the program before
private static void UseServiceEncoderHardwareConnectionProvider(TechnologicalInstanceDB
technologyObject, DeviceItem devItem)
{
    //Retrieve service EncoderHardwareConnectionProvider
    EncoderHardwareConnectionProvider connectionProvider =
    technologyObject.GetService<EncoderHardwareConnectionProvider>();

    //Connect SensorInterface with DeviceItem
    connectionProvider.SensorInterface.Connect(devItem);

    //Check ConnectionState of SensorInterface
    bool sensorInterfaceConnectionState = connectionProvider.SensorInterface.IsConnected;
}
```

Nocken und Nockenspur mit Hardware verschalten**Voraussetzung**

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Eine S7-1500 CPU ist im Projekt angelegt.
- Ein Technologieobjekt des Typs TO_OutputCam oder TO_CamTrack ist im Projekt angelegt.
- Ein Digitalausgabemodul ist im Projekt angelegt, zum Beispiel TM Timer DIDQ.

Anwendung

Um ein Technologieobjekt Nocken oder Nockenspur mit einem Digitalausgang zu verschalten, ist es erforderlich, mehrere Werte zusammen in einem einzigen Aufruf anzugeben. Der Typ der Public API OutputCamHardwareConnectionProvider bietet die folgenden Methoden, die zur Verschaltung oder Trennung der Aktor- oder Sensorschnittstellen verwendet werden können:

Methode	Beschreibung
void Connect(HW.Channel channel)	Verschaltet mit einem Kanal
void Connect(SW.Tags.PlcTag outputTag)	Verschaltet mit einer CPU-Variablen
void Connect(int address)	Verschaltet Bit-Adressen direkt
void Disconnect()	Trennt eine existierende Verschaltung

Sie können die folgenden schreibgeschützten Attribute verwenden, um festzustellen, wie das Technologieobjekt verschaltet wird:

Attribut	Datentyp	Beschreibung
IsConnected	bool	TRUE: Technologieobjekt ist verschaltet FALSE: Technologieobjekt ist nicht verschaltet
Channel	HW.Channel	Verschalteter Kanal
OutputTag	SW.Tags.PlcTag	Verschaltete CPU-Variable
OutputAddress	int	Die logische Ausgangsadresse des verschalteten Objekts ist beispielsweise 256.

Programmcode: Technologieobjekt Nocken oder Nockenspur verschalten

Ändern Sie den folgenden Programmcode, um ein Technologieobjekt Nocken oder Nockenspur zu verschalten:

```
//An instance of technology object and channel item is already available in the program
before
private static void UseServiceOutputCamHardwareConnectionProvider(TechnologicalInstanceDB
technologyObject, Channel channel)

{
    //Retrieve service OutputCamHardwareConnectionProvider
    OutputCamHardwareConnectionProvider connectionProvider =
    technologyObject.GetService<OutputCamHardwareConnectionProvider>();

    //Connect technology object with Channel
    connectionProvider.Connect(channel);

    //Check ConnectionState of technology object
    bool connectionState = connectionProvider.IsConnected;
}
```

Messtaster mit Hardware verschalten

Voraussetzung

- Die Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Eine S7-1500 CPU ist im Projekt angelegt.
- Ein Technologieobjekt des Typs TO_MeasuringInput ist im Projekt angelegt.
- Ein Digitaleingabemodul ist am Antrieb oder im Projekt angelegt, zum Beispiel TM Timer DIDQ.

Anwendung

Um ein Technologieobjekt Messtaster mit einem Digitaleingang zu verschalten, ist es erforderlich, mehrere Werte zusammen in einem einzigen Aufruf anzugeben. Der Typ der Public API MeasuringInputHardwareConnectionProvider bietet die folgenden Methoden, die zur Verschaltung oder Trennung der Aktor- oder Sensorschnittstellen verwendet werden können:

Methode	Beschreibung
void Connect(HW.Channel channel)	Verschaltet mit einem Kanal
void Connect(HW.Deviceltem moduleIn, int channelIndex)	Verschaltet mit einem Modul und legt einen zusätzlichen Kanalindex fest
void Connect(int address)	Verschaltet Bit-Adressen direkt
void Disconnect()	Trennt eine existierende Verschaltung

Sie können die folgenden schreibgeschützten Attribute verwenden, um festzustellen, wie das Technologieobjekt verschaltet wird:

Attribut	Datentyp	Beschreibung
IsConnected	bool	TRUE: Technologieobjekt ist verschaltet FALSE: Technologieobjekt ist nicht verschaltet
InputModule	HW.Deviceltem	Verschaltetes Modul, das Eingangsadressen enthält
ChannelIndex	int	Index des verschalteten Kanals hinsichtlich des Eingabemoduls
Channel	HW.Channel	Verschalteter Kanal
InputAddress	int	Die logische Eingangsadresse des verschalteten Objekts ist beispielsweise 256.

Programmcode: Technologieobjekt Messtaster verschalten

Ändern Sie den folgenden Programmcode, um ein Technologieobjekt Messtaster zu verschalten:

```
//An instance of technology object and channel item is already available in the program  
before  
private static void  
UseServiceMeasuringInputHardwareConnectionProvider(TechnologicalInstanceDB  
technologyObject, Channel channel)  
{  
    //Retrieve service MeasuringInputHardwareConnectionProvider  
    MeasuringInputHardwareConnectionProvider connectionProvider =  
    technologyObject.GetService<MeasuringInputHardwareConnectionProvider>();  
  
    //Connect technology object with Channel  
    connectionProvider.Connect(channel);  
  
    //Check ConnectionState of technology object  
    bool connectionState = connectionProvider.IsConnected;  
}
```

Gleichlaufachse mit Leitwerten verschalten

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Eine S7-1500 CPU ist im Projekt angelegt.
- Ein Technologieobjekt des Typs TO_PositioningAxis, TO_SynchronousAxis oder TO_ExternalEncoder als Leitachse ist im Projekt angelegt.
- Ein Technologieobjekt des Typs TO_SynchronousAxis als Folgeachse ist im Projekt angelegt.

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Anwendung

Um ein Technologieobjekt Gleichlaufachse mit Leitwerten zu verschalten, ist es erforderlich, mehrere Werte zusammen in einem einzigen Aufruf anzugeben. Der Typ der Public API SynchronousAxisMasterValues bietet die folgenden Methoden, die zur Verschaltung oder Trennung von Leitwerten verwendet werden können. Leitwerte können als Sollwertkopplungen (S7-1500 CPU, S7-1500T CPU) oder Istwertkopplungen (S7-1500T CPU) verschaltet werden. Alle Methoden und Attribute sind für beide Arten von Kopplungen relevant.

Methode	Beschreibung
int IndexOf (TechnologicalInstanceDB element)	Gibt den entsprechenden Index eines Leitwertes zurück
bool Contains (TechnologicalInstanceDB element)	TRUE: Der Behälter enthält den Leitwert FALSE: Der Behälter enthält nicht den Leitwert
IEnumerator GetEnumerator <TechnologicalInstanceDB>()	Dient zur Unterstützung jeder Iteration
void Add (TechnologicalInstanceDB element)	Verschaltet die Folgeachse mit einem Leitwert
bool Remove (TechnologicalInstanceDB element)	Trennt die Folgeachse vom Leitwert TRUE: Verschaltung wurde erfolgreich getrennt FALSE: Verschaltung konnte nicht getrennt werden

Sie können die folgenden schreibgeschützten Attribute verwenden:

Attribut	Datentyp	Beschreibung
Count	int	Anzahl der Leitwerte
IsReadonly	bool	TRUE: Der Behälter ist schreibgeschützt FALSE: Der Behälter ist nicht schreibgeschützt
Parent	IEngineeringObject	Gibt die übergeordneten Werte des Behälters zurück. In diesem Fall ist der übergeordnete Wert der Service SynchronousAxisMasterValues.
this [id] { get; }	TechnologicalInstanceDB	Indexbasierter Zugriff auf Leitwerte

Programmcode: Gleichlaufachse mit einem Leitwert verschalten

Ändern Sie den folgenden Programmcode, um eine Gleichlaufachse mit einem Leitwert zu verschalten:

```
//An instance of leading axis and following axis is already available in the program before
private static void UseServiceSynchronousAxisMasterValues(TechnologicalInstanceDB
masterTechnologyObject, TechnologicalInstanceDB synchronousTechnologyObject)
{
    //Retrieve service SynchronousAxisMasterValues
    SynchronousAxisMasterValues masterValues =
    synchronousTechnologyObject.GetService<SynchronousAxisMasterValues>();

    //Connect following axis and leading axis with setpoint coupling
    masterValues.SetPointCoupling.Add(masterTechnologyObject);

    //Get container of connected leading axis with setpoint coupling
    TechnologicalInstanceDBAssociation setPointMasterValues =
    masterValues.SetPointCoupling;

    //Remove connected leading axis with setpoint coupling
    masterValues.SetPointCoupling.Remove(masterTechnologyObject);

    //Connect following axis and leading axis with actual value coupling
    masterValues.ActualValueCoupling.Add(masterTechnologyObject);

    //Get container of connected leading axis with actual value coupling
    TechnologicalInstanceDBAssociation actualValueMasterValues =
    masterValues.ActualValueCoupling;

    //Remove connected leading axis with actual value coupling
    masterValues.ActualValueCoupling.Remove(masterTechnologyObject);
}
```

Exportieren und Importieren des Technologieobjekts Cam (S7-1500T)

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79).
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111).
- Eine S7-1500 CPU ist im Projekt angelegt.
Siehe PLC-Target und HMI-Target abfragen (Seite 185)
- Das Technologieobjekt liegt vor.

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Anwendung

Um die Daten eines Technologieobjekts Cam zu exportieren oder zu importieren, müssen Sie das Format und das gewünschte Trennzeichen angeben. Der Typ der Public API CamDataSupport bietet die folgenden Methoden, die zur Exportieren der Daten des Technologieobjekts Cam verwendet werden können:

Methode	Beschreibung
void SaveCamDataBinary(System.IO.FileInfo destinationFile)	Exportiert die Daten im Binärformat in die Zielfile.
void SaveCamDataPointList(System.IO.FileInfo destinationFile, CamDataFormatSeparator separator, int samplePoints)	Exportiert die Daten im Format "PointList" in die Zielfile.
void SaveCamData(System.IO.FileInfo destinationFile, CamDataFormat format, CamDataFormatSeparator separator)	Exportiert die Daten in die Zielfile. Sie können das Datenformat als "MCD", "SCOUT" oder "Pointlist" und als Trennzeichen "Tabulator" oder "Komma" angeben. Wenn Sie "PointList" wählen, werden 360 Interpolationspunkte exportiert.
void LoadCamData(System.IO.FileInfo sourceFile, CamDataFormatSeparator separator)	Importiert die Cam-Daten im Format "MCD", "SCOUT" oder "Pointlist" in das Projekt.
void LoadCamDataBinary(System.IO.FileInfo sourceFile)	Importiert die Cam-Daten aus einer Binärdatei in das Projekt.

Die folgenden Attribute können Sie verwenden:

Attribut	Datentyp	Beschreibung
separator	CamDataFormatSeparator	Zulässige Werte <ul style="list-style-type: none"> • Tabulator • Komma
samplePoints	int	Anzahl zu exportierender Interpolationspunkte.
format	CamDataFormat	Zulässige Werte <ul style="list-style-type: none"> • MCD • SCOUT • Pointlist
destinationFile	System.IO.FileInfo	Name der Zielfile. Darf nicht null sein. Zugriffsrechte und ausreichend Speicherplatz müssen auf dem Speichermedium gegeben sein. Eine vorhandene Datei wird überschrieben.
sourceFile	System.IO.FileInfo	Name der Quelldatei. Darf nicht null sein. Zugriffsrechte müssen gegeben sein. Der Inhalt muss im angegebenen Format vorliegen.

Programmcode: Cam-Daten exportieren

Ändern Sie folgenden Programmcode, um Cam-Daten zu exportieren:

```
//An instance of technology object is already available in the program before
private static void ExportCamData(TechnologicalInstanceDB technologyObject,
System.IO.FileInfo destinationFile)
{
    //Retrieve service CamDataSupport
    CamDataSupport camData = technologyObject.GetService<CamDataSupport>();

    //Save cam data in MCD format, using the separator Tab
    camData.SaveCamData(destinationFile, CamDataFormat.MCD, CamDataFormatSeparator.Tab);
}
```

Programmcode: Cam-Daten importieren

Ändern Sie folgenden Programmcode, um Cam-Daten zu importieren:

```
//An instance of technology object is already available in the program before
private static void ImportCamData(TechnologicalInstanceDB technologyObject,
System.IO.FileInfo sourceFile)
{
    //Retrieve service CamDataSupport
    CamDataSupport camData = technologyObject.GetService<CamDataSupport>();

    //Load cam data from source file, using the separator Tab
    camData.LoadCamData(sourceFile, CamDataFormatSeparator.Tab);
}
```

7.19.23.16 PID-Regelung

Parameter für PID_Compact, PID_3Step, PID_Temp, CONT_C, CONT_S, TCONT_CP und TCONT_S

Eine Liste sämtlicher verfügbarer Parameter finden Sie in der Produktinformation "Parameter von Technologieobjekten in TIA Portal Openness" im Internet (<https://support.industry.siemens.com/cs/ww/de/view/109744932>).

Für jeden Parameter werden die folgenden Eigenschaften angegeben:

- Name in der Konfiguration (TIA Portal)
- Name in Openness
- Datentyp in Openness
- Standardzugriff
- Wertebereich

Hinweis

Im TIA Portal finden Sie in der Parametersicht der Technologieobjektkonfiguration die Spalte "Name in Openness".

Weitere Informationen

Weitere Informationen finden Sie im Funktionshandbuch SIMATIC S7-1200/S7-1500 PID control im Internet (<https://support.industry.siemens.com/cs/ww/de/view/108210036>).

7.19.23.17 Zählen

Parameter für High_Speed_Counter und SSI_Absolute_Encoder

Eine Liste sämtlicher verfügbarer Parameter finden Sie in der Produktinformation "Parameter von Technologieobjekten in TIA Portal Openness" im Internet (<https://support.industry.siemens.com/cs/ww/de/view/109744932>).

Für jeden Parameter werden die folgenden Eigenschaften angegeben:

- Name in der Konfiguration (TIA Portal)
- Name in Openness
- Datentyp in Openness
- Standardzugriff
- Wertebereich

Weitere Informationen

Weitere Informationen finden Sie im Funktionshandbuch SIMATIC S7-1500, ET 200MP, ET 200SP Zählen, Messen und Positionserfassung im Internet (<http://support.automation.siemens.com/WW/view/de/59709820>).

7.19.23.18 Easy Motion Control

Parameter für AXIS_REF

Eine Liste sämtlicher verfügbarer Parameter finden Sie in der Produktinformation "Parameter von Technologieobjekten in TIA Portal Openness" im Internet (<https://support.industry.siemens.com/cs/ww/de/view/109744932>).

Für jeden Parameter werden die folgenden Eigenschaften angegeben:

- Name in der Konfiguration (TIA Portal)
- Name in Openness
- Datentyp in Openness

- Standardzugriff
- Wertebereich

Hinweis

Im TIA Portal finden Sie in der Parametersicht der Technologieobjektkonfiguration die Spalte "Name in Openness".

Weitere Informationen

Weitere Informationen zu Easy Motion Control finden Sie im Informationssystem von STEP 7 (TIA Portal).

7.19.24 Variablen und Variablenklassen**7.19.24.1 Starten des PLC-Variableneeditors****Voraussetzung**

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Eine Instanz des TIA Portals ist mit Benutzeroberfläche geöffnet.

Programmcode

Ändern Sie den folgenden Programmcode, um den entsprechenden Editor für eine Objektreferenz des Typs `PlcTagTable` in der Instanz des TIA Portals zu starten:

```
//Öffnet Variablenabelle im Editor "Tags"  
private static void OpenTagtableInEditor(PlcSoftware plcSoftware)  
{  
    PlcTagTable plcTagTable = plcSoftware.TagTableGroup.TagTables.Find("MyTagTable");  
    plcTagTable.ShowInEditor();  
}
```

Siehe auch

Import von Projektierungsdaten (Seite 780)

7.19.24.2 Systemgruppen für PLC-Variablen abfragen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Eine PlcSoftware-Instanz wurde von einem PLC-Geräteelement abgerufen.
Siehe PLC-Target und HMI-Target abfragen (Seite 185)

Programmcode

Ändern Sie den folgenden Programmcode, um eine Systemgruppe für PLC-Variablen abzufragen:

```
//Retrieves the plc tag table group from a plc
private PlcTagTableSystemGroup GetControllerTagfolder(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    return plcTagTableSystemGroup;
}
```

7.19.24.3 PLC-Variablenabelle anlegen

Voraussetzungen

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Eine PlcSoftware-Instanz wurde von einem PLC-Geräteelement abgerufen.
Siehe PLC-Target und HMI-Target abfragen (Seite 185)

Programmcode

Um die PLC-Variablenabelle anzulegen, ändern Sie folgenden Programmcode: Es wird eine neue Variablenabelle mit dem angegebenen Namen in der Zusammensetzung angelegt.

```
PlcTagTable myTable = plc.TagTableGroup.TagTables.Create("myTable");
```

Siehe auch

PLC-Target und HMI-Target abfragen (Seite 185)

7.19.24.4 Benutzerdefinierte Gruppen für PLC-Variablen enumerieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Eine PlcSoftware-Instanz wurde von einem PLC-Geräteelement abgerufen.
Siehe PLC-Target und HMI-Target abfragen (Seite 185)

Verwendung

Enthaltene Unterordner werden beim Enumerieren rekursiv berücksichtigt.

Programmcode: Benutzerdefinierte Gruppen für PLC-Variablen enumerieren

Ändern Sie den folgenden Programmcode, um benutzerdefinierte Gruppen für PLC-Variablen zu enumerieren:

```
//Enumerates all plc tag table user groups including subgroups
private static void EnumeratePlcTagTableUserGroups(PlcSoftware plcSoftware)
{
    foreach (PlcTagTableUserGroup plcTagTableUsergroup in plcSoftware.TagTableGroup.Groups)
    {
        EnumerateTagTableUserGroups(plcTagTableUsergroup);
    }
}
private static void EnumerateTagTableUserGroups(PlcTagTableUserGroup tagTableUsergroup)
{
    foreach (PlcTagTableUserGroup plcTagTableUsergroup in tagTableUsergroup.Groups)
    {
        EnumerateTagTableUserGroups(plcTagTableUsergroup);
        // recursion
    }
}
```

Programmcode: Auf eine benutzerdefinierte Gruppe zugreifen

Ändern Sie den folgenden Programmcode, um auf eine benutzerdefinierte Gruppe für PLC-Variablen zuzugreifen:

```
//Gives individual access to a specific plc tag table user folder
private static void AccessPlcTagTableUserGroupWithFind(PlcSoftware plcSoftware, string
folderToFind)
{
    PlcTagTableUserGroupComposition plcTagTableUserGroupComposition =
plcSoftware.TagTableGroup.Groups;
    PlcTagTableUserGroup controllerTagUserFolder =
plcTagTableUserGroupComposition.Find(folderToFind);
    // The parameter specifies the name of the user folder
}
```

7.19.24.5 Benutzerdefinierte Gruppen für PLC-Variablen erzeugen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Die Schnittstelle TIA Portal Openness API unterstützt die Erzeugung einer benutzerdefinierten Gruppe für PLC-Variablen.

Programmcode

Ändern Sie den folgenden Programmcode, um eine benutzerdefinierte Gruppe für PLC-Variablen zu erzeugen:

```
//Creates a plc tag table user group
private static void CreatePlcTagTableUserGroup(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup systemGroup = plcSoftware.TagTableGroup;
    PlcTagTableUserGroupComposition groupComposition = systemGroup.Groups;
    PlcTagTableUserGroup myCreatedGroup = groupComposition.Create("MySubGroupName");
    // Optional;
    // create a subgroup
    PlcTagTableUserGroup mySubCreatedGroup =
myCreatedGroup.Groups.Create("MySubSubGroupName");
}
```

7.19.24.6 Benutzerdefinierte Gruppen für PLC-Variablen löschen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Die Schnittstelle TIA Portal Openness API unterstützt das Löschen einer bestimmten benutzerdefinierten Gruppe für PLC-Variablen tabellen.

Programmcode

Ändern Sie den folgenden Programmcode, um eine bestimmte benutzerdefinierte Gruppe für PLC-Variablen tabellen zu löschen:

```
private static void DeletePlcTagTableUserGroup(PlcSoftware plcSoftware)
{
    PlcTagTableUserGroup group = plcSoftware.TagTableGroup.Groups.Find("MySubGroupName");
    if (group != null)
    {
        group.Delete();
    }
}
```

7.19.24.7 PLC-Variablen tabellen in einem Ordner enumerieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode: PLC-Variablenklassen enumerieren

Ändern Sie den folgenden Programmcode, um alle PLC-Variablenklassen in Systemgruppen oder benutzerdefinierten Gruppen zu enumerieren:

```
//Enumerates all plc tag tables in a specific system group or and user group
private static void EnumerateAllPlcTagTablesInFolder(PlcSoftware plcSoftware)
{
    PlcTagTableComposition tagTables = plcSoftware.TagTableGroup.TagTables;
    // alternatively, PlcTagTableComposition tagTables =
plcSoftware.TagTableGroup.Groups.Find("UserGroup XYZ").TagTables;
    foreach (PlcTagTable tagTable in tagTables)
    {
        // add code here
    }
}
```

Programmcode: Auf PLC-Variablenklassen zugreifen

Um auf die PLC-Variablenklasse zuzugreifen, ändern Sie folgenden Programmcode:

```
//Gives individual access to a specific Plc tag table
private static void AccessToPlcTagTableWithFind(PlcSoftware plcSoftware)
{
    PlcTagTableComposition tagTables = plcSoftware.TagTableGroup.TagTables;
    // alternatively, PlcTagTableComposition tagTables =
plcSoftware.TagTableGroup.Groups.Find("UserGroup XYZ").TagTables;
    PlcTagTable controllerTagTable = tagTables.Find("Tag table XYZ");
    // The parameter specifies the name of the tag table
}
```

7.19.24.8 Informationen einer PLC-Variablenklasse abfragen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Über PLC-Variablenklassen können Sie auf Anwenderkonstanten, Systemkonstanten und Variablen zugreifen. Die Anzahl der Variablenzusammensetzung einer Variablenklasse entspricht der Anzahl der Variablen in der Variablenklasse. Die PLCTagTable enthält die folgenden Navigatoren, Attribute und Aktionen.

In der PLC-Variablenklasse wird auf die folgenden Attribute zugegriffen.

Name	Typ	Typ
IsDefault	Bool	Schreibgeschützt
ModifiedTimeStamp	DateTime	Schreibgeschützt
Name	String	Schreibgeschützt

Die PLC-Variablenliste enthält die nachstehend aufgeführten Aktionen.

Name	Rückgabetyp	Beschreibung
Delete	leer	Löscht die Instanz. Löst eine Ausnahme aus, wenn IsDefault wahr ist.
Exportieren	leer	Exportiert SIMATIC ML einer PLC-Variablenliste.
ShowInEditor	leer	Zeigt die Variablenliste im PLC-Variablenlisteneditor an.

Programmcode

Um die Informationen einer PLC-Variablenliste abzufragen, ändern Sie den folgenden Programmcode:

```
private static void AccessPlcConstantsUsingFind(PlcTagTable tagTable)
{
    PlcUserConstantComposition plcUserConstants = tagTable.UserConstants;
    PlcUserConstant plcUserConstant = plcUserConstants.Find("Constant XYZ");
    //PlcSystemConstantComposition plcSystemConstants = tagTable.SystemConstants;
    //PlcSystemConstant plcSystemConstant = plcSystemConstants.Find("Constant XYZ");
}
private static void EnumeratePlcTags(PlcTagTable tagTable)
{
    PlcTagComposition plcTags = tagTable.Tags;
    foreach (PlcTag plcTag in plcTags)
    {
        string name = plcTag.Name; string typeName = plcTag.DataTypeName;
        string logicalAddress = plcTag.LogicalAddress;
    }
}
private static void EnumeratePlcTagsUsingFind(PlcTagTable tagTable)
{
    PlcTagComposition plcTags = tagTable.Tags;
    PlcTag plcTag = plcTags.Find("Constant XYZ");
}
```

7.19.24.9 Zeitpunkt der letzten Änderung einer PLC-Variablenliste lesen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Das Format des Zeitstempels ist UTC.

Programmcode

Um den Zeitstempel einer bestimmten PLC-Variablenliste zu lesen, ändern Sie den folgenden Programmcode:

```
//Reads Time-Stamp of a plc Tag Table
private static void GetLastModificationDateOfTagtable(PlcSoftware plcSoftware)
{
    PlcTagTable plcTagTable = plcSoftware.TagTableGroup.TagTables.Find("MyTagTable");
    DateTime modifiedTagTableTimeStamp = plcTagTable.ModifiedTimeStamp;
}
```

7.19.24.10 PLC-Variablenliste aus einer Gruppe löschen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode

Ändern Sie den folgenden Programmcode, um eine bestimmte Variablenliste aus einer Gruppe zu löschen:

```
//Deletes a PlcTagTable of a group
private static void DeletePlcTagTableInAGroup(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup group = plcSoftware.TagTableGroup;
    PlcTagTable tagtable = group.TagTables.Find("MyTagTable");
    if (tagtable!= null)
    {
        tagtable.Delete();
    }
}
```

7.19.24.11 PLC-Variablen enumerieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Programmcode: PLC-Variablen in Variablenlisten enumerieren

Um alle PLC-Variablen in einer Variablenliste zu enumerieren, ändern Sie den folgenden Programmcode:

```
//Enumerates all plc tags in a specific tag table
private static void EnumerateAllPlcTagsInTagTable(PlcSoftware plcSoftware)
{
    PlcTagTable tagTable = plcSoftware.TagTableGroup.TagTables.Find("Tagtable XYZ");
    foreach (PlcTag tag in tagTable.Tags)
    {
        // add code here
    }
}
```

7.19.24.12 Auf PLC-Variablen zugreifen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Der Typ `PlcTagComposition` stellt eine Sammlung von PLC-Variablen dar.

Programmcode: Auf eine bestimmte PLC-Variable zugreifen

Um auf die gewünschte PLC-Variable zuzugreifen, ändern Sie folgenden Programmcode. Sie haben Zugriff auf die folgenden Attribute:

- Name (nur Lesezugriff)
- Name des Datentyps
- Logische Adresse
- Kommentar
- ExternalAccessible
- ExternalVisible
- ExternalWritable

```
//Gives individual access to a specific plc tag
private static void AccessPlcTag(PlcTagTable tagTable)
{
    PlcTag tag = tagTable.Tags.Find("Tag XYZ");
    // The parameter specifies the name of the tag
}
```

Programmcode: Variablen anlegen

Ändern Sie folgenden Programmcode:

```
private static void CreateTagInPLCTagtable(PlcSoftware plcsoftware)
// Create a tag in a tag table with default attributes
{
    string tagName = "MyTag";
    PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
    PlcTagComposition tagComposition = table.Tags;
    PlcTag tag = tagComposition.Create(tagName);
}
```

Ändern Sie folgenden Programmcode:

```
private static void CreateTagInPLCTagtable(PlcSoftware plcsoftware)
// Create a tag of data type bool and logical address not set
{
    string tagName = "MyTag";
    string dataType = "Bool";
    string logicalAddress = "";
    PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
    PlcTagComposition tagComposition = table.Tags;
    PlcTag tag = tagComposition.Create(tagName, dataType, logicalAddress);
}
```

Programmcode: Variablen löschen

Ändern Sie folgenden Programmcode:

```
private static void DeleteTagFromPLCTagtable(PlcSoftware plcsoftware)
// Deletes a single tag of a tag table
{
    string tagName = "MyTag";
    PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
    PlcTagComposition tagComposition = table.Tags;
    PlcTag tag = tagComposition.Find(tagName);
    if (tag != null)
    {
        tag.Delete();
    }
}
```

7.19.24.13 Auf PLC-Konstanten zugreifen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Der Typ `PlcUserConstantComposition` stellt eine Sammlung von PLC-Benutzerkonstanten dar. Sie haben Zugriff auf die folgenden Attribute:

- Name (nur Lesezugriff)
- Name des Datentyps
- Wert

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Der Typ `PlcSystemConstantComposition` stellt eine Sammlung von PLC-Systemkonstanten dar. Sie haben Zugriff auf die folgenden Attribute:

- `Name` (nur Lesezugriff)
- `Datentypname` (nur Lesezugriff)
- `Wert` (nur Lesezugriff)

Programmcode: Benutzerkonstanten anlegen

Ändern Sie folgenden Programmcode:

```
private static void CreateUserConstantInPLCTagtable(PlcSoftware plcsoftware)
// Create a user constant in a tag table
{
    string constantName = "MyConstant";
    PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
    PlcUserConstantComposition userConstantComposition = table.UserConstants;
    PlcUserConstant userConstant = userConstantComposition.Create(constantName);
}
```

Programmcode: Benutzerkonstanten löschen

Ändern Sie folgenden Programmcode:

```
private static void DeleteUserConstantFromPLCTagtable(PlcSoftware plcsoftware)
// Deletes a single user constant of a tag table
{
    PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
    PlcUserConstantComposition userConstantComposition = table.UserConstants;
    PlcUserConstant userConstant = userConstantComposition.Find("MyConstant");
    if (userConstant != null)
    {
        userConstant.Delete();
    }
}
```

Programmcode: Auf Systemkonstanten zugreifen

Ändern Sie folgenden Programmcode:

```
//Gives individual access to a specific system constant
private static void AccessSystemConstant(PlcTagTable tagTable)
{
    PlcTag systemConstant = tagTable.SystemConstants.Find("Constant XYZ");
    // The parameter specifies the name of the tag
}
```

Siehe auch

- [Benutzerdefinierte Gruppen für PLC-Variablen erzeugen \(Seite 586\)](#)
- [Benutzerdefinierte Gruppen für PLC-Variablen löschen \(Seite 587\)](#)
- [PLC-Variablenliste aus einer Gruppe löschen \(Seite 590\)](#)
- [Auf PLC-Variablen zugreifen \(Seite 592\)](#)
- [Starten des PLC-Variableneeditors \(Seite 583\)](#)
- [Zeitpunkt der letzten Änderung einer PLC-Variablenliste lesen \(Seite 590\)](#)

7.19.25 Funktionen für Software-Einheiten

7.19.25.1 Mit Software-Einheiten arbeiten

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Die Einheiten sind wichtige Bestandteile der PLC-Programmierung und in der PlcSoftware zu finden.

Der unit provider wird über GetService aus der PlcSoftware abgerufen, und daraus kann wiederum die PlcUnitComposition abgerufen werden.

Mit Einheiten können Sie die folgenden grundlegenden Aktionen durchführen, während Sie TIA Portal verwenden:

- Einheiten erstellen
- Einheiten löschen
- Einheiten umbenennen

Programmcode: Einheiten erstellen

Ändern Sie folgenden Programmcode, um Einheiten zu erstellen.

```
PlcUnitProvider provider = m_Target.GetService<PlcUnitProvider>();  
PlcUnitComposition unitComposition = provider.UnitGroup.Units;  
//Creating Unit  
PlcUnit unit1 = unitComposition.Create("Unit1");
```

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Wenn Sie den Namen falsch eingeben, z. B. mit einem Leerzeichen am Anfang oder mit ungültigen Zeichen, oder wenn der Name zu lang ist, wird eine Fehlermeldung ausgegeben: Eine Recoverable Exception wird mit folgender Fehlermeldung ausgelöst: "Der Wert von Attribut 'Name' enthält ein ungültiges Zeichen an Position 0."

```
try
{
PlcUnit unit1 = unitComposition.Create("Unit1");
}
catch (EngineeringTargetInvocationException e)
{
Console.WriteLine(e.Message);
}
```

Wenn Sie versuchen, eine Einheit mit einem Namen zu erstellen, der bereits verwendet wird, wird eine Fehlermeldung ausgegeben. Eine Recoverable Exception wird mit folgender Meldung ausgelöst: "Einheit mit dem Namen 'Einheit1' ist bereits vorhanden."

```
PlcUnit unit1 = unitComposition.Create("Unit1");
...
...
try
{
PlcUnit unit2 = unitComposition.Create("Unit1");
}
catch (EngineeringTargetInvocationException e)
{
Console.WriteLine(e.Message);
}
```

Programmcode: Einheiten löschen

Ändern Sie folgenden Programmcode, um Einheiten zu löschen:

```
unitProvider.UnitGroup.Units.Find("Unit_2").Delete();
```

Es wird eine NullReferenceException ausgegeben, wenn die Methode Find() aus dem obigen Beispiel keinen Verweis auf eine Einheit bereitstellt.

Programmcode: Einheiten umbenennen

Sie können die Attribute auf mehrere Arten festlegen. Die Festlegung kann durch Zuordnung eines Werts oder durch Aufruf von SetAttribute oder SetAttributes erfolgen. Die Eigenschaft Name kann durch GetAttribute oder GetAttributes validiert werden. Damit Sie SetAttribute oder GetAttribute verwenden können, muss das Objekt vorher an IEngineeringObject übertragen werden.

Ändern Sie folgenden Programmcode, um Einheiten umzubenennen.

```
//Set Value
PlcUnit unit1 = unitComposition.Create("Unit1");
unit1.Name = "Unit1_new";
//Using SetAttributes():
PlcUnit unit1 = unitComposition.Create("Unit1");
var attrList = new List<KeyValuePair<string, object>>()
{
new KeyValuePair<string, object>("Name", "Unit1_new")
};
unit1.SetAttributes(attrList);
//Using SetAttribute()
PlcUnit unit2 = unitComposition.Create("Unit2");
IEngineeringObject unit = (IEngineeringObject)unit2;
unit.SetAttribute("Name", "Unit2_new");
```

Wenn Sie den Namen falsch eingeben, z. B. mit einem Leerzeichen am Anfang oder mit ungültigen Zeichen, wird eine Fehlermeldung ausgegeben. Eine Recoverable Exception wird mit folgender Fehlermeldung ausgelöst: "Der Wert von Attribut 'Name' enthält ein ungültiges Zeichen an Position 0."

```
PlcUnit unit1 = unitComposition.Create("Unit_1");
try
{
unit1.Name = "Unit_1";
}
catch (EngineeringTargetException e)
{
Console.WriteLine(e.Message);
}
```

Wenn Sie versuchen, eine Einheit in einen Namen umzubenennen, der bereits verwendet wird, wird eine Fehlermeldung ausgegeben. Eine Recoverable Exception wird mit folgender Fehlermeldung ausgelöst: "Die Eigenschaft 'Name' hat einen ungültigen Wert." 'Einheit1'. Eine Software-Einheit mit demselben Namen ist bereits vorhanden."

```
PlcUnit unit1 = unitComposition.Create("Unit1");
PlcUnit unit2 = unitComposition.Create("Unit_2");
try
{
PlcUnit unit2 = unitComposition.Create("Unit1");
}
catch (EngineeringTargetException e)
{
Console.WriteLine(e.Message);
}
```

Siehe auch

[Projekt öffnen \(Seite 111\)](#)

7.19.25.2 Auf Software-Einheiten zugreifen

Voraussetzung

- Die Anwendung TIA Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Sie können mit TIA Portal Openness auf die Einheiten zugreifen. Die Einheiten sind wichtige Bestandteile der PLC-Programmierung. Einheiten werden nur von bestimmten PLCs unterstützt. Deshalb werden die Einheiten im Objektmodell nicht statisch modelliert, sodass sie im PLC-Softwarecontainer nicht direkt auffindbar sind.

Mit Hilfe des PlcUnitProvider ist jedoch der Zugriff auf die Einheiten möglich. Der Provider ist in der PLC-Software über GetService() verfügbar, wie im folgenden Beispiel-Programmcode dargestellt:

Der PlcUnitProvider ermöglicht den Zugriff über den UnitGroup navigator auf die PlcUnitSystemGroup, welche die PlcUnitComposition mit den spezifischen Möglichkeiten der allgemeinen Zusammensetzung enthält (Iterieren, Finden, Erstellen, Importieren usw. in Bezug auf PlcUnits).

Die Darstellung der Einheit (PlcUnit) enthält die eigenen spezifischen Eigenschaften der Einheit als Openness-Attribute.

Sie können auf die folgenden Attribute und Methoden für Eigenschaften von Openness-Einheiten zugreifen:

Attributname	Typ
Author	String
Name	String

Methodename	Rückgabetyp
Export	void
Delete	void
GetAttributes	IList<object>
SetAttributes	void

Unter der PlcUnit können Sie nacheinander die folgenden Objekte durchlaufen:

- Kommentar: MultilingualText
- die BlockGroup (PlcBlockSystemGroup)
 - unit.BlockGroup

- TagTableGroup (PlcTagTableSystemGroup)
 - unit.TagTableGroup
- TypeGroup (PlcTypeSystemGroup)
 - unit.TypeGroup

Programmcode: Auf Einheiten zugreifen

```
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);  
PlcUnitProvider unitProvider = plcTarget.GetService<PlcUnitProvider>();  
PlcUnit unit = unitProvider.UnitGroup.Units.Find("Unit_2");
```

Siehe auch

- Verbindung zum TIA Portal aufbauen (Seite 79)
- Projekt öffnen (Seite 111)

7.19.25.3 Auf Software-Einheiten zugrundeliegende Objekte zugreifen

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Sie können mit TIA Portal Openness zugreifen:

- rekursiv auf die Systemgruppe "Programmbausteine", die darin enthaltenen Gruppen und Bausteine
- rekursiv auf die Systemgruppe "PLC-Datentypen", die darin enthaltenen Gruppen und Datentypen
- rekursiv auf die Systemgruppe "PLC-Variablen", die darin enthaltenen Gruppen und Variablenlisten

Programmcode: Auf Programmsteine und Bausteingruppen zugreifen

Um auf den Programmsteine unter der aktuellen PLC-Einheit zuzugreifen, ändern Sie folgenden Programmcode, indem Sie die Zusammenstellung der Programmsteine abrufen und die darin enthaltenen Bausteine durchiterieren:

```
PlcBlockComposition blockComposition = m_PlcUnit.BlockGroup.Blocks;
foreach (PlcBlock block in blockComposition)
{
...
//usage of block
...
}
```

Um auf die Programmsteingruppe unter der aktuellen PLC-Einheit zuzugreifen, ändern Sie folgenden Programmcode, indem Sie die Zusammenstellung der Programmsteingruppen abrufen und die darin enthaltenen Gruppen durchiterieren:

```
PlcBlockUserGroupComposition usergroupComposition = m_PlcUnit.BlockGroup.Groups;
foreach (PlcBlockUserGroup group in usergroupComposition)
{
    PlcBlockComposition blockComposition = group.Blocks;
    foreach (PlcBlock block in blockComposition)
    {
...
//usage of the block
...
    }
}
}
```

Programmcode: Auf PLC-Datentypen und Datentypgruppen zugreifen

Um auf die PLC-Datentypen unter der aktuellen PLC-Einheit zuzugreifen, ändern Sie folgenden Programmcode, indem Sie die Zusammenstellung der PLC-Datentypen abrufen und die darin enthaltenen Datentypen durchiterieren:

```
PlcTypeComposition typeComposition = m_PlcUnit.TypeGroup.Types;
foreach (PlcType type in typeComposition)
{
...
// usage of the type
...
}
```

Um auf die PLC-Datentypgruppe unter der aktuellen PLC-Einheit zuzugreifen, ändern Sie folgenden Programmcode, indem Sie die Zusammenstellung der PLC-Datentypen abrufen und die darin enthaltenen Gruppen durchiterieren:

```
PlcTypeUserGroupComposition usergroupComposition = m_PlcUnit.TypeGroup.Groups;
foreach (PlcTypeUserGroup group in usergroupComposition)
{
    PlcTypeComposition typeComposition = group.Types;
    foreach (PlcType type in typeComposition)
    {
        ...
        // usage of the type
        ...
    }
}
```

Programmcode: Auf PLC-Variablenklassen und Variablenklassengruppen zugreifen

Um auf die PLC-Variablenklassen unter der aktuellen PLC-Einheit zuzugreifen, ändern Sie folgenden Programmcode, indem Sie die Zusammenstellung der PLC-Variablenklassen abrufen und die darin enthaltenen Variablenklassen durchiterieren:

```
PlcTagTableComposition tagtableComposition = m_PlcUnit.TagTableGroup.TagTables;
foreach (PlcTagTable type in tagtableComposition)
{
    ...
    // usage of the tag table
    ...
}
```

Um auf die PLC-Variablenklassengruppen unter der aktuellen PLC-Einheit zuzugreifen, ändern Sie folgenden Programmcode, indem Sie die Zusammenstellung der PLC-Variablenklassengruppen abrufen und die darin enthaltenen Gruppen durchiterieren:

```
PlcTagTableUserGroupComposition usergroupComposition = m_PlcUnit.TagTableGroup.Groups;
foreach (PlcTagTableUserGroup group in usergroupComposition)
{
    PlcTagTableComposition tagtableComposition = group.TagTables;
    foreach (PlcTagTable type in tagtableComposition)
    {
        ...
        // usage of the tag table
        ...
    }
}
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.19.25.4 Auf vorhandene Beziehungen einer Einheit zugreifen

Voraussetzung

- Die Anwendung TIA Portal Openess ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Sie können mit TIA Portal Openess auf alle vorhandenen Beziehungen einer Einheit zugreifen und so die Eigenschaften dieser Beziehungen lesen.

Die folgenden Attribute werden von der Einheitsbeziehung in Openess unterstützt:

Attributname	Typ	Beschreibung
RelationType	UnitRelationType	Art der Beziehung
RelatedObject	String	Enthält den Namen des zugänglichen Elements

Die folgenden ENUM-Werte werden für das Attribut RelationType bereitgestellt:

- Software Unit
- Non-Unit DB
- TO DB

Programmcode: Auf Beziehungen zugreifen

Beziehungen, die Beziehungsarten und die Namen der bezogenen Objekte können auf mehrere Arten abgerufen werden.

Ändern Sie folgenden Programmcode, um die Zusammensetzung einer Beziehung abzurufen:

```
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices,
PLCName);
PlcUnitProvider provider = plcTarget.GetService<PlcUnitProvider>();
//assuming existing units
m_PlcUnit = provider.UnitGroup.Units[0];
PlcUnitRelationComposition unitRelations = m_PlcUnit.Relations;
```

Ändern Sie folgenden Programmcode, um Beziehungen nach Indexer abzurufen:

```
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);
PlcUnitProvider provider = plcTarget.GetService<PlcUnitProvider>();
m_PlcUnit = provider.UnitGroup.Units[0]; //assuming existing units
PlcUnitRelation unitRelation = m_PlcUnit.Relations[1];
```

Ändern Sie folgenden Programmcode, um Beziehungen durch Iteration abzurufen:

```
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);
PlcUnitProvider provider = plcTarget.GetService<PlcUnitProvider>();
//assuming existing units
m_PlcUnit = provider.UnitGroup.Units[0];
PlcUnitRelationComposition unitRelations = m_PlcUnit.Relations;
foreach (PlcUnitRelation relation in unitRelations)
{
// using 'relation'
}
```

Ändern Sie folgenden Programmcode, um den Beziehungstyp einer Einheit durch Attribut RelationType abzurufen:

```
PlcSoftware plcTarget =
GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);
PlcUnitProvider provider = plcTarget.GetService<PlcUnitProvider>();
//assuming existing units
m_PlcUnit = provider.UnitGroup.Units[0];
UnitRelationType unitRelationType = m_PlcUnit.Relations[2].RelationType;
```

Ändern Sie folgenden Programmcode, um den Namen des bezogenen Objekts einer Einheit durch das Attribut RelatedObject abzurufen:

```
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);
PlcUnitProvider provider = plcTarget.GetService<PlcUnitProvider>();
//assuming existing units
m_PlcUnit = provider.UnitGroup.Units[0];
string unitRelatedObjectName = m_PlcUnit.Relations[1].RelatedObject;
```

Ändern Sie folgenden Programmcode, um eine Beziehung mit Hilfe von "Suchen" mit dem Namen des zugänglichen Elements (RelatedObject) zu finden:

```
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);
PlcUnitProvider provider = plcTarget.GetService<PlcUnitProvider>();
//assuming existing units
m_PlcUnit = provider.UnitGroup.Units[0];
PlcUnitRelation relation = m_PlcUnit.Relations.Find(unitRelatedObjectName);
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

7.19.25.5 Eigenschaften von Software-Einheiten aktualisieren

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Sie können mit TIA Portal Openness Einheiten-Eigenschaften wie z. B. Autor und Kommentar aktualisieren.

Programmcode: Eigenschaft "Author" der Einheit aktualisieren

Ändern Sie folgenden Programmcode, um die Eigenschaft "Author" der Einheit zu aktualisieren.

```
//Set value
string newAuthor = "Z012345";
unit1.Author = newAuthor;
//Using SetAttributes()
string newAuthor = "Z012345";
var attrList = new List<KeyValuePair<string, object>>()
{
    new KeyValuePair<string, object>("Author", newAuthor)
};
unit1.SetAttributes(attrList));
//Using SetAttribute()
string newAuthor = "Z012345";
IEngineeringObject unit = (IEngineeringObject)unit2;
unit.SetAttribute("Author", newAuthor);
```

Fehlermeldung

Wenn versucht wird, einen falschen Wert einzugeben, z. B. mit Leerzeichen am Anfang, oder wenn der Wert unzulässige Zeichen enthält, wird eine Fehlermeldung ausgegeben. Eine Recoverable Exception wird mit folgender Fehlermeldung ausgelöst: "Der Wert von Attribut 'Author' enthält ein ungültiges Zeichen an Position 0."

```
PlcUnit unit1 = unitComposition.Create("Unit1");
try
{
    unit1.Author = " Author";
}
catch (EngineeringTargetException e)
{
    Console.WriteLine(e.Message);
}
```

Programmcode: Eigenschaft "Comment" der Einheit aktualisieren

Die Eigenschaft 'Comment' der Einheit kann nicht direkt festgelegt oder abgerufen werden, weil es sich dabei um MultilingualText handelt. Sie können aber die Eigenschaft "Text" des Kommentars aktualisieren. Das Comment.Item enthält die verschiedenen Kulturen ("cultures") des TIA Projekts. Es kann über die Zuordnung von Werten festgelegt werden. Die Projektsprachen sind in der Comment.Item composition indexiert, wobei mit 0 die erste Projekt-Standardsprache angegeben wird. Wenn mehrere Sprachen angegeben sind, können Sie die Liste durchlaufen. Ansonsten wird, wenn die Kultur nicht vorhanden ist, eine EngineeringTargetException ausgelöst.

```
//Setting the default first culture comment Item text:  
unit1.Comment.Items[0].Text = "new Comment";  
//Setting other culture comment Item Text:  
unit1.Comment.Items[1].Text = "neuro Kommentar";
```

Fehlermeldung

Wenn versucht wird, SetAttribute, SetAttributes, GetAttribute, GetAttributes. zu verwenden, wird eine EngineeringNotSupportedException mit folgender Fehlermeldung ausgegeben: "Kommentar wird durch Typ 'Siemens.Engineering.SW.Units.PlcUnit' nicht unterstützt."

```
try  
{  
    ((IEngineeringObject)unit1).SetAttribute("Comment", "new comments");  
}  
catch (EngineeringNotSupportedException e)  
{  
    Console.WriteLine(e.Message);  
}
```

Wenn versucht wird, eine Beziehung zu einer nicht vorhandenen Kultur herzustellen, wird eine Fehlermeldung ausgegeben. Eine EngineeringTargetException wird mit folgender Fehlermeldung ausgelöst: "Das Argument 'index' (3) liegt außerhalb des gültigen Wertebereichs."

```
try  
{  
    unit1.Comment.Items[3].Text = "new Comment";  
}  
catch (EngineeringTargetException e)  
{  
    Console.WriteLine(e.Message)  
}
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

7.19.25.6 Objekt einer Software-Einheit veröffentlichen

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Sie können mit TIA Portal Openness das Attribut "Access" zugrundeliegender Objekte von Einheiten abrufen oder einstellen, so dass Sie deren Zugänglichkeit zwischen Einheiten beeinflussen können.

In Openness-Einheiten werden die folgenden Objekte unterstützt:

- Programmsteine und Datensteine – außer OBs
- PLC-Typen

Der folgende mögliche Wert des Attributs Access dargestellt als Enumeration UnitAccessType:

- Veröffentlicht
- Unveröffentlicht

Hinweis

Verfügbar ist das Attribut "Access" nur an Objekten unter einer Einheit. Wenn es also nicht unter mit zu PlcUnit gehörenden Objekten abgerufen oder eingestellt werden soll, wird eine EngineeringNotSupportedException ausgelöst. Im Fall eines falschen Werts (z. B. ungültige Enumerationswerte) wird eine EngineeringTargetInvocationException ausgelöst.

Programmcode: Zugriffsattribut von PLC-Bausteinen konfigurieren

```
//Getting attribute value with GetAttribute
...
PlcBlock block = m_PlcUnit.BlockGroup.Find("FB_1");
UnitAccessType currentAccessValue = (UnitAccessType)block.GetAttribute("Access");
...
//Getting attribute value with GetAttributes
...
PlcBlock block = m_PlcUnit.BlockGroup.Block.Find("FB_1");
List<string> attrList = new List<string>();
{
    "Access"
};
IList<object> getAttributeValues = block.GetAttributes(attrList));
...
//Setting attribute value with SetAttribute
...
PlcBlock block = m_PlcUnit.BlockGroup.Blocks.Find("FB_1");
UnitAccessType newAccessValue = UnitAccessType.Published;
block.SetAttribute("Access", newAccessValue);
...
//Setting attribute value with SetAttributes
...
PlcBlock block = m_PlcUnit.BlockGroup.Blocks.Find("FB_1");
UnitAccessType newAccessValue = UnitAccessType.Published;
IList attrList = new List<KeyValuePair<string, object>>()
{
    new KeyValuePair<string, object>("Access", newAccessValue),
};
block.SetAttributes(attrList);
...
```

Programmcode: Zugriffsattribut von PLC-Bausteinen konfigurieren

```
//Getting attribute value with GetAttribute
...
PlcType type = m_PlcUnit.TypeGroup.Types.Find("UDT_1");
UnitAccessType currentValue = (UnitAccessType)type.GetAttribute("Access");
...
//Getting attribute value with GetAttributes
...
PlcType type = m_PlcUnit.TypeGroup.Types.Find("UDT_1");
List<string> attrList = new List<string>()
{
    "Access"
};
IList<object> getAttributeValue = type.GetAttributes(attrList));
...
//Setting attribute value with SetAttribute
...
PlcType type = m_PlcUnit.TypeGroup.Types.Find("UDT_1");
UnitAccessType newAccessValue = UnitAccessType.Published;
type.SetAttribute("Access", newAccessValue);
...
//Setting attribute value with SetAttributes
...
PlcType type = m_PlcUnit.TypeGroup.Types.Find("UDT_1");
UnitAccessType newAccessValue = UnitAccessType.Published;
IList attrList = new List<KeyValuePair<string, object>>()
{
    new KeyValuePair<string, object>("Access", newAccessValue),
};
type.SetAttributes(attrList));
...
```

Siehe auch

- Verbindung zum TIA Portal aufbauen (Seite 79)
- Projekt öffnen (Seite 111)

7.19.25.7 Externe Quellen in Einheiten hinzufügen

Voraussetzung

- Die Anwendung TIA Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Sie können mit TIA Portal Openness eine externe Quelle unter einer Software-Einheit hinzufügen. Sie können Quelldateien wie .SCL, .DB, .UDT unter Einheiten hinzufügen.

Mit der Openness API können Sie die folgenden Aufgaben mit ExternalSourceGroup unter Software-Einheiten ausführen:

- ExternalSourceGroup unter Einheiten hinzufügen
- Bausteine/UDTs aus einer externen Quelle unter Software-Einheiten generieren
- Quelle aus Bausteinen/UDTs unter Einheiten exportieren

Programmcode: ExternalSourceGroup hinzufügen

Ändern Sie folgenden Programmcode, um eine externe Quelle unter einer Software-Einheit hinzuzufügen:

```
PlcUnitProvider unitProvider = controllerTarget.GetService<PlcUnitProvider>();
PlcUnit newUnit = unitProvider.UnitGroup.Units.Create("Unit_1");
PlcExternalSource unitExternalSource =
newUnit.ExternalSourceGroup.ExternalSources.CreateFromFile(externalSourceFilename,
GetFilePath(externalSourceFilename));
```

Programmcode: Bausteine/UDTs aus Quelle generieren

Ändern Sie folgenden Programmcode, um Bausteine/UDTs mit leeren Rückgabetypen aus einer externen Quelle unter einer Software-Einheit zu generieren:

```
unitExternalSource.GenerateBlocksFromSource();
```

Ändern Sie folgenden Programmcode, um Bausteine/UDTs mit einer aus einer externen Quelle abgerufenen Bausteinsammlung unter einer Software-Einheit zu generieren:

```
// Blocks/UDTs gets generated only if there are no compilation errors
IList<IEngineeringObject> generatedBlocksUnderUnitList =
unitExternalSource.GenerateBlocksFromSource(GenerateBlockOption.None);
// Blocks/UDTs will be generated regardless of any compilation errors
IList<IEngineeringObject> generatedBlocksUnderUnitList =
unitExternalSource.GenerateBlocksFromSource(GenerateBlockOption.KeepOnError);
```

Programmcode: Quelle aus Bausteinen/UDTs exportieren

Ändern Sie folgenden Programmcode, um eine Quelle aus Bausteinen unter einer Software-Einheit zu generieren:

```
// For blocks
PlcBlock unitPlcBlock = newUnit.BlockGroup.Blocks.Find(generatedBlockName);
// Generate source from blocks with dependencies
newUnit.ExternalSourceGroup.GenerateSource(new[] { unitPlcBlock }, new
FileInfo(outputFileGeneratedPath), GenerateOptions.WithDependencies);
// Generate source from blocks without dependencies
newUnit.ExternalSourceGroup.GenerateSource(new[] { unitPlcBlock }, new
FileInfo(outputFileGeneratedPath), GenerateOptions.None);
```

Ändern Sie folgenden Programmcode, um eine Quelle aus UDTs unter einer Software-Einheit zu generieren:

```
// For UDTs
PlcType unitPlcUdt = newUnit.TypeGroup.Types.Find(generatedUdtName);
//Generate source from UDTs with dependencies
newUnit.ExternalSourceGroup.GenerateSource(new[] { unitPlcUdt }, new
FileInfo(outputFileGeneratedPath), GenerateOptions.WithDependencies);
//Generate source from UDTs without dependencies
newUnit.ExternalSourceGroup.GenerateSource(new[] { unitPlcUdt }, new
FileInfo(outputFileGeneratedPath), GenerateOptions.None);
```

Programmcode: Externe Quelldateigruppe enumerieren

Ändern Sie folgenden Programmcode, um die Zusammenstellung der externen Quelldateigruppe unter der aktuellen Einheit zu enumerieren:

```
PlcUnitProvider unitProvider = controllerTarget.GetService<PlcUnitProvider>();
PlcUnit newUnit = unitProvider.UnitGroup.Units.Find(textBoxAddNewUnit.Text);
PlcExternalSourceComposition unitExtSrcComposition =
newUnit.ExternalSourceGroup.ExternalSources;
foreach (PlcExternalSource unitExtSrc in unitExtSrcComposition)
{
unitExtSrc.GenerateBlocksFromSource(GenerateBlockOption.None);
}
```

7.19.25.8 Einheiten als Masterkopien

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Eine Einheit wird erstellt
Siehe Mit Software-Einheiten arbeiten (Seite 595)

Anwendung

Sie können mit TIA Portal Openness Einheiten als Masterkopien in die Projektbibliothek und in die globale Bibliothek kopieren.

Mit Einheiten als Masterkopie können Sie folgende Aktionen durchführen, während Sie TIA Portal Openness verwenden:

- Eine Einheit als Masterkopie in der Projektbibliothek aus Software-Einheiten erstellen
- Eine Einheit als Masterkopie in der globalen Bibliothek aus Software-Einheiten erstellen
- Eine Einheit aus einer Masterkopie in der Projektbibliothek in Software-Einheiten in PNV neu erstellen
- Eine Einheit aus einer Masterkopie in der globalen Bibliothek in Software-Einheiten in PNV neu erstellen

Programmcode

Ändern Sie folgenden Programmcode, um eine neue Einheit als Masterkopie in der Projektbibliothek aus Software-Einheiten mit Hilfe von MasterCopyComposition zu erstellen.

```
PlcUnitProvider m_UnitProvider = plc.GetService<PlcUnitProvider>();
PlcUnitComposition m_UnitComposition = m_UnitProvider.UnitGroup.Units;
PlcUnit m_SoftwareUnit1 = unitComposition.Create("Unit_1");//Assuming existing units
IMasterCopySource m_UnitAsMasterCopy = (IMasterCopySource)m_SoftwareUnit1;//Assuming that
m_SoftwareUnit1 is present in the PLC
m_ProjectLibrary.MasterCopyFolder.MasterCopies.Create(m_UnitAsMasterCopy);
```

Ändern Sie folgenden Programmcode, um eine Einheit als Masterkopie in der globalen Bibliothek aus Software-Einheiten zu erstellen.

```
...
IMasterCopySource m_UnitAsMasterCopy = (IMasterCopySource)m_SoftwareUnit1;//Assuming that
m_SoftwareUnit1 is present in the PLC
m_TiaPortal.GlobalLibraries.Open(libfile, OpenMode.ReadWrite);
GlobalLibrary m_GlobalLibrary = m_TiaPortal.GlobalLibraries[0];
m_GlobalLibrary.MasterCopyFolder.MasterCopies.Create(m_UnitAsMasterCopy)
...
```

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Wenn versucht wird, eine neue Einheit in der schreibgeschützten globalen Bibliothek zu erstellen, wird eine Fehlermeldung ausgegeben. Eine Recoverable Exception wird mit der Meldung "Schreiben in schreibgeschützte Bibliotheken nicht möglich" ausgelöst.

```
m_TiaPortal.GlobalLibraries.Open(libfile, OpenMode.ReadOnly);
GlobalLibrary m_GlobalLibrary = m_TiaPortal.GlobalLibraries[0];
try
{
m_GlobalLibrary.MasterCopyFolder.MasterCopies.Create(m_UnitAsMasterCopy);
}
catch (Exception e)
{
Console.WriteLine(e.Message);
}
```

Ändern Sie folgenden Programmcode, um eine Einheit aus einer Masterkopie in der Projektbibliothek in Software-Einheiten in PNV neu zu erstellen:

```
...
ProjectLibrary m_ProjectLibrary = project.ProjectLibrary;
PlcUnitProvider m_UnitProvider = plc.GetService<PlcUnitProvider>();
PlcUnitComposition m_UnitComposition = m_UnitProvider.UnitGroup.Units;
...
MasterCopy mc_Unit_2 = m_ProjectLibrary.MasterCopyFolder.MasterCopies.Find("Unit_2");
m_UnitComposition.CreateFrom(mc_Unit_2); //Recreate a Unit from Project Library to
SoftwareUnits folder
```

Ändern Sie folgenden Programmcode, um eine Einheit aus einer Masterkopie in der globalen Bibliothek in Software-Einheiten in PNV neu zu erstellen:

```
...
GlobalLibrary m_GlobalLibrary = m_TiaPortal.GlobalLibraries[0];
...
mc_Unit_2 = m_GlobalLibrary.MasterCopyFolder.MasterCopies.Find("Unit_2");
m_UnitComposition.CreateFrom(mc_Unit_2); //Recreate a unit from Global Library to
SoftwareUnits folder
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

[Mit Software-Einheiten arbeiten \(Seite 595\)](#)

7.19.25.9 Vorhandene Beziehungen aktualisieren und Beziehungen erstellen/löschen

Voraussetzung

- Die Anwendung TIA Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Sie können mit TIA Portal Openness die Beziehungen von PLC-Einheiten ändern und so die Zugänglichkeit der Objekte in anderen PLC-Einheiten oder außerhalb von PLC-Einheiten steuern.

Mit TIA Portal Openness können Sie die folgenden Arten von Änderungen vornehmen:

- Neue Beziehungen erstellen
- Vorhandene Beziehungen löschen
- Vorhandene Beziehungen ändern

Programmcode: Neue Beziehungen erstellen

Sie können eine neue Beziehung erstellen, indem Sie die entsprechende Beziehungsart und den Namen des bezogenen Objekts angeben.

```
private PlcUnitRelation m_SoftwareUnitRelation;
private PlcUnitRelation m_NonUnitDBRelation;
private PlcUnitRelation m_TODBRelation;
...
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);
PlcUnitProvider plcUnitProvider = plcTarget.GetService<PlcUnitProvider>();
m_PlcUnit = plcUnitProvider.UnitGroup.Units[0]; //assuming existing units
```

Ändern Sie folgenden Programmcode, um eine neue Beziehung zu erstellen, mit der Sie auf ein Objekt in einer anderen PLC-Einheit zugreifen:

```
...
//assuming Plc unit "Unit_2" is already existing in the Plc
m_SoftwareUnitRelation = m_PlcUnit.Relations.Create("Unit_2",
UnitRelationType.SoftwareUnit);
...
```

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Ändern Sie folgenden Programmcode, um eine neue Beziehung zu erstellen, mit der Sie auf einen PLC-Datenbaustein außerhalb von PLC-Einheiten zugreifen:

```
...
//assuming Plc data block "Data_block_1" is already existing in the Plc
m_NonUnitDBRelation = m_PlcUnit.Relations.Create("Data_block_1",
UnitRelationType.NonUnitDB);
...

```

Ändern Sie folgenden Programmcode, um eine neue Beziehung zu erstellen, mit der Sie auf ein Technologieobjekt außerhalb von PLC-Einheiten zugreifen:

```
...
//assuming Technological object "SpeedAxis_1" is already existing in the Plc
m_TODBRelation = m_PlcUnit.Relations.Create("SpeedAxis_1", UnitRelationType.TODB);
...

```

Mögliche Fehlerszenarios beim Erstellen einer neuen Beziehung:

Ändern Sie folgenden Programmcode, um eine neue Beziehung zu erstellen, mit der auf ein nicht vorhandenes bezogenes Objekt zugegriffen wird:

```
...
//assuming Plc unit "NonExistingUnit" is not existing in the Plc yet
m_SoftwareUnitRelation = m_PlcUnit.Relations.Create("NonExistingUnit", UnitRelationType.
SoftwareUnit);
...

```

Hinweis

Im obigen Programmcode wird die Beziehung erstellt und keine Ausnahme ausgelöst.

Ändern Sie folgenden Programmcode, um eine neue Beziehung zu erstellen, indem Sie einen Namen für das bezogene Objekt angeben, der nicht den TIA-Benennungsregeln entspricht:

```
...
m_SoftwareUnitRelation = m_PlcUnit.Relations.Create(" Unit_2",
UnitRelationType.SoftwareUnit);
...

```

Hinweis

Im obigen Programmcode wird die Beziehung nicht erstellt und es wird eine wiederherstellbare Ausnahme aufgrund von führenden Leerzeichen ausgelöst.

Ändern Sie folgenden Programmcode, um eine neue Beziehung von der Einheit mit sich selbst zu erstellen:

```
...
//assuming m_PlcUnit is assigned to Plc unit "Unit_1"
m_SoftwareUnitRelation = m_PlcUnit.Relations.Create("Unit_1",
UnitRelationType.SoftwareUnit);
...
```

Ändern Sie folgenden Programmcode, um eine neue Beziehung zu erstellen, die ein Duplikat einer bereits bestehenden Beziehung mit der gleichen Beziehungsart und dem gleichen bezogenen Objekt ist:

```
...
//assuming a relation with the same relation type and related object is already existing
m_SoftwareUnitRelation = m_PlcUnit.Relations.Create("Unit_2",
UnitRelationType.SoftwareUnit);
...
```

Ändern Sie folgenden Programmcode, um eine neue Beziehung zu erstellen, indem Sie einen leeren String für den Namen des bezogenen Objekts angeben:

```
...
m_SoftwareUnitRelation = m_PlcUnit.Relations.Create(string.Empty,
UnitRelationType.SoftwareUnit);
...
```

Hinweis

In allen obigen Programmcode-Abschnitten mit Fehlerszenarios wird die Beziehung nicht erstellt und es wird eine wiederherstellbare Ausnahme ausgelöst.

Programmcode: Vorhandene Beziehungen löschen

Sie können eine neue Beziehung erstellen, indem Sie die entsprechende Beziehungsart und den Namen des bezogenen Objekts angeben.

```
private PlcUnitRelation m_SoftwareUnitRelation;
private PlcUnitRelation m_NonUnitDBRelation;
private PlcUnitRelation m_TODBRelation;
...
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);
PlcUnitProvider plcUnitProvider = plcTarget.GetService<PlcUnitProvider>();
m_PlcUnit = plcUnitProvider.UnitGroup.Units[0]; //assuming existing units
m_PlcUnit2 = plcUnitProvider.UnitGroup.Units[1]; //assuming existing units
m_SoftwareUnitRelation = m_PlcUnit.Relations.Create("Unit_2",
UnitRelationType.SoftwareUnit);
m_NonUnitDBRelation = m_PlcUnit.Relations.Create("DB_Global", UnitRelationType.NonUnitDB);
m_TODBRelation = m_PlcUnit.Relations.Create("Axis_TO", UnitRelationType.TODB);
```

7.19 Funktionen für den Zugriff auf die Daten eines PLC-Geräts.

Sie können Beziehungen auf mehrere Arten löschen.

Ändern Sie folgenden Programmcode, um die Beziehung zu löschen, auf die der Indexer zugreift:

```
...  
m_PlcUnit.Relations[0].Delete();  
...
```

Ändern Sie folgenden Programmcode, um die Beziehung direkt zu löschen:

```
...  
m_SoftwareUnitRelation.Delete();  
...
```

Programmcode: Vorhandene Beziehungen ändern

Sie können Beziehungen auf mehrere Arten ändern:

Sie können eine neue Beziehung erstellen, indem Sie die entsprechende Beziehungsart und den Namen des bezogenen Objekts angeben.

```
private PlcUnitRelation m_Relation;  
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);  
PlcUnitProvider plcUnitProvider = plcTarget.GetService<PlcUnitProvider>();  
m_PlcUnit = plcUnitProvider.UnitGroup.Units[0];  
//assuming existing units  
m_PlcUnit2 = plcUnitProvider.UnitGroup.Units[1];  
//assuming existing units  
m_Relation = m_PlcUnit.Relations.Create("Unit_2", UnitRelationType.SoftwareUnit);  
...
```

Ändern Sie folgenden Programmcode, um den Namen des bezogenen Objekts zu aktualisieren, indem Sie einen neuen Wert für das Attribut RelatedObject direkt angeben:

```
...  
m_Relation.RelatedObject = "Unit_3";  
...
```

Ändern Sie folgenden Programmcode, um den Namen des bezogenen Objekts zu aktualisieren, indem Sie einen neuen Wert für das Attribut RelatedObject über SetAttribute zuordnen:

```
...  
m_Relation.SetAttribute("RelatedObject", "Unit_4");  
...
```

Ändern Sie folgenden Programmcode, um den Namen des bezogenen Objekts zu aktualisieren, indem Sie einen neuen Wert für das Attribut RelatedObject über SetAttributes zuordnen:

```
...
IList attrList = new List<KeyValuePair<string, object>>();
{
    new KeyValuePair<string, object>("RelatedObject", "Unit_4")
};
m_Relation.SetAttributes(attrList);
...
```

Mögliche Fehlerszenarien beim Ändern vorhandener Beziehungen

Ändern Sie folgenden Programmcode, um eine Beziehung so zu aktualisieren, dass sie auf ein nicht vorhandenes bezogenes Objekt zugreift:

```
...
//assuming Plc unit "NonExistingUnit" is not existing in the Plc yet
m_Relation.RelatedObject = "NonExistingUnit";
...
```

Hinweis

Durch den obigen Programmcode wird die Beziehung geändert und keine Ausnahme ausgelöst.

Ändern Sie folgenden Programmcode, um eine Beziehung zu aktualisieren, indem Sie einen Namen für das bezogene Objekt angeben, der nicht den TIA-Benennungsregeln entspricht:

```
...
m_Relation.RelatedObject = " Unit_2";
...
```

Ändern Sie folgenden Programmcode, um eine Beziehung so zu aktualisieren, dass sie auf sich selbst zugreift:

```
...
//assuming m_Relation is defined under Plc unit "Unit_1"
m_Relation.RelatedObject = "Unit_1";
...
```

Hinweis

Durch den obigen Programmcode wird die Beziehung nicht geändert und es wird eine wiederherstellbare Ausnahme aufgrund von führenden Leerzeichen ausgelöst.

7.20 Funktionen der Versionskontrollschnittstelle (VCI)

Ändern Sie folgenden Programmcode, um eine Beziehung so zu aktualisieren, dass ein Duplikat zu einer bereits bestehenden Beziehung mit der gleichen Beziehungsart und dem gleichen bezogenen Objekt entsteht:

```
...
//assuming a relation with the same relation type and related object is already existing
m_Relation.RelatedObject = "Unit_2";
...
```

Ändern Sie folgenden Programmcode, um eine Beziehung zu aktualisieren, indem Sie einen leeren String für den Namen des bezogenen Objekts angeben:

```
...
m_Relation.RelatedObject = string.Empty;
...
```

Hinweis

In allen obigen Programmcode-Abschnitten wird die Beziehung nicht geändert und es wird eine wiederherstellbare Ausnahme ausgelöst.

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.20 Funktionen der Versionskontrollschnittstelle (VCI)

7.20.1 Auf die VCI-Systemgruppe im Projekt zugreifen

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe [Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
Siehe [Projekt öffnen \(Seite 111\)](#)

Anwendung

Sie können in TIA Portal Openness vom Projekt aus auf die VCI-Systemgruppe zugreifen, indem Sie aus dem Projekt ein VersionControlInterface als Service aufrufen. Das VersionControlInterface selbst ist immer ein Service und gibt ein Objekt zurück, das nicht null ist.

Programmcode

Ändern Sie folgenden Programmcode, um die Workspace-Systemgruppe aus dem VersionControlInterface abzurufen:

```
Siemens.Engineering.Project project = tiaPortal.Projects[0];
Siemens.Engineering.VersionControl.VersionControlInterface versionControlInterface =
project.GetService<VersionControlInterface>();
Siemens.Engineering.VersionControl.WorkspaceSystemGroup workspaceSystemGroup =
versionControlInterface.WorkspaceGroup;;
```

Siehe auch

[Projekt öffnen \(Seite 111\)](#)

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

7.20.2 Benutzergruppen in VCI-Gruppe enumerieren

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
[Siehe Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
[Siehe Projekt öffnen \(Seite 111\)](#)

Programmcode

Ändern Sie folgenden Programmcode, um alle VCI-Workspace-Benutzergruppen in anderen VCI-Workspace-Gruppen zu enumerieren.

```
WorkspaceUserGroupComposition userGroupComposition = workspaceGroup.Groups;
foreach (Siemens.Engineering.VersionControl.WorkspaceUserGroup workspaceUserGroup in
userGroupComposition)
{
//...
}
```

7.20 Funktionen der Versionskontrollschnittstelle (VCI)

Ändern Sie folgenden Programmcode, um auf einzelne Workspace-Benutzergruppen in anderen VCI-Workspace-Gruppen zuzugreifen:

```
Siemens.Engineering.VersionControl.WorkspaceUserGroup workspaceUserGroup =  
workspaceGroup.Groups.Find("Some Group Name");
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.20.3 VCI-Benutzergruppe in VCI-Gruppe erstellen

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
[Siehe Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
[Siehe Öffnen eines Projekts \(Seite 111\)](#)

Anwendung

Sie können mit TIA Portal Openness eine Workspace-Benutzergruppe in einer Workspace-Gruppe erstellen:

Programmcode

Ändern Sie folgenden Programmcode, um eine Workspace-Benutzergruppe zu erstellen:

```
VersionControlInterface versionControlInterface =  
project.GetService<VersionControlInterface>();  
WorkspaceSystemGroup workspaceGroupComposition =  
versionControlInterface.WorkspaceGroup.Groups;  
WorkspaceUserGroup result = workspaceGroupComposition.Create("NewWorkspaceUserGroup");
```

Hinweis

Um eine neue Workspace-Benutzergruppe anhand des Namens erstellen zu können, muss der angegebene Name gültig sein und darf nicht bereits in einer anderen Workspace-Benutzergruppe vorhanden sein.

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.20.4 Eigenschaften von VCI-Gruppen aktualisieren

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
[Siehe Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
[Siehe Projekt öffnen \(Seite 111\)](#)

Anwendung

Sie können mit TIA Portal Openness die Eigenschaften von Workspace-Benutzergruppen aktualisieren.

Die folgende Eigenschaft ist zur Aktualisierung in einer Workspace-Benutzergruppe verfügbar:

Name der Eigenschaft	Rückgabetyp	Beschreibung	Zugriffsmöglichkeit
Name	System.String	Name der Workspace-Benutzergruppe	Lesen/Schreiben

Programmcode

Ändern Sie folgenden Programmcode, um den Namen der Workspace-Benutzergruppe zu aktualisieren:

```
var workspaceUserGroup = ...;
workspaceUserGroup.Name = "New_Group_Name";
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.20.5 VCI-Benutzergruppe löschen

Voraussetzung

- Die Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Sie können mit TIA Portal Openess Workspace-Benutzergruppen löschen. Wenn Sie Workspace-Benutzergruppen löschen, werden alle anderen in der Workspace-Benutzergruppe enthaltenen Objekte ebenfalls gelöscht. Hierzu gehören auch andere Workspace-Benutzergruppen und VCI-Workspaces.

Programmcode

Ändern Sie folgenden Programmcode, um eine Workspace-Benutzergruppe zu löschen:

```
WorkspaceUserGroup workspaceUserGroup = ...;  
workspaceUserGroup.Delete();
```

Siehe auch

Verbindung zum TIA Portal aufbauen (Seite 79)

Projekt öffnen (Seite 111)

7.20.6 VCI-Workspaces in VCI-Gruppe enumerieren

Voraussetzung

- Die Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Programmcode

Ändern Sie folgenden Programmcode, um alle VCI-Workspaces in einer VCI-Gruppe zu enumerieren:

```
WorkspaceComposition workspaceComposition = workspaceSystemGroup.Workspaces;
foreach (Siemens.Engineering.VersionControl.Workspace workspace in workspaceComposition)
{
//...
}
```

Ändern Sie folgenden Programmcode, um einen bestimmten Workspace anhand des Namens zu finden:

```
Siemens.Engineering.VersionControl.Workspace workspace =
workspaceUserGroup.Workspaces.Find("SomeWorkspaceName");
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

7.20.7 VCI-Workspace in VCI-Gruppe erstellen

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
[Siehe Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
[Siehe Öffnen eines Projekts \(Seite 111\)](#)

Anwendung

Sie können mit TIA Portal Openness einen Workspace in einer Workspace-Benutzergruppe erstellen.

Einen Siemens.Engineering.VersionControl.Workspace können Sie mit der folgenden Erstellungsaktion erstellen:

```
Siemens.Engineering.VersionControl.WorkspaceComposition.Create(string name)
```

Programmcode

Ändern Sie folgenden Programmcode, um einen Workspace zu erstellen:

```
VersionControlInterface versionControlInterface =  
project.GetService<VersionControlInterface>();  
WorkspaceComposition workspaceComposition =  
versionControlInterface.WorkspaceGroup.Workspaces;  
Workspace result = workspaceComposition.Create("NewWorkspace");
```

Hinweis

Um einen neuen Workspace anhand des Namens erstellen zu können, muss der angegebene Name gültig sein und darf nicht bereits in einer anderen Workspace-Benutzergruppe vorhanden sein.

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.20.8 VCI-Workspace-Eigenschaften aktualisieren

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
[Siehe Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
[Siehe Öffnen eines Projekts \(Seite 111\)](#)

Anwendung

Sie können mit TIA Portal Openness die Eigenschaften eines VCI-Workspace aktualisieren.

Die folgenden Eigenschaften sind zur Aktualisierung in einem Workspace verfügbar:

Name der Eigenschaft	Rückgabetyp	Beschreibung	Zugriffsmöglichkeit
Name	System.String	Der Name des Workspace.	Lesen/Schreiben
RootPath	System.IO.DirectoryInfo	Der konfigurierte Workspace-Pfad.	Lesen/Schreiben

Programmcode: Eigenschaft "Name" aktualisieren

Ändern Sie folgenden Programmcode, um die Eigenschaft "Name" im Workspace zu aktualisieren:

```
var workspace = ...;
workspace.Name = "New_Workspace_Name";
```

Programmcode: Eigenschaft "Rootpath" aktualisieren

Ändern Sie folgenden Programmcode, um den Workspace-Pfad zu einer Verzeichnisangabe mit Stammverzeichnis konfigurieren: Wird dieser Wert auf null gesetzt, so wird dadurch die Konfiguration des Workspace-Stammpfads aufgehoben.

```
var workspace = ...;
workspace.RootPath = new DirectoryInfo(@"D:\Project_WS");
```

Ändern Sie folgenden Programmcode, um die Konfiguration eines Workspace aufzuheben:

```
var workspace = ...;
workspace.RootPath = null;
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.20.9 VCI-Workspace löschen

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
[Siehe Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
[Siehe Öffnen eines Projekts \(Seite 111\)](#)

Anwendung

Sie können mit TIA Portal Openness einen Workspace löschen. Beim Löschen eines Workspace werden alle Zuordnungen des Workspace ebenfalls gelöscht.

Programmcode

Ändern Sie folgenden Programmcode, um einen Workspace zu löschen:

```
Workspace workspace = ...;  
workspace.Delete();
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.20.10 VCI-Workspace-Zuordnungen in VCI enumerieren

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
[Siehe Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
[Siehe Projekt öffnen \(Seite 111\)](#)

Programmcode

Ändern Sie folgenden Programmcode, um alle VCI-Workspace-Zuordnungen in einem Workspace zu enumerieren:

```
var workspace = ...;  
WorkspaceMappingComposition mappings = workspace.Mappings;  
foreach (Siemens.Engineering.VersionControl.WorkspaceMapping workspaceMapping in mappings)  
{  
//...  
}
```

Ändern Sie folgenden Programmcode, um auf einzelne Workspace-Zuordnungen für das spezielle Engineering-Objekt zuzugreifen:

```
var workspace = ...;  
IEngineeringObject versionControlSupportedEngineeringObject = ...;  
Siemens.Engineering.VersionControl.WorkspaceMapping workspaceMapping =  
workspace.Mappings.Find(versionControlSupportedEngineeringObject);
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.20.11 VCI-Workspace-Zuordnung in VCI-Workspace erstellen

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Sie können mit TIA Portal Openness eine Workspace-Zuordnung in einem VCI-Workspace erstellen.

Eine Siemens.Engineering.VersionControl.WorkspaceMapping können Sie mit der folgenden Signatur einer Erstellungsaktion erstellen:

```
Siemens.Engineering.VersionControl.WorkspaceMappingComposition.Create  
e(string relativeWorkspacePath, IEngineeringObject  
linkedProjectObject)
```

Programmcode

Ändern Sie folgenden Programmcode, um eine Workspace-Zuordnung zu erstellen:

```
var workspace = ...;  
var plcBlock = ...;  
var result = workspace.Mappings.Create(@"\TestCopy\Block_1.xml", plcBlock);
```

Eine wiederherstellbare Ausnahme wird in folgenden Fällen ausgegeben:

- Das verknüpfte Objekt ist kein gültiges VCI-unterstütztes Objekt.
- Das verknüpfte Objekt ist bereits zugeordnet.
- Der relative Dateipfad enthält ungültige Dateizeichen.
- Der relative Dateipfad enthält die Navigation zum übergeordneten Verzeichnis.

Siehe auch

Verbindung zum TIA Portal aufbauen (Seite 79)

Projekt öffnen (Seite 111)

7.20.12 VCI-Workspace-Zuordnungen aktualisieren

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Sie können mit dem TIA Openness Portal die Eigenschaften einer VCI-Workspace-Zuordnung aktualisieren.

Die folgenden Eigenschaften sind bei einer Workspace-Zuordnung verfügbar:

Name der Eigenschaft	Rückgabetyp	Beschreibung	Zugriffsmöglichkeit
RelativeWorkspacePath	System.String	Der relative Dateipfad zur Darstellung der verknüpften Objekte in der Quellcodeverwaltung.	Lesen/Schreiben
LinkedProjectObject	IEngineeringObject	Das mit VCI kompatible Planungsobjekt, das mit der Datei in der Quellcodeverwaltung verknüpft ist.	Lesen

Programmcode

Ändern Sie folgenden Programmcode, um den relativen Dateipfad der Workspace-Zuordnung zu aktualisieren:

```
var workspaceMapping = ...;
workspaceMapping.RelativeWorkspacePath = @"\\Test\\NewBlock_1.xml";
```

Siehe auch

Verbindung zum TIA Portal aufbauen (Seite 79)

Projekt öffnen (Seite 111)

7.20.13 VCI-Workspace-Zuordnung löschen

Voraussetzung

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Prinzip

Sie können mit TIA Portal Openness eine VCI-Workspace-Zuordnung löschen.

Programmcode

Ändern Sie folgenden Programmcode, um eine VCI-Workspace-Zuordnung zu löschen:

```
WorkspaceMapping workspaceMapping = ...;  
workspaceMapping.Delete();
```

Siehe auch

Verbindung zum TIA Portal aufbauen (Seite 79)

7.20.14 Workspace-Zuordnung synchronisieren

Prinzip

- Die Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Sie können mit TIA Portal Openness den Status einer Workspace-Zuordnung anzeigen.

Programmcode: Synchronisationsstatus einzelner Objekte aktualisieren

Um den Status einzelner Objekte anzuzeigen, zu aktualisieren oder zu synchronisieren, müssen Sie zunächst aus einer Workspace-Zuordnung den Siemens.Engineering.VersionControl.WorkspaceMapping.IndividualObjectSynchronizationStatus anfordern.

7.20 Funktionen der Versionskontrollschnittstelle (VCI)

Es wird empfohlen, vor dem Aufruf von Aktionen des Synchronisationsdienstes für einzelne Objekte eine Prüfung auf null durchzuführen, da unklar ist, ob die bestehende Zuordnung die Synchronisation einzelner Objekte unterstützt.

```
var individualObjectSynchronizationStatus =
workspaceMapping.GetService<IndividualObjectSynchronizationStatus>();
if(individualObjectSynchronizationStatus != null);
{
    //GetStatus()...
    //UpdateStatus()..
    //Synchronize()..
}
```

Programmcode: Status abrufen

Sie können den Synchronisationsstatus eines einzelnen Objekts mithilfe der folgenden Aktion anzeigen:

`Siemens.Engineering.VersionControl.IndividualObjectSynchronizationStatus.GetStatus()`.

Die Aktion gibt `Siemens.Engineering.VersionControl.IndividualObjectCompareResult` zurück.

Das Objekt `IndividualObjectCompareResult` informiert über den aktuellen Synchronisationsstatus.

```
public class IndividualObjectCompareResult
{
    CompareState CompareState { get; }
    IndividualObjectCompareDetails {get; }
}
```

Mit der Merker-Enumeration `IndividualObjectCompareDetails` werden Sie darüber informiert, welcher Teil der Workspace-Zuordnung geändert wurde (oder auch, dass beide Teile geändert wurden). Dieser Wert ist `None`, wenn der

`CompareState Equal` lautet. Wenn der Wert für `CompareState "ungleich"` lautet, kann diese Enumeration den Wert `ProjectObjectChanged` oder `WorkspaceFileChanged` oder sowohl `ProjectObjectChanged` als auch

`WorkspaceFileChanged` haben.

```
[Flags]public enum IndividualObjectCompareDetails
{
    None = 0,
    ProjectObjectChanged = 1,
    WorkspaceFileChanged = 2
}
```

Die Enumeration CompareState informiert darüber, ob ein Objekt Änderungen aufweist.

```
public enum CompareState
{
    Equal,
    Unequal,
    WorkspaceFileMissing,
    Unknown
}
```

Ändern Sie folgenden Programmcode, um den Status einer Workspace-Zuordnung abzurufen:

```
var workspaceMapping = ...;
var individualObjectSynchronizationStatus =
workspaceMapping.GetService<IndividualObjectSynchronizationStatus>();
if(individualObjectSynchronizationStatus != null)
{
    var compareResult = individualObjectSynchronizationStatus.GetStatus();
    //when compareState is unequal
    {
        if(compareResult.CompareState == CompareState.Unequal)
        {
            if(compareResult.IndividualObjectCompareDetails ==
IndividualObjectCompareDetails.WorkspaceFileChanged)
            {
                //Workspace file has changed
            }
            elseif(compareResult.IndividualObjectCompareDetails ==
(IndividualObjectCompareDetails.ProjectObjectChanged |
IndividualObjectCompareDetails.WorkspaceFileChanged))
            {
                //Both project object and workspace file has changed
            }
        }
    }
}
```

Programmcode: Synchronisationsstatus aktualisieren

Sie können das System dazu zwingen, einen Echtzeitvergleich zwischen dem aktuell verknüpften Projektobjekt und der Workspace-Datei durchzuführen. Der Vergleich kann bewirken, dass der Synchronisationsstatus auf den aktuellen Systemzustand geändert wird, und kann außerdem dazu dienen, den Status einer vor Kurzem verknüpften (aber nicht synchronisierten) Zuordnung festzustellen, ohne dabei das vorhandene Projektobjekt oder die Workspace-Datei zu verändern.

Ändern Sie folgenden Programmcode, um den Status einer Workspace-Zuordnung zu aktualisieren:

```
var workspaceMapping = ...;
var individualObjectSynchronizationStatus =
workspaceMapping.GetService<IndividualObjectSynchronizationStatus>();
if(individualObjectSynchronizationStatus != Null)
{
individualObjectSynchronizationStatus.UpdateStatus();
}
```

Programmcode: Workspace synchronisieren

Sie können das System dazu zwingen, einen Echtzeitvergleich zwischen dem aktuell verknüpften Projektobjekt und der Workspace-Datei durchzuführen. Der Vergleich kann bewirken, dass der Synchronisationsstatus auf den aktuellen Systemzustand geändert wird, und kann außerdem dazu dienen, den Status einer vor Kurzem verknüpften (aber nicht synchronisierten) Zuordnung festzustellen, ohne dabei das vorhandene Projektobjekt oder die Workspace-Datei zu verändern.

Mit der Enumeration SynchronizationMode wird das System darüber informiert, in welche Richtung es die Synchronisation vornehmen soll.

```
public enum SynchronizationMode
{
ProjectToWorkspace,
WorkspaceToProject
}
```

Mit dieser Aktion wird das System angewiesen, die Synchronisation vom verknüpften Projektobjekt zum verknüpften Workspace-Dateiobjekt bzw. in umgekehrter Richtung durchzuführen.

Ändern Sie folgenden Programmcode, um einen Workspace zu synchronisieren:

```
var workspaceMapping = ...;
var individualObjectSynchronizationStatus =
workspaceMapping.GetService<IndividualObjectSynchronizationStatus>();
if(individualObjectSynchronizationStatus != null)
{
individualObjectSynchronizationStatus.Synchronize(SynchronizationMode.ProjectToWorkspace);
}
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.21 Funktionen auf OPC

7.21.1 Konfiguration des sicheren Kommunikationsprotokolls für den OPC UA-Server

Voraussetzungen

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Einleitung

Sie können mit der TIA Portal Openness-Anwendung den OPC UA-Server mit der Sicherheitsrichtlinie Basic256Sha256 konfigurieren. Die Sicherheitsrichtlinie Basic256Sha256 muss in den Runtime-Einstellungen hinzugefügt werden. RDP muss die Eigenschaften in der xml-Konfigurationsdatei übersetzen.

Die Standardwerte lauten Enabled, Sign sowie Sign and Encrypt.

In der XML-Datei <Project>\OPC\uaserver\OPCUaServerWinCCPro.xml müssen Sie die Sicherheitsrichtlinie sowie Sicherheitsrichtlinien gemäß der ES-Gerätekonfiguration festlegen.

7.21 Funktionen auf OPC

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <OPCUA_Server_WinCC xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ua="http://opcfoundation.org/UA">
3
4      <SecuredApplication>
5          <BaseAddresses>
6              <ua:String>opc.tcp://[HostName]:4861</ua:String>
7          </BaseAddresses>
8          <SecurityProfileUris>
9              <SecurityProfile>
10                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#None</ProfileUri>
11                 <Enabled>true</Enabled>
12             </SecurityProfile>
13             <SecurityProfile>
14                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15</ProfileUri>
15                 <Enabled>true</Enabled>
16             </SecurityProfile>
17             <SecurityProfile>
18                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256</ProfileUri>
19                 <Enabled>true</Enabled>
20             </SecurityProfile>
21             <SecurityProfile>
22                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</ProfileUri>
23                 <Enabled>true</Enabled>
24             </SecurityProfile>
25         </SecurityProfileUris>
26     </SecuredApplication>
27
28     <ServerConfiguration>
29         <SecurityPolicies>
30             <SecurityPolicy>
31                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#None</ProfileUri>
32                 <MessageSecurityModes>None</MessageSecurityModes>
33             </SecurityPolicy>
34             <SecurityPolicy>
35                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15</ProfileUri>
36                 <MessageSecurityModes>Sign</MessageSecurityModes>
37             </SecurityPolicy>
38             <SecurityPolicy>
39                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic128Rsa15</ProfileUri>
40                 <MessageSecurityModes>SignAndEncrypt</MessageSecurityModes>
41             </SecurityPolicy>
42             <SecurityPolicy>
43                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256</ProfileUri>
44                 <MessageSecurityModes>Sign</MessageSecurityModes>
45             </SecurityPolicy>
46             <SecurityPolicy>
47                 <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256</ProfileUri>
48                 <MessageSecurityModes>SignAndEncrypt</MessageSecurityModes>
49             </SecurityPolicy>
50         <SecurityPolicy>
51             <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</ProfileUri>
52             <MessageSecurityModes>Sign</MessageSecurityModes>
53         </SecurityPolicy>
54         <SecurityPolicy>
55             <ProfileUri>http://opcfoundation.org/UA/SecurityPolicy#Basic256Sha256</ProfileUri>
56             <MessageSecurityModes>SignAndEncrypt</MessageSecurityModes>
57         </SecurityPolicy>
58     </SecurityPolicies>

```

Siehe auch

Verbindung zum TIA Portal aufbauen (Seite 79)

Projekt öffnen (Seite 111)

7.21.2 Sicherheitsrichtlinie für OPC UA festlegen

Voraussetzungen

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Öffnen eines Projekts (Seite 111)
- OPC UA Server ist aktiviert

Anwendung

Sie können mit der TIA Portal Openness-Anwendung die Sicherheitsrichtlinie in OPC UA festlegen. Sie können die Sicherheitsrichtlinie als dynamisches Attribut vom Typ markierte Enumeration implementieren: OpcUaSecurityPolicies. Die Sicherheitsrichtlinie ist in TIA Portal Openness nur verfügbar, wenn der OPC UA Server aktiviert ist. Wenn der OPC UA Server deaktiviert ist, wird bei dem Versuch, wegen eines anderen, nicht verfügbaren Attributs auf die Sicherheitsrichtlinie zuzugreifen, eine EngineeringNotSupportedException ausgelöst.

Die nachstehende Tabelle zeigt die möglichen Werte, die für Sicherheitsrichtlinien zu finden sind:

Name im TIA UI	Enumerations-Eintrag	Wert	Anmerkungen
-	NoneSelected	0	Entspricht der TIA UI, wenn kein Kontrollkästchen markiert ist.
No security	OpcUaSecurityPolicies-None	1	
Basic128Rsa15 - Sign	OpcUaSecurityPolicies128RSAS	2	
Basic128Rsa15 - Sign & Encrypt	OpcUaSecurityPolicies128RSASE	4	
Basic256 - Sign	OpcUaSecurityPolicies256S	8	
Basic256 - Sign & Encrypt	OpcUaSecurityPolicies256SE	16	
Basic256Sha256 - Sign	OpcUaSecurityPolicies256SHAS	32	
Basic256Sha256 - Sign & Encrypt	OpcUaSecurityPolicies256SHASE	64	

Programmcode

Um die Sicherheitsrichtlinie in OPC UA mit TIA Portal Openness festzulegen, ändern Sie folgenden Programmcode:

```
DeviceItem UpcUaSubmodule= ...;
object SecurityPolicies = UpcUaSubmodule.GetAttribute("OpcUaSecurityPolicies");
if(SecurityPolicies | OpcUaSecurityPolicies.OpcUaSecurityPolicies256S ==
OpcUaSecurityPolicies.OpcUaSecurityPolicies256S)
{
//Do something
}
UpcUaSubmodule.SetAttribute("OpcUaSecurityPolicies",
OpcUaSecurityPolicies.OpcUaSecurityPolicies256S |
OpcUaSecurityPolicies.OpcUaSecurityPolicies256SHASE);
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

7.22 SiVArc Openness

7.22.1 Einleitung

Einleitung

Mit der TIA Portal Openness-Anwendung können Sie SiVArc instanziieren. Sie benötigen eine Client-Anwendung für den Zugriff auf das TIA Portal und starten dann über die Openness-Funktion die SiVArc-Dienste. Weitere Einzelheiten zur Einrichtung und zum Zugriff auf Openness finden Sie im Benutzerhandbuch zum TIA Portal.

Einrichten der Anwendung

Um eine Client-Anwendung einzurichten, gehen Sie wie folgt vor:

1. Erstellen Sie eine Konsolenanwendung. Fügen Sie eine Referenz auf die Public API (`Siemens.Engineering.dll`) hinzu, verfügbar unter `_deployed \TIAPIV15SP1_11010001\PublicAPI\V15.1\ 936 Siemens.Engineerin.dll` oder über die installierten Binärdateien unter `PublicAPI\V15.1\ 937 Siemens.Engineerin.dll`.
2. Fügen Sie der Konfigurationsdatei Konfigurationsdetails hinzu. Ausführliche Informationen über die Konfigurationsdetails und den Zugriff auf die Public API finden Sie im TIA Openness Wiki.
3. Für den Zugriff auf den SiVArc-Dienst verwenden Sie die nachfolgend erwähnte API:

```
using (TiaPortal tia = new
TiaPortal(TiaPortaMode.WithUserInterface))
{
    Project myProject = tia.Projects.Open(new FileInfo(@"C:\Users
\z003exve\Documents\Automation\Project_Demo\Project_Demo.ap15"));
    //Wenn SiVArc nicht installiert ist, kann der Benutzer nicht auf
    den SiVArc-Dienst zugreifen (Übersetzungsfehler)
    Sivarc sivarc =myproject?.GetService<Sivarc>();
    if (sivarc !=null)
    {
    }
}
}
```

7.22.2 SiVArc-Diensteigenschaften

SiVArc-Diensteigenschaften

Die nachfolgende Tabelle führt die unterstützten Eigenschaften und Methoden für SiVArc auf:

Name der Eigenschaft	Beschreibung	Datentyp
AlarmRules	Ankerobjekt für alle Alarmregelobjekte	AlarmRulesBrowsable
ScreenRules	Ankerobjekt für alle Bildregelobjekte	ScreenRulesBrowsable
TextlistRules	Ankerobjekt für alle Textlistenregelobjekte	TextlistRulesBrowsable
TagRules	Ankerobjekt für alle Variablenregelobjekte	TagRulesBrowsable
CopyRules	Ankerobjekt für alle Kopierregelobjekte	CopyRulesBrowsable
Alarm Rules	Enumeration sämtlicher direkter Alarmregeln der ersten Ebene	AlarmRuleComposition
Groups	Enumeration sämtlicher direkter Alarmregelgruppen der ersten Ebene	AlarmRuleGroupComposition
ScreenRules	Enumeration sämtlicher direkter Bildregeln der ersten Ebene	ScreenRuleComposition
ScreenRules-Groups	Enumeration sämtlicher direkter Bildregelgruppen der ersten Ebene	ScreenRuleGroupComposition

Name der Eigenschaft	Beschreibung	Datentyp
TextlistRules	Enumeration sämtlicher direkter Textlistenregeln der ersten Ebene	TextlistRuleComposition
TextlistGroups	Enumeration sämtlicher direkter Textlistenregelgruppen der ersten Ebene	TextlistRuleGroupComposition
TagRules	Enumeration sämtlicher direkter Variablenregeln der ersten Ebene	TagRuleComposition
TagRulesGroups	Enumeration sämtlicher direkter Variablenregelgruppen der ersten Ebene	TagRuleGroupComposition
CopyRules	Enumeration sämtlicher direkter Kopierregeln der ersten Ebene	CopyRuleComposition
CopyRulesGroups	Enumeration sämtlicher direkter Kopierregelgruppen der ersten Ebene	CopyRuleGroupComposition

In der folgenden Tabelle wird die Zusammensetzung von AlarmRule und AlarmRuleGroup aufgeführt. Gleiches gilt für andere SiVArc-Objekte.

Methodenname	Parameter	Beschreibung	Datentyp
Find	Zeichenkette – Alarmregel-/Regelgruppenname	Sucht die Alarmregel/Alarmregelgruppe in der Sammlung der Alarmregeln/Alarmgruppen	AlarmRule
CreateFrom	MasterCopy – Masterkopie Alarmregel/Alarmregelgruppe	Kopiert die Masterkopie der Alarmregel/Alarmregelgruppe aus der Bibliothek in das Projekt und verwendet die Standardoption zum Ersetzen	AlarmRule
CreateFrom	MasterCopy – Masterkopie Alarmregel/Alarmregelgruppe, CreateOptions – Umbenennen/Ersetzen	Kopiert die Masterkopie der Alarmregel/Alarmregelgruppe aus der Bibliothek in das Projekt und verwendet die Option zum Erstellen	AlarmRule

7.22.3 Kopieren von Regeln oder Gruppen aus der Bibliothek

Voraussetzung

- Starten Sie die TIA Portal Openness-Anwendung. Weitere Informationen zu Verbindungen finden Sie im Benutzerhandbuch zum TIA Portal.
- Ein TIA Portal-Projekt mit Bildregeleditor, Bildregelgruppen und Masterkopie ist vorhanden.

Fall 1: Wenn Sie Regeln/Regelgruppen aus der Masterkopie in den Bildregeleditor kopieren

Über die globale Bibliothek "CreateFrom" können Regeln und Regelgruppen aus der globalen Bibliothek in den SiVarc-Regeleditor kopiert werden. Wenn erfolgreich, gibt die API-Funktion die ScreenRule/ScreenRuleGroup zurück. Der folgende Code zeigt, wie Regeln oder Regelgruppen aus der Masterkopie in den SiVARC-Editor kopiert werden:

```
// Finds screen rule master copy "Screen rule_1"
MasterCopy screenRuleMasterCopy =
    myProject.ProjectLibrary.MasterCopyFolder.MasterCopies.Find("Screen rule_1");

if (screenRuleMasterCopy != null)
{
    var rule = sivarc.ScreenRules.Rules.CreateFrom(screenRuleMasterCopy);
    if (rule != null)
    {
        Console.WriteLine("Copied Screen Rule Name: " + rule.Name);
        Console.WriteLine("Copied Screen Rule Comment: " + rule.Comment);
        Console.WriteLine("Copied Screen Rule Enabled: " + rule.Enabled);
        Console.WriteLine("Copied Screen Rule Condition: " + rule.Condition);
    }
}
```

Standardmäßig ist das Verhalten "Ersetzen".

Fall 2: Wenn Sie Regeln/Regelgruppen aus der Masterkopie kopieren, können die vorhandenen Regeln/Regelgruppen basierend auf der zweiten Parametervorgabe umbenannt oder ersetzt werden.

Wenn Regeln/Regelgruppen der Masterkopie bereits im SiVARC-Editor vorhanden sind und Sie versuchen, diese zu kopieren, werden die Regeln/Regelgruppen umbenannt. Die API "CreateOptions" erstellt die Regeln/Regelgruppen im SiVARC-Editor, sofern sie nicht vorhanden sind, und ersetzt andernfalls bereits vorhandene Regeln/Regelgruppen. Wenn erfolgreich, benennt die API-Funktion die ScreenRule/ScreenRuleGroup um. Das folgende Code-Snippet zeigt das Ersetzen von Regeln/Regelgruppen:

```
// Finds screen rule master copy "Screen rule_1"
MasterCopy screenRuleMasterCopy =
    myProject.ProjectLibrary.MasterCopyFolder.MasterCopies.Find("Screen rule_1");

if (screenRuleMasterCopy != null)
{
    var rule = sivarc.ScreenRules.Rules.CreateFrom(screenRuleMasterCopy, CreateOptions.Rename);
    if (rule != null)
    {
        Console.WriteLine("Copied Screen Rule Name: " + rule.Name);
        Console.WriteLine("Copied Screen Rule Comment: " + rule.Comment);
        Console.WriteLine("Copied Screen Rule Enabled: " + rule.Enabled);
        Console.WriteLine("Copied Screen Rule Condition: " + rule.Condition);
    }
}
```

7.22.4 Suchen von Bildregeln oder Bildregelgruppen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden. Weitere Informationen zu Verbindungen finden Sie im Benutzerhandbuch zum TIA Portal.
- Ein TIA Portal-Projekt mit Bildregeln und Bildregelgruppen ist vorhanden.

Suchen von Bildregeln in Bildgruppen

Fall 1: Suchen von Bildregeln im Bildregeleditor.

Mit der API `sivarc.ScreenRules.Rules` können Sie die verfügbaren Regeln im SiVArc-Bildregeleditor wie nachfolgend gezeigt finden:

```
// Collection of all immediate first level screen rules
ScreenRuleComposition screenRules = sivarc.ScreenRules.Rules;
if(screenRules != null && screenRules.Count > 0)
{
    // Finds screen rule
    ScreenRule rule = screenRules.Find("Screen rule_7");
    if(rule != null)
    {
        Console.WriteLine("Screen Rule Name: " + rule.Name);
        Console.WriteLine("Screen Rule Comment: " + rule.Comment);
        Console.WriteLine("Screen Rule Enabled: " + rule.Enabled);
        Console.WriteLine("Screen Rule Condition: " + rule.Condition);
    }
}
```

Fall 2: Suchen von Bildregelgruppen im Bildregeleditor.

Mit der API `sivarc.ScreenRules.Groups` können Sie die verfügbaren Regeln und Regelgruppen im SiVARc-Bildregeleditor wie nachfolgend gezeigt finden:

```
var groups = sivarc.ScreenRules.Groups;
if (groups != null && groups.Count > 0)
{
    var rule = groups.Find("Screen rule group").Rules.Find("Screen rule_2");
    if (rule != null)
    {
        Console.WriteLine("Found Screen Rule Name: " + rule.Name);
        Console.WriteLine("Found Screen Rule Comment: " + rule.Comment);
        Console.WriteLine("Found Screen Rule Enabled: " + rule.Enabled);
        Console.WriteLine("Found Screen Rule Condition: " + rule.Condition);
    }

    var group = groups.Find("Screen rule group").Groups.Find("Screen rule group_2");
    if (group != null)
    {
        Console.WriteLine("Found Screen Rule Group Name: " + group.Name);
        Console.WriteLine("Found Screen Rule Group Comment: " + group.Comment);
        Console.WriteLine("Found Screen Rule Group Enabled: " + group.Enabled);
        Console.WriteLine("Found Screen Rule Group Condition: " + group.Condition);
    }
}
```

7.22.5 Löschen von Regeln und Regelgruppen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden. Weitere Informationen zu Verbindungen finden Sie im Benutzerhandbuch zum TIA Portal.
- Ein TIA Portal-Projekt mit Bildregeln und Bildregelgruppen ist vorhanden.

Löschen von Regeln und Regelgruppen

Um eine Regel oder Regelgruppe zu löschen, verwenden Sie die folgende API:
`sivarc.ScreenRules.Rules.Find("Screen rule_1").Delete();`

`ScreenRules` ist ein Ankerobjekt für alle Bildregelobjekte. Um eine Löschung durchzuführen, ist es obligatorisch, die Regel im Regeleditor zu finden. Weitere Informationen zum Suchen der Bildregel finden Sie im Abschnitt "Suchen von Bildregeln oder Bildregelgruppen".

Um Bilder aus einer Bildgruppe zu löschen, verwenden Sie die folgende API:

`sivarc.ScreenRules.Rules.Find("Screen rule group_1").Delete();`

7.22.6 UMAC-Einrichtung für Openness

Wissenswertes zu UMAC

Um über Openness auf UMAC zuzugreifen, stellen Sie sicher, dass Sie die erforderlichen UMAC-Anmelddaten und Zugriffsrechte haben. Wenn Sie kein gültiger UMAC-Benutzer sind, gibt die Anwendung für alle SiVArc-Ankerregelobjekte den Wert `NULL` zurück. Weitere Informationen zu UMAC finden Sie im Thema "UMAC" im SiVArc-Benutzerhandbuch.

7.22.7 SiVArc-Generierung

Voraussetzung

- Starten Sie die TIA Portal Openness-Anwendung. Weitere Informationen zu Verbindungen finden Sie im Benutzerhandbuch zum TIA Portal.
- Ein mit einem HMI-Gerät verbundenes TIA Portal-Projekt ist vorhanden und der PLC konfiguriert.

Wichtige Hinweise

- Stellen Sie sicher, dass die SiVArc-Lizenz auf Ihrem PC installiert ist, andernfalls wird während der Generierung eine Ausnahme gemeldet - "*SiVArc-Lizenz fehlt; eine SiVArc-Lizenz ist für die Änderung von Daten obligatorisch*".
- Stellen Sie sicher, dass ein gültiger Gerätename verwendet wird, andernfalls wird eine Ausnahme gemeldet - "*HMI-Gerät 'Gerätename' nicht gefunden*".
- Stellen Sie sicher, dass ein gültiger PLC-Name aufgerufen wird, andernfalls wird eine Ausnahme gemeldet - "*PLC-Gerät 'PLC-Gerätename' nicht gefunden*".
- Stellen Sie sicher, dass der Name eines unterstützten Geräts verwendet wird, andernfalls wird eine Ausnahme gemeldet - "*HMI-Gerät 'Gerätename' wird nicht unterstützt*".
- Stellen Sie sicher, dass der Name eines unterstützten PLC verwendet wird, andernfalls wird eine Ausnahme gemeldet - "*PLC-Gerät 'PLC-Gerätename' wird nicht unterstützt*".
- Stellen Sie sicher, dass Sie einen gültigen Parameter GenerationOption übergeben. Wird kein Parameter übergeben, erfolgt die SiVArc-Generierung, und standardmäßig werden dafür die TIA Portal-Projekteinstellungen verwendet.
- Stellen Sie sicher, dass Sie einen gültigen PLC-Namen verwenden, der nicht bei der vorherigen Generierung verwendet wurde, da ansonsten das System einfriert.

Generierungsoptionen - Enum (Merker)

SiVArc unterstützt Enum-Optionen, und Sie können in der API `Generate` eine Kombination aus zwei Werten übergeben. In der folgenden Tabelle werden die Enum-Optionen aufgeführt:

SN	Werte	Beschreibung
1	None	Keine Auswahl getroffen, für die Generierung werden die Standardeinstellungen übernommen
2	AllTags	Es werden alle Variablen generiert
3	UsedHmi- Tags	Es werden nur relevante (verwendete) Variablen generiert
4	FullGenerati- on	Wenn die Option FullGeneration nicht ausgewählt ist, entscheidet SiVArc intern basierend auf der Konfiguration, ob eine vollständige Generierung oder eine Delta-Generierung durchgeführt werden muss. Wenn Sie den Parameter FullGeneration übergeben, führt SiVArc eine erzwungene vollständige Generierung durch.

Um SiVArc zu generieren, verwenden Sie die folgende API:

```
sivarcs.Generate("HMI_1", new List<string> {PLC_1},  
GenerateOptions.AllTags | GenerateOptions.FullGeneration);
```

SiVArcGenerationResult und SivarcFeedbackMessage

Bei der SiVArc-Generierung wird bei erfolgreicher Generierung auf die folgenden Eigenschaften zugegriffen:

- IsGenerationSuccessful - Informiert darüber, ob die SiVArc-Generierung erfolgreich ist.
- WarningCount - Gesamtzahl von Warnungen nach der SiVArc-Generierung
- ErrorCount - Gesamtzahl der Fehler nach der SiVArc-Generierung
- Messages - Zusammensetzung der Rückmeldung

Um das SiVArc-Ergebnis zu generieren, verwenden Sie die folgende API:

- Print SiVARC generated feedback messages

```
private void WriteSiVARCGenerationResults(SiVARCGenerationResult result)
{
    sb.Append("Is SiVARC generation successful:" +
        result.IsGenerationSuccessful);
    sb.Append(Environment.NewLine);
    sb.Append("Total Warning Count:" + result.WarningCount);
    sb.Append(Environment.NewLine);
    sb.Append("Total Error Count:" + result.ErrorCount);
    sb.Append(Environment.NewLine);

    RecursivelyWriteMessages(result.Messages);
}
```

Bei der SiVARC-Generierung wird bei erfolgreicher Generierung auf die folgenden Rückmeldungen zugegriffen:

- Path: Header-Text der Rückmeldung (Header-Meldungen haben immer ein leeres Beschreibungsfeld)
- DateTime: Datum und Uhrzeit der Rückmeldung
- MessageType: Typ der Rückmeldung
- Description: Beschreibung/Inhalt der Rückmeldung (nur wenn der Pfad leer ist, um sicherzustellen, dass es sich nicht um eine Header-Meldung handelt)
- WarningCount: Anzahl der Warnungen für eine Header-Meldung
- ErrorCount: Anzahl der Fehler für eine Header-Meldung
- Messages: Zusammensetzung der Rückmeldung (SiVARCFeedbackMessage)

Mit dem folgenden Code-Snippet können Sie rekursive Rückmeldungen anzeigen:

```
private void RecursivelyWriteMessages(SiVARCFeedbackMessageComposition messages)
{
    foreach (SiVARCFeedbackMessage message in messages)
    {
        sb.Append("Path: " + message.Path);
        sb.Append(Environment.NewLine);
        sb.Append("DateTime: " + message.DateTime);
        sb.Append(Environment.NewLine);
        sb.Append("State: " + message.MessageType);
        sb.Append(Environment.NewLine);
        sb.Append("Description: " + message.Description);
        sb.Append(Environment.NewLine);
        sb.Append("Warning Count: " + message.WarningCount);
        sb.Append(Environment.NewLine);
        sb.Append("Error Count: " + message.ErrorCount);
        sb.Append(Environment.NewLine);
        sb.Append(Environment.NewLine);

        RecursivelyWriteMessages(message.Messages, indent);
    }
}
```

7.23 Openness für CP 1604/CP 1616/CP 1626

Allgemein

Mithilfe der TIA Portal Openness-Anwendung können Sie die Transferbereiche und die Zuordnungsregeln der Transferbereiche konfigurieren. Dies gilt für die Kommunikationsprozessoren CP 1604/CP 1616 ab V2.8 (und je nach Artikelnummer auch ab V2.7) sowie CP 1626 ab V1.1.

Voraussetzungen

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe "Verbindung zum TIA Portal herstellen".
- Ein Projekt ist geöffnet.
Siehe "Projekt öffnen".
- Vor dem Übersetzen des Projekts müssen alle Geräte "offline" sein.

Konfiguration von Transferbereichen

Transferbereiche erstellen

Um z. B. einen Transferbereich vom Typ "CD" für einen CP 1604 zu erstellen, verwenden Sie folgenden Programmcode:

```
NetworkInterface cpItf = CP
1604Interface.GetService<NetworkInterface>() ;
// Transferbereiche erstellen
TransferAreaComposition transferAreas = cpItf.TransferAreas;

// Einfacher Transferbereich vom Typ Eingang
TransferArea transferAreaInput =
transferAreas.Create("Input CD", TransferAreaType.CD);
```

Attribut	Beschreibung	
name	Gibt den Namen des zu erstellenden Transferbereichs an.	
type	Gibt den Typ des zu erstellenden Transferbereichs an. Folgende Typen sind möglich:	
	TransferAreaType.CD	Steuerungsgerät für Datenaustausch
	TransferAreaType.F_PS	Datenaustausch PROFIsafe
	TransferAreaType.TM	Zuordnung des Transfermoduls
	Hinweis: Es ist nicht möglich, den Typ später zu ändern.	

Attribute der Transferbereiche festlegen

Um die Attribute eines Transferbereichs festzulegen, ändern Sie z. B. folgenden Programmcode:

```

transferAreaTm.LocalToPartnerLength = 8;
transferAreaTm.Direction = TransferAreaDirection.LocalToPartner;
string name = transferAreaTm.Name

```

Einige Attribute müssen festgelegt oder abgefragt werden, alle Attribute können jedoch mit den allgemeinen Aufrufen "GetAttribute()" oder "SetAttribute()" festgelegt oder abgefragt werden. Verwenden Sie z. B. folgenden Programmcode:

```

const string myIndividualComment = "MyIndividualComment";
transferAreaTm.SetAttribute("Comment", myIndividualComment);
Int32 updateTime = transferAreaTm.GetAttribute("TransferUpdateTime")

```

Attribut	Beschreibung	
Name (Zeichenkette)	Gibt den Namen des Transferbereichs an.	
Richtung	Gibt die Richtung an, in der die Daten des Transferbereichs übertragen werden. Folgende Richtungen sind möglich:	
	TransferAreaDirection.LocalTo-Partner	Daten des Transferbereichs werden vom IO-Device an die überlagerte Steuerung übertragen.
	TransferAreaDirection.partnerTo-Local	Daten des Transferbereichs werden von der überlagerten Steuerung an das IO-Device übertragen.
	TransferAreaDirection.bidirectional	Daten des Transferbereichs können zwischen der überlagerten Steuerung und dem IO-Device in beiden Richtungen übertragen werden. Mit dem Attribut "LocalToPartnerLength" wird die Länge angegeben, mit der Daten des Transferbereichs vom IO-Device an die überlagerte Steuerung übertragen werden. Das Attribut "PartnerToLocalLength" legt die Datenmenge fest, die von der überlagerten Steuerung an das IO-Device übertragen werden kann.
Comment (Zeichenkette)	Textfeld für Kommentar zum Transferbereich.	
LocalToPartnerLength	Gibt die Länge der Daten des Transferbereichs an, die vom IO-Device an die überlagerte Steuerung übertragen werden.	
PartnerToLocalLength	Gibt die Länge der Daten des Transferbereichs an, die von der überlagerten Steuerung an das IO-Device übertragen werden.	
LocalAdresses	Gibt die Eingangs- und Ausgangsadressen des Transferbereichs im lokalen Gerät an.	
PartnerAdresses	Gibt die Eingangs- und Ausgangsadressen des Transferbereichs in der überlagerten Steuerung an.	
TransferUpdateTime(Int32)	Gibt die Aktualisierungszeit des Transferbereichs an. Nur bei einem Transferbereich vom Typ "TransferAreaType.TM" festgelegt oder abgefragt.	
PositionNumber	Gibt die Anzahl der virtuellen Submodule dieses Transferbereichs an.	
Typ	Gibt den Typ des Transferbereichs an, schreibgeschützt.	
TransferAreaMappingRules	Gibt die Routing-Tabelle des Routing-Bereichs an, schreibgeschützt.	

Transferbereiche löschen

Verwenden Sie zum Löschen von Transferbereichen folgenden Programmcode:

```
transferAreaInput.Delete();
```

Iteration über Transferbereiche

Verwenden Sie zum Iterieren von Transferbereichen folgenden Programmcode:

```
TransferAreaComposition transferAreas = cpItf.TransferAreas;
foreach (TransferArea transferArea in transferAreas)
{
    transferArea.Delete();
}
```

Konfiguration des E/A-Routing

E/A-Routen erstellen

Verwenden Sie zum Erstellen von E/A-Routen folgenden Programmcode:

```
// TransferAreaMappingRule erstellen
TransferAreaMappingRuleComposition routingTable
    = transferArea.TransferAreaMappingRules;

// Neue E/A-Route erstellen
TransferAreaMappingRule route1 =
    routingTable.Create();
```

Attribute von E/A-Routen festlegen

Die folgenden Attribute können für das E/A-Routing festgelegt werden:

Attribut	Beschreibung
Offset	In Bit angegebener Offset im Routing-Bereich, dem die Daten zugewiesen werden sollen. Die Länge des Offsets laut Festlegung durch die Attribute "Begin" und "End".
Ziel	Gibt das Modul oder Submodul des IO-Device an, das die Daten enthält, die der Konfiguration des IO-Device zugeordnet werden sollen, d. h. einen Transferbereich vom Typ "TM".
IoType	Das Attribut "IoType" kann nur in einem Transferbereich vom Typ "Input" geändert werden. Zusätzlich muss ein gemischtes Modul für diesen Transferbereich als "Ziel" konfiguriert werden. Erst kann können Sie auswählen, ob die Daten der Eingänge (IoType.Input) zu lesen sind oder ob die Daten der Ausgänge (IoType.Output) (zurück-)gelesen werden sollen.
Begin	Gibt den Anfang der mit dem Attribut "Target" zu lesenden Daten an.
End	Gibt das Ende der mit dem Attribut "Target" zu lesenden Daten an.

E/A-Routen löschen

Verwenden Sie zum Löschen von E/A-Routen folgenden Programmcode:

```
transferAreaMappingRule.Delete();
```

Iteration über E/A-Routing

Für eine Iteration über E/A-Routing verwenden Sie folgenden Programmcode:

```
TransferAreaMappingRuleComposition routingTable =
    transferArea.TransferAreaMappingRules;

foreach (TransferAreaMappingRule route in routingTable)
{
    route.Delete();
}
```

7.24 Openness-Transferbereiche für PN/PN-Koppler

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts
- Vor dem Übersetzen des Projekts müssen alle Geräte "offline" sein.

Einleitung

Mit der Anwendung TIA Portal Openness können Sie Module/Submodule hinzufügen sowie Transferbereiche für PN/PN-Koppler löschen und suchen.

Konfiguration von Transferbereichen

Transferbereiche erstellen

Um beispielsweise ein Modul an der nächsten freien Position zu erstellen, verwenden Sie folgenden Programmcode:

```
NetworkInterface coupler_Itf = deviceItem.GetService<NetworkInterface>();
TransferAreaComposition transferAreas = coupler_Itf.TransferAreas;
// Create a transfer area of type input at next free positionnumber
TransferArea transferAreaInput = transferAreas.Create("Input", TransferAreaType.IN);
```

Hinweis

Es ist nicht möglich, ein Submodul ohne Positionsnummer zu erstellen.

Um beispielsweise ein Modul an einer Positionsnummer und ein Submodul an der nächsten freien Submodul-Positionsnummer zu erstellen, verwenden Sie folgenden Programmcode:

```
// Create a transfer area of type input at positionnumber 5  
TransferArea transferAreaInput = transferAreas.Create("Input", TransferAreaType.IN, 5);
```

7.24 Openness-Transferbereiche für PN/PN-Koppler

Attribut	Beschreibung	
Type		Gibt den Typ des zu erstellenden Transferbereichs an. Folgende Typen sind möglich:
	TransferAreaType.None	Standardwert
	TransferAreaType.IN	Eingang des Transferbereichs
	TransferAreaType.OUT	Ausgang des Transferbereichs
	TransferAreaType.MSI	Gemeinsamer Eingang des Transferbereichs (MSI)
	TransferAreaType.MSO	Gemeinsamer Ausgang des Transferbereichs (MSO)
	TransferAreaType.MSO_LOCAL	Gemeinsamer lokaler Ausgang des Transferbereichs (MSO_LOCAL)
	TransferAreaType.RECORD_WRITE_STO	Transferbereich für Record Data Write, bis zu acht Datensätze werden gepuffert
	TransferAreaType.RECORD_WRITE_PUB	Transferbereich für Record Data Write überschreibt vorherigen Datensatz
	TransferAreaType.RECORD_READ_STO	Transferbereich für Record Data Write liest den ältesten Datensatz im Puffer
	TransferAreaType.RECORD_READ_PUB	Transferbereich für Record Data Write liest den zuletzt geschriebenen Datensatz
	TransferAreaType.MSI_MSO	Gemeinsamer Eingang des Transferbereichs (MSI) / Gemeinsamer Ausgang des Transferbereichs (MSO)
	TransferAreaType.IN_OUT	Eingang des Transferbereichs / Ausgang des Transferbereichs
	TransferAreaType.LOCAL_RECORD_STO	Transferbereich für lokale Datensatzkopplung, bis zu acht Datensätze werden gepuffert
	TransferAreaType.LOCAL_RECORD_PUB	Transferbereich für lokale Datensatzkopplung, vorheriger Datensatz wird überschrieben
	TransferAreaType.SUB_MSI	Transferbereich für gemeinsamen Eingang (MSI) kopieren
	TransferAreaType.SUB_MSO	Transferbereich für gemeinsamen Ausgang (MSO) kopieren
	TransferAreaType.SUB_LOCAL_RECORD_STO_READ	Transferbereich für Record Data Read zum Lesen des ältesten Datensatzes im Puffer
	TransferAreaType.SUB_LOCAL_RECORD_PUB_READ	Transferbereich für Record Data Read zum Lesen des zuletzt geschriebenen Datensatzes
	TransferAreaType.PROFISA-FE_IN12_OUT6	Transferbereich mit 12-Byte-Eingang und 6-Byte-Ausgang
	TransferAreaType.PROFISA-FE_IN6_OUT12	Transferbereich mit 6-Byte-Eingang und 12-Byte-Ausgang

Attribute der Transferbereiche festlegen

Um die Attribute eines Transferbereichs festzulegen, ändern Sie z. B. folgenden Programmcode:

```
// Change input length
CouplerTransf.LocalToPartnerLength = 5;
// Change output length
CouplerTransf.PartnerToLocalLength = 5;
// Change name
CouplerTransf.Name = "Testname";
```

Einige Attribute müssen festgelegt oder abgefragt werden, alle Attribute können jedoch mit den allgemeinen Aufrufen "TransferArea.GetAttribute()" oder "SetAttribute()" festgelegt oder abgefragt werden. Verwenden Sie z. B. folgenden Programmcode:

```
// Set Comment of Transferarea
string myIndividualComment = "MyIndividualComment";
CouplerTransf.SetAttribute("Comment", myIndividualComment);
// Get positionNumber of Transferarea
Int32 positionNumber = (Int32)CouplerTransf.GetAttribute("PositionNumber");
```

Attribut	Beschreibung	
Name (string)	Gibt den Namen des Transferbereichs an.	
Direction	Gibt die Richtung an, in der die Daten des Transferbereichs übertragen werden. Folgende Richtungen sind möglich:	
	TransferAreaDirection.LocalTo- Partner	Daten des Transferbereichs werden vom IO-Device an die überlagerte Steuerung übertragen.
	TransferAreaDirection.partnerTo- Local	Daten des Transferbereichs werden von der überlagerten Steuerung an das IO-Device übertragen.
	TransferAreaDirection.bidirectional	Daten des Transferbereichs können zwischen der überlagerten Steuerung und dem IO-Device in beiden Richtungen übertragen werden. Mit dem Attribut "LocalToPartnerLength" wird die Länge angegeben, mit der Daten des Transferbereichs vom IO-Device an die überlagerte Steuerung übertragen werden. Das Attribut "PartnerToLocal- Length" legt die Datenmenge fest, die von der überlagerten Steuerung an das IO-Device übertragen werden kann.
Comment (string)	Textfeld für Kommentar zum Transferbereich	
LocalToPartnerLength	Gibt die Eingangsdatenlänge des Transferbereichs an	
PartnerToLocalLength	Gibt die Ausgangsdatenlänge des Transferbereichs an.	
PositionNumber	Gibt die Steckplatznummer des Transferbereichs an.	
ExtendedPositionNumber	Gibt die Untersteckplatznummer des Transferbereichs an	
Type	Gibt den Typ des Transferbereichs an	
SharedDeviceAccessConfigured	Steuert Zugriff auf PLC	
UpdateAlarm	Gibt den Aktualisierungsalarm des Transferbereichs an	
RecordIndex	Legt die Datensatznummer fest, die Sie beim Schreiben des Datensatzes verwenden	

Transferbereiche löschen

Verwenden Sie zum Löschen von Transferbereichen folgenden Programmcode:

```
TransferArea TransferAreaInput = transferAreas.Create("Input", TransferAreaType.IN);  
TransferAreaInput.Delete();
```

Transferbereiche suchen

Um den Transferbereich mit Positionsnummer und erweiterter Positionsnummer abzurufen, verwenden Sie folgenden Programmcode:

```
// Find a transfer area with Positionnumber and ExtendedPositionNumber  
TransferArea transferAreaFind = transferAreas.Find(4, 3);
```

Hinweis

Die erweiterte Positionsnummer wird nur für Transferbereiche mit mehreren Submodulen wie MSI benötigt.

Um den ersten Transferbereich mit dieser Positionsnummer abzurufen, verwenden Sie folgenden Programmcode:

```
// Find a transfer area with Positionnumber  
TransferArea transferAreaFind = transferAreas.Find(5);
```

Hinweis

Wenn eines dieser Attribute nicht verfügbar ist, kommt es zu einer Ausnahme.

7.25 Virtuelle Openness-Module/Submodule für ET 200SP PN HF

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts

Einleitung

Sie können mit TIA Portal Openness virtuelle Module/Submodule für ET 200 SP PN HF hinzufügen und löschen:

Die folgenden Eigenschaften werden bei virtuellen Modulen unterstützt:

Name der Eigenschaft	Datentyp	EomAtomValue	EomAtom-Beschreibung
SharedDeviceAccessConfigured	Boolean		
Name	String		
PositionNumber	Int32		
VirtualType	UInt64	1	MsoLocal
AddSubModules	Int64		

Die folgenden Eigenschaften werden bei virtuellen Submodulen unterstützt:

Eigenschaft	Datentyp	EomAtomValue	EomAtom-Beschreibung
SharedDeviceAccessConfigured	Boolean		
Comment	String		
PositionNumber	Int32		
Name	String		
ActivateDataStatus	Boolean		
InputAddressLength	Int64		
OutputAddressLength	Int64		
VirtualSubType	UInt64	1	MsoLocal
		2	MsoSub

Programmcode: Virtuelles Modul erstellen und löschen

Ändern Sie folgenden Programmcode, um ein neues Modul hinzuzufügen:

```
string Type = "OrderNumber:6ES7 155-6AU30-0CN0/V4.2";
Device ET200SP = newProject.Devices.CreateWithItem(Type, "ET200SP",
"ET200SP");
DeviceItem Rack = ET200SP.DeviceItems.First();
string TypeIdentifier = "OrderNumber:ET200SP.Virtual.Module";
string Name = "VirtualIO_1";
int PositionNumber = 100;
DeviceItem VIM = Rack.PlugNew(TypeIdentifier, Name, PositionNumber);
```

Ändern Sie folgenden Programmcode, um ein Modul zu löschen:

```
DeviceItem Rack = ET200SP.DeviceItems.First();
string TypeIdentifier = "OrderNumber:ET200SP.Virtual.Module";
string Name = "VirtualIO_1";
int PositionNumber = 100;
DeviceItem VIM = Rack.PlugNew(TypeIdentifier, Name, PositionNumber);
VIM.Delete();
```

Programmcode: Virtuelles Submodul erstellen und löschen

Ändern Sie folgenden Programmcode, um in Openness virtuelle Submodule zu erstellen:

```
VIM1.SetAttribute ("AddSubModules", (Int64)1);  
VIM2.SetAttribute ("AddSubModules", (Int64)2);
```

Hinweis

Um ein Submodul hinzuzufügen, müssen Sie die Moduleigenschaft "AddSubModules" auf die Anzahl der hinzugefügten Submodule setzen. Wenn Sie versuchen, mehr Submodule als zulässig hinzuzufügen, wird eine Ausnahme ausgegeben.

Ändern Sie folgenden Programmcode, um ein Submodul zu löschen:

```
SubModule3.Delete();
```

Hinweis

Die ersten zwei Submodule können Sie nicht löschen.

7.26 Funktionen für SINUMERIK

7.26.1 Einführung

Über TIA Portal-Openess automatisieren Sie das Engineering und steuern das TIA Portal über ein von Ihnen selbst erstelltes Programm.

Für dieses selbst erstellte Programm finden Sie in dieser Hilfe viele Informationen und Beispielcodes. Auch für die TIA Portal-Anwendung "SINUMERIK" können Sie eigene Programme zusammenstellen und anwenden.

Weitere Informationen

Bevor Sie für SINUMERIK aus den nachfolgend gelisteten Beispielcodes ein eigenes Programm zusammenstellen, beachten Sie bitte die übergreifenden Informationen zu Openness, die Sie unter folgenden Stichworten in dieser Hilfe finden:

- Voraussetzungen für TIA Portal Openness
- TIA Portal Openness installieren
- Auf das TIA Portal zugreifen
- TIA Portal Openness-Objektmodell
- Programmierschritte

7.26.2 Typkennung- Kennzeichnung der Komponenten

Jede SINUMERIK-Komponente besitzt eine eindeutige Nummer, die als Typkennung (TypeIdentifier) bezeichnet wird. Im Openness-Programmcode können Sie die Typkennung nutzen, um eine Komponente eindeutig zu identifizieren und zu benennen. Beispielsweise lautet die Typkennung für SINUMERIK 840D sl NCU 710.3 PN "Artikel-Nr. :6FC5 371-0AA30-0Axx/Vy.z".

Die Typkennung sehen Sie beim Anlegen Ihres Geräts im Dialog "Neues Gerät hinzufügen" sowie in der tabellarischen Geräteübersicht.

Sie können die Typkennung in Ihre Openness-Anwendung kopieren.

Die Typkennung wird in Openness als Aktualparameter beim Aufruf einer Methode, z. B. der CreateWithItem()-Methode, erwartet.

Anzeige des TypeIdentifiers in SINUMERIK aktivieren

1. Wählen Sie in der Projektansicht das Menü "Extras > Einstellungen".
Der Konfigurationsbereich "Einstellungen" wird geöffnet.
2. Wählen Sie in der Sekundärnavigation den Eintrag "Hardware-Konfiguration".
3. Aktivieren Sie die Option "Anzeige des Type Identifiers für Geräte und Module aktivieren".
Die Anzeige des TypeIdentifiers ist nun aktiv.

7.26.3 Grundlagen

Übersicht

Folgende SINUMERIK-Geräte können Sie in TIA Portal Openness anlegen:

- NCU
- NX-Baugruppe
- ADI4

Zum Anlegen der SINUMERIK-Geräte verwenden Sie die Methode `CreateWithItem()` der `Devices Collection`.

Hinweis

Alle integrierten Subkomponenten einer SINUMERIK-NCU wie PLC, NCK, CP, HMI und SINAMICS Integrated, werden unterlagert auf der gleichen Ebene automatisch angelegt.

Besonderheiten beim Anlegen der SINUMERIK-Geräte

Die Namen der Baugruppenträger entsprechen den NCU-Typen und sind in der Oberfläche schreibgeschützt. Die Bezeichnungen der Baugruppenträger müssen Sie auch in TIA Portal Openness verwenden.

7.26 Funktionen für SINUMERIK

Verwenden Sie die folgenden Standardnamen der Baugruppenträger für die Erstellung eines Geräts über eine Geräteelement-Typkennung mit der `Device CreateWithItem`-Methode:

SINUMERIK NCU	Baugruppenträger-Standardname
SINUMERIK 840D sl NCU 710.3 PN	NCU 710.3 PN
SINUMERIK 840D sl NCU 720.3 PN	NCU 720.3 PN
SINUMERIK 840D sl NCU 730.3 PN	NCU 730.3 PN
SINUMERIK 840D sl NCU 730.3 PN /319	NCU 730.3 PN /319

Hinweis

Alternativ können Sie die Parameternamen weglassen. Wenn der Name "Null" oder "String.Empty" ist, wird der Standardname verwendet.

Folgende Parameter können mit der Methode `CreateWithItem()` verwendet werden:

- default name,
z. B.
`project.Devices.CreateWithItem("OrderNumber:6FC5 371-0AA30-0AAB/V4.8", "NCU 710.3 PN", "TestDevice");`
- null,
z. B.
`project.Devices.CreateWithItem("OrderNumber:6FC5 371-0AA30-0AAB/V4.8", null, "TestDevice");`
- string.Empty,
z. B.
`project.Devices.CreateWithItem("OrderNumber:6FC5 371-0AA30-0AAB/V4.8", string.Empty, "TestDevice");`

Weitere Informationen zu Aufrufparametern der Methode `CreateWithItem()` finden Sie im Kapitel "Erstellen eines Geräts".

Die folgende Tabelle zeigt die Zuordnung der Geräte und ihrer Typkennungen:

SINUMERIK NCU	Typkennung
SINUMERIK 840D sl NCU 710.3 PN	Artikel-Nr.:6FC5 371-0AA30-0Axx/Vy.z
SINUMERIK 840D sl NCU 720.3 PN	Artikel-Nr.:6FC5 372-0AA30-0Axx/Vy.z
SINUMERIK 840D sl NCU 730.3 PN	Artikel-Nr.:6FC5 373-0AA30-0Axx/Vy.z
SINUMERIK 840D sl NCU 730.3 PN /319	Artikel-Nr.:6FC5 373-0AA31-0Axx/Vy.z

Beim Anlegen der SINUMERIK-Geräte sind Platzhalter in der Typkennung zulässig. Die Platzhalter tauschen Sie später gegen passende gerätespezifische Zeichen aus. Wenn Sie z. B. die Typkennung "OrderNumber:6FC5 372-0AA30-0AA0/V4.8" eingeben, wird SINUMERIK 840D sl NCU 720.3 PN mit der Firmware-Version FW4.8 angelegt.

Klassifikation des Geräteelements

Jedes Gerät oder Geräteelement besitzt obligatorische Attribute, die gelesen und geschrieben werden. Weitere Informationen zu den Attributen, die in Openness unterstützt werden, finden Sie im Kapitel "Funktionen auf Geräteelementen".

Die Klassifikationsattribute des Geräteelements sind schreibgeschützt und in der Benutzeroberfläche des TIA Portals nicht sichtbar.

Die Klassifikationsattribute haben den Wert "DeviceItemClassification":

Wert des Klassifikationsattributs	Beschreibung
DeviceItemClassifications.None (0)	Keine Klassifikation.
DeviceItemClassifications.CPU (1)	Das Geräteelement ist eine CPU.
DeviceItemClassifications.HM (2)	Das Geräteelement ist ein Kopfmodul.

Wenn Sie den Wert des Geräteelements über die integrierte PLC abfragen, ist es für ein SINUMERIK-Gerät der Wert "CPU (1)".

Als Kopfmodule fungieren im Openness-Objektmodell die folgenden Komponenten:

- SINAMICS Integrated
- NX-Baugruppe
- ADI4

In allen anderen Fällen ist der Wert des Klassifikationsattributs "DeviceItemClassification" "None" (0).

Geräteelemente über die Eigenschaft "Kopfmodul" finden

Die folgenden Beispiele zeigen, wie Sie die Geräteelemente mit der Eigenschaft "Kopfmodul" finden, bevor Sie z. B. ein Telegramm konfigurieren.

Sie können diese Eigenschaft nur lesen, aber nicht an Geräteelemente setzen.

Geräteelemente über die Eigenschaft "Kopfmodul" finden

```
foreach (Device device in project.Devices)
{
    foreach (DeviceItem deviceItem in device.DeviceItems)
    {
        if (deviceItem.Classification == DeviceItemClassifications.HM)
        {
            var driveObjectContainer = deviceItem.GetService<DriveObjectContainer>();
            // do something
        }
    }
}
```

Alternativ verwenden Sie das folgende Code-Beispiel, um eine PLC unabhängig von konkreter Realisierung (integrierte SINUMERIK PLC, SIMATIC PLC, Software PLC im PC) anhand der Eigenschaft "CPU" zu finden:

PLCs finden

```
foreach (Device device in project.Devices)
{
    DeviceItem rack = device.DeviceItem[0]; //additional necessary step in SINUMERIK side
    foreach (DeviceItem deviceItem in rack.DeviceItems)
    {
        if (deviceItem.Classification == DeviceItemClassifications.CPU)
        {
            var softwareContainer = deviceItem.GetService<SoftwareContainer>();
            // do something
        }
    }
}
```

Siehe auch

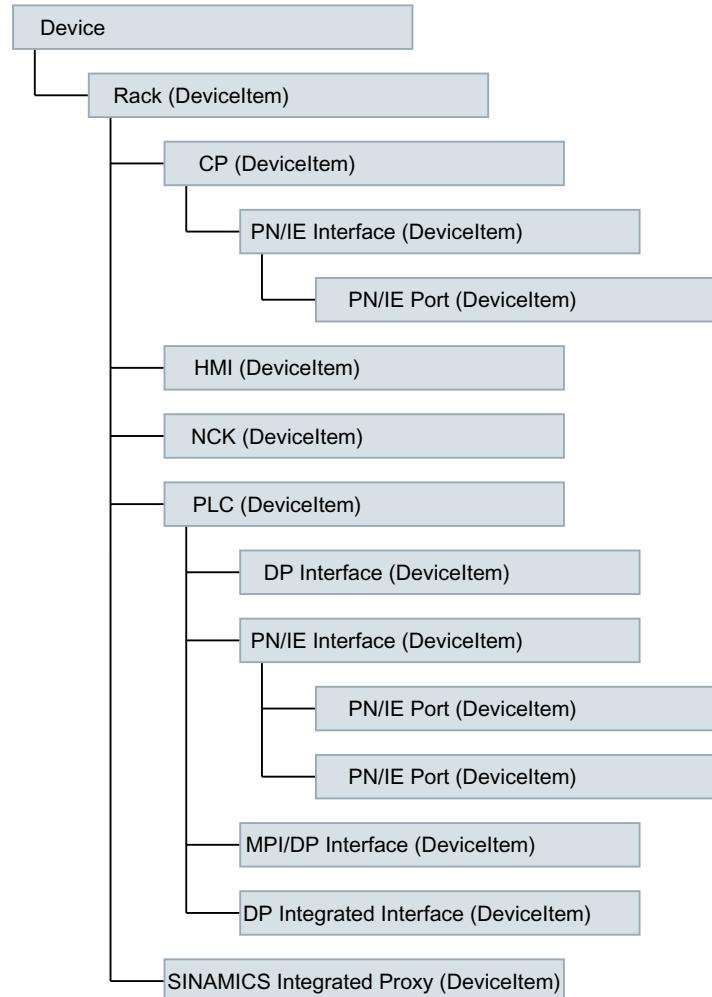
[NX-Baugruppe anlegen \(Seite 665\)](#)

[NCU anlegen \(Seite 665\)](#)

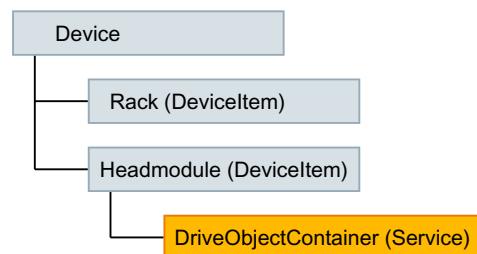
7.26.4 Objektmodell

Übersicht

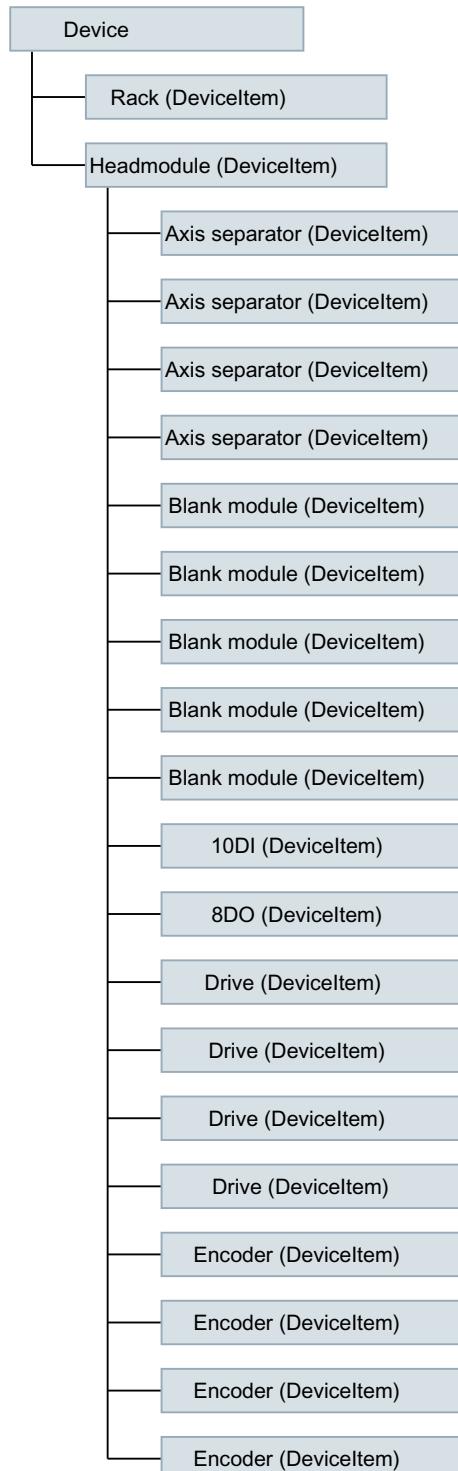
Das folgende Diagramm beschreibt die Objekte, die sich bei SINUMERIK 840D sl unter "Device" befinden:



Das folgende Diagramm beschreibt die Objekte, die sich bei SINUMERIK NX 10.3/15.3 unter "Device" befinden:



Das folgende Diagramm beschreibt die Objekte, die sich bei ADI4 unter "Device" befinden:



Weitere Informationen zu TIA Portal Openness-Objektmodell, finden Sie unter "TIA Portal Openness API".

7.26.5 Referenz

7.26.5.1 DriveObject

DriveObject

Die `DriveObject` Klasse ermöglicht den Zugriff auf das Antriebsobjekt. Über das Antriebsobjekt ist z. B. der Zugriff auf das Telegramm möglich.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Zugriffsart	Zugriff	Beschreibung
Telegrams	TelegramComposition	read	-	Gibt eine Liste mit den verfügbaren Telegrammen des Antriebsobjekts zurück. Die Liste kann mit der <code>TelegramComposition</code> Klasse verändert werden.
Parent	IEngineeringObject	read	-	Gibt die Referenz auf die übergeordnete Klasse (<code>DriveObjectContainer</code>) zurück.

Die folgende Tabelle beschreibt die **Methoden** der Klasse:

Name	Beschreibung
GetAttribute	Greift lesend auf ein Attribut eines Antriebobjekts zu
SetAttribute	Greift schreibend auf ein Attribut eines Antriebobjekts zu
GetEnumerator	Ermöglicht die Iteration über die gegebene Menge der Elemente

7.26.5.2 DriveObjectContainer

DriveObjectContainer

Der `DriveObjectContainer` ist ein Service des Antriebsobjekts (`DeviceItem`) vom aktuellen Gerät (`Device`).

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Navigatoren** des DriveObjectContainer:

Name	Datentyp	Zugriffsart	Zugriff	Beschreibung
DriveObjects	DriveObjectComposition	read	-	Gibt eine Liste mit den verfügbaren Antriebsobjekten zurück. Die Antriebsobjekte ermöglichen den Zugriff auf Antriebsparameter und Telegramme.
Parent	IEngineeringObject	read	-	Gibt die Referenz auf die übergeordnete Klasse (DeviceItem) zurück.

7.26.5.3 DriveObjectComposition

DriveObjectComposition

Die DriveObjectComposition Klasse ermöglicht den Zugriff auf verfügbare Telegramme eines Antriebobjekts.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Zugriffsart	Zugriff	Beschreibung
IndexOf	int32	read	-	Gibt den Index in der Menge der Elemente für die angefragte Instanz zurück.
Contains	bool	read	-	Ermittelt, ob die angegebene Instanz in der Menge der Elemente enthalten ist. TRUE: Der Container enthält die Instanz FALSE: Der Container enthält die Instanz nicht
GetEnumerator	IEnumerator<DriveObject>	read	-	Ermöglicht die Iteration über die gegebene Menge der Elemente

7.26.5.4 AddressComposition

AddressComposition

Die AddressComposition Klasse repräsentiert die Adresse eines Telegramms.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Zugriffsart	Zugriff	Beschreibung
IoType	enum: AddressIoType	read	-	Gibt Informationen zum Typ der Adresse zurück.
Context	enum: AddressContext	read	Zugriff nur über GetAttribute oder SetAttribute möglich	Gibt Informationen zum Kontext der Adresse zurück.
StartAddress	Int32	read/write	-	Gibt die Startadresse des Telegramms zurück oder legt die Startadresse fest.
Length	Int32	read	-	Gibt die Länge des Telegramms zurück.
IndexOf	int32	read	-	Gibt den Index in der Menge der Elemente für die angefragte Instanz zurück.
Contains	bool	read	-	Ermittelt, ob die angegebene Instanz in der Menge der Elemente enthalten ist. TRUE: Der Container enthält die Instanz FALSE: Der Container enthält die Instanz nicht
GetEnumerator	IEnumerator<DriveObject>	read	-	Ermöglicht die Iteration über die gegebene Menge der Elemente

Hinweis

Wenn Sie über Address.Parent zurück zum Telegramm navigieren möchten, wird statt des Namespace `Siemens.Engineering.MC.Drives.Telegram` der Namespace `Siemens.Engineering.MC.DriveConfiguration.Telegram` aufgerufen.

Weitere Informationen zu TIA Portal Openness-Bibliotheken, finden Sie unter "Standard-Bibliotheken".

Siehe auch

[AddressIoType \(Seite 664\)](#)

[AddressContext \(Seite 664\)](#)

7.26.5.5 AddressContext

AddressContext

Das Enum AddressContext enthält Informationen zum Kontext der Adresse.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Enum-Einträge**:

Name	Beschreibung
AddressContext.None	Für die Adresse wurde kein Kontext gefunden
AddressContext.Device	Kontext ist eine Geräteadresse
AddressContext.Head	Kontext ist eine Kopfadresse

Weitere Informationen zu TIA Portal Openness-Bibliotheken, finden Sie unter "Standard-Bibliotheken".

7.26.5.6 AddressIoType

AddressIoType

Das Enum AddressIoType enthält Informationen zum Typ der Adresse.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Enum-Einträge**:

Name	Beschreibung
AddressIoType.None	Der IO-Typ ist nicht anwendbar
AddressIoType.Input	Typ ist eine Eingangsadresse
AddressIoType.Output	Typ ist eine Ausgangsadresse
AddressIoType.Diagnosis	Typ ist eine Diagnoseadresse
AddressIoType.Substitute	Typ ist eine Ersatzadresse

Weitere Informationen zu TIA Portal Openness-Bibliotheken, finden Sie unter "Standard-Bibliotheken".

7.26.6 Codebeispiel

7.26.6.1 Allgemein

Die folgenden Codebeispiele beschreiben die grundsätzliche Vorgehensweise für verschiedene Anwendungsfälle. Der Code ist nicht zwingend vollständig und kompilierbar.

7.26.6.2 Erste Schritte in SINUMERIK durchführen

- Verbinden Sie die TIA Portal Openness-Anwendung mit dem TIA Portal.
- Öffnen Sie Ihr Projekt.

Das folgende Beispiel zeigt, wie Sie ermitteln können, welche SINUMERIK Toolbox-Version installiert ist.

SINUMERIK Toolbox-Version ermitteln

```
using Siemens.Engineering;
if (tiaProcess.InstalledSoftware.Any(sw => sw.Name.Equals("SINUMERIK Toolbox")) &&
sw.Version.Equals("V16")))
{
    Console.WriteLine("SINUMERIK Toolbox is available");
}
// "V16" is the current SINUMERIK version started at December 2019.
```

7.26.6.3 NCU anlegen

Mit der Methode `CreateWithItem()` der `Devices Collection` legen Sie eine SINUMERIK NCU an. Die SINUMERIK NCUs werden über Parameter der Methode spezifiziert. Das Format der Parameter ist nachfolgend beschrieben.

NCU anlegen

Format der Parameter für SINUMERIK NCU:

```
CreateWithItem(@"OrderNumber:mlfb/
FirmwareVersion/", "NameOfTheDevice", positionNumber)
```

Der Parameter `positionNumber` ist optional.

Das folgende Beispiel zeigt, wie Sie eine Steuerung SINUMERIK 840D sl NCU 720.3 PN anlegen.

SINUMERIK 840D sl NCU 720.3 PN anlegen

```
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
Project tiaproject= portal.Projects.Open(...); //The path of the project

Device NCUDevice = tiaproject.Devices.CreateWithItem(@"OrderNumber:6FC5
372-0AA30-0AA0/4.8/", "NCU 720.3 PN", string.Empty, "TestDevice");
```

7.26.6.4 NX-Baugruppe anlegen

Mit der `Device CreateWithItem`-Methode legen Sie eine NX-Baugruppe an. Anschließend verbinden Sie die NX-Baugruppe mit einer NCU.

Folgende Typkennungen werden für SINUMERIK NX-Baugruppen verwendet:

SINUMERIK NX-Baugruppe	Typkennung
SINUMERIK NX10.3	OrderNumber:6SL3 040-1NC00-0Axx/Vy.z
SINUMERIK NX15.3	OrderNumber:6SL3 040-1NB00-0Axx/Vy.z

Das folgende Beispiel zeigt, wie Sie eine SINUMERIK NX-Baugruppe anlegen.

NX-Baugruppe anlegen

```
project.Devices.CreateWithItem("OrderNumber:6SL3040-1NC00-0AA0/
V5.1", "MyNXDevice", "TestDevice");
```

Versionskompatibilität

Die Firmware-Version der NX-Baugruppe muss mit der Firmware-Version von SINAMICS Integrated übereinstimmen und mit der Firmware-Version der NCU kompatibel sein.

Die von SINAMICS Integrated abweichende NX Firmware-Version kann über Openness nicht zugeordnet werden.

Die folgende Tabelle zeigt die Versionskompatibilität für 840D sl:

NCU-Firmware (840D sl)	SINAMICS Integrated/NX-Firmware
V4.5	V4.5
V4.7	V4.7
V4.8	V5.1
V4.91	V5.2
V4.92	
V4.93	

7.26.6.5 NX-Baugruppe mit NCU verbinden

NX-Baugruppe mit einer NCU verbinden

Um eine NX-Baugruppe mit einer NCU über den Subnetztyp "ProfibusIntegrated" zu verbinden, muss der Dienst "NetworkInterface" über das Kopfmodul von NX geladen werden.

Das folgende Beispiel zeigt, wie Sie den Dienst "NetworkInterface" laden.

NetworkInterface-Dienst laden

```
foreach (Device device in project.Devices)
{
    foreach (DeviceItem deviceItem in device.DeviceItems)
    {
        if (deviceItem.Classification == DeviceItemClassifications.HM)
        {
            var networkInterface = deviceItem.GetService<NetworkInterface>();
            // do something
        }
    }
}
```

Das folgende Beispiel zeigt, wie Sie eine NX-Baugruppe über "ProfibusIntegrated" mit einer NCU verbinden:

NX-Baugruppe über ProfibusIntegrated verbinden

```
Subnet pbiSubnet = ...;
Node node = networkInterface.Nodes.FirstOrDefault();
node.ConnectToSubnet(pbiSubnet);
```

Die DP-Adresse wird in Openness über DRIVE-CLiQ Port der fest NX zugeordnet. Jedes Port-Label verfügt über eine feste DP-Integrated-Adresse.

Die DP-Adresse muss vor dem Verbinden mit der NCU der NX zugeordnet werden.

7.26.6.6 Archive erstellen

Archive für Serieninbetriebnahme erstellen

Mit TIA Portal-Openness erstellen und exportieren Sie SINUMERIK-Archive, um z. B. die Serieninbetriebnahme zu vereinfachen.

Hinweis

Während der Erstellung der Archive muss sich die PLC im Offline-Modus befinden. Der Safety-Modus darf nicht aktiv sein.

Sie erstellen SINUMERIK-Archive über die TIA Portal-Projekteigenschaft `HwUtilities` mit dem Service `SinumerikArchiveProvider`.

Das folgende Beispiel zeigt, wie Sie den Service `SinumerikArchiveProvider` aufrufen:

SinumerikArchiveProvider aufrufen

```
Project project = ...;
SinumerikArchiveProvider archiveProvider =
project.HwUtilities.Find("SinumerikArchiveProvider") as
SinumerikArchiveProvider;

if (archiveProvider != null)
{
// Work with the provider
}
```

Das folgende Beispiel zeigt, wie Sie ein PLC-Archiv erstellen, das die Hardware-Informationen und alle Datenbausteine enthält:

PLC-Archiv erstellen

```
...
Siemens.Engineering.HW.DeviceItem plc = ...;

try {
    // The file extension is required
    string archivePath = string.Format(@"D:\some_path\{0}.dsf", plc.Name);

    // Comment and author arguments are optional
    archiveProvider.Archive(plc, new FileInfo(archivePath),
    SinumerikArchivationMode.HardwareAndAllProgramBlocks[, "Comment", "Author
    name"]);
}
catch (EngineeringTargetInvocationException ex)
{
    // Handle archive failure
}
```

Das folgende Beispiel zeigt, wie Sie die Software-Anteile eines bereits erstellten PLC-Archivs aktualisieren:

Archivanteile aktualisieren

```
...
Siemens.Engineering.HW.DeviceItem plc_1 = ...;
Siemens.Engineering.HW.DeviceItem plc_1_copy = ...;
try {
    // The file extension is required
    string archivePath = @"D:\some_path\SinumerikArchive.dsf";

    // Create a Sinumerik archive with HardwareAndAllProgramBlocks
    archiveProvider.Export(plc_1, new FileInfo(archivePath),
    SinumerikArchivationMode.HardwareAndAllProgramBlocks);

    // Update the software part in the previously created archive using
    UpdateProgramBlocksOfArchive method
    archiveProvider.UpdateProgramBlocksOfArchive(plc_1_copy, new
    FileInfo(archivePath));
}
catch (EngineeringException ex)
{
    // Handle export failure
}
```

7.26.6.7 Safety Integrated aktivieren

Safety Integrated aktivieren

Mit TIA Portal Openness können Sie Safety Integrated (F-PLC) in den Eigenschaften der NCU aktivieren.

Hinweis

Auswirkungen auf Telegrammkonfiguration

Der verwendete Safety Integrated-Modus hat Auswirkungen auf die Telegrammkonfiguration, da im Modus Safety Integrated plus (F-PLC) andere Telegramme verwendet werden, als bei inaktivem Safety Integrated-Modus.

Hinzugefügte oder geänderte Telegramme bleiben jedoch erhalten, sofern diese mit dem neu gewählten Safety Integrated-Modus kompatibel sind.

Stellen Sie ggf. nach der Modusänderung in der Telegrammkonfiguration sicher, dass etwaige Anpassungen noch vorhanden sind.

Sie aktivieren und deaktivieren Safety Integrated (F-PLC) mit dem Service `SafetyModeProvider`.

Hinweis

Wenn Sie Safety Integrated (F-PLC) aktivieren oder deaktivieren, muss sich die PLC im Offline-Modus befinden.

Das folgende Beispiel zeigt, wie Sie den Service `SafetyModeProvider` aufrufen:

SafetyModeProvider aufrufen

```
...
Siemens.Engineering.HW.Device ncu = ...;
try
{
    SafetyModeProvider provider = ncu.GetService<SafetyModeProvider>();
    //Perform the safety mode change:
    provider.SetSafetyMode(SafetyMode.DbSI);
}
catch( EngineeringException ex )
{
    // Handle safety mode change failure
}
```

Das folgende Beispiel zeigt, wie Sie die aktuelle Safety Integrated-Einstellung auf dem Gerät abrufen:

Safety-Einstellung des Geräts aufrufen

```
...
Siemens.Engineering.HW.Device ncu = ...;
try
{
    SafetyModeProvider provider = ncu.GetService<SafetyModeProvider>();
    //Query the safety mode:
    SafetyMode safetyMode = provider.CurrentMode;
}
catch( (EngineeringException ex) )
{
// Handle any failure
}
```

7.27 Funktionen für SINUMERIK ONE

7.27.1 Einführung

Über TIA Portal-Openness automatisieren Sie das Engineering und steuern das TIA Portal über ein von Ihnen selbst erstelltes Programm.

Für dieses selbst erstellte Programm finden Sie in dieser Hilfe viele Informationen und Beispielcodes. Auch für die TIA Portal-Anwendung "SINUMERIK" können Sie eigene Programme zusammenstellen und anwenden.

Weitere Informationen

Bevor Sie für SINUMERIK aus den nachfolgend gelisteten Beispielcodes ein eigenes Programm zusammenstellen, beachten Sie bitte die übergreifenden Informationen zu Openness, die Sie unter folgenden Stichworten in dieser Hilfe finden:

- Voraussetzungen für TIA Portal Openness
- TIA Portal Openness installieren
- Auf das TIA Portal zugreifen
- TIA Portal Openness-Objektmodell
- Programmierschritte

7.27.2 Typkennung-Kennzeichnung der Komponenten

Jede SINUMERIK-Komponente besitzt eine eindeutige Nummer, die als Typkennung (TypeIdentifier) bezeichnet wird. Im Openness-Programmcode können Sie die Typkennung nutzen, um eine Komponente eindeutig zu identifizieren und zu benennen. Beispielsweise lautet die Typkennung für SINUMERIK ONE NCU 1750 "Artikel-Nr.: 6FC5 317-5AA00-0Axx/Vy.z".

Die Typkennung sehen Sie beim Anlegen Ihres Geräts im Dialog "Neues Gerät hinzufügen" sowie in der tabellarischen Geräteübersicht.

Sie können die Typkennung in Ihre Openness-Anwendung kopieren.

Die Typkennung wird in Openness als Aktualparameter beim Aufruf einer Methode, z. B. der CreateWithItem()-Methode, erwartet.

Anzeige des TypeIdentifiers in SINUMERIK aktivieren

1. Wählen Sie in der Projektansicht das Menü "Extras > Einstellungen".
Der Konfigurationsbereich "Einstellungen" wird geöffnet.
2. Wählen Sie in der Sekundärnavigation den Eintrag "Hardware-Konfiguration".
3. Aktivieren Sie die Option "Anzeige des Type Identifiers für Geräte und Module aktivieren".
Die Anzeige des TypeIdentifiers ist nun aktiv.

7.27.3 Grundlagen

Folgende SINUMERIK-Geräte können Sie im TIA Portal mit Openness anlegen:

- NCU
- NX-Baugruppe
- PPU

Zum Anlegen der SINUMERIK-Geräte verwenden Sie die Methode `CreateWithItem()` der `Devices` Collection.

Hinweis

Alle integrierten Subkomponenten einer SINUMERIK-NCU wie PLC, NCK, CP, HMI und SINAMICS Integrated, werden unterlagert auf der gleichen Ebene automatisch angelegt.

Besonderheiten beim Anlegen der SINUMERIK-Geräte

Die Namen der Baugruppträger entsprechen den NCU-Typen und sind in der Oberfläche schreibgeschützt. Die Bezeichnungen der Baugruppträger müssen Sie auch in TIA Portal Openness verwenden. Verwenden Sie die folgenden Standardnamen der Baugruppträger für die Erstellung eines Geräts über eine Geräteelement-Typkennung mit der `Device CreateWithItem`-Methode:

SINUMERIK-Gerät	Baugruppträger-Standardnamen
SINUMERIK ONE NCU 1750	NCU 1750
SINUMERIK ONE NCU 1760	NCU 1760
SINUMERIK ONE PPU 1740	PPU 1740

Hinweis

Alternativ können Sie die Parameternamen weglassen. Wenn der Name "null" oder "String.Empty" ist, wird der Standardname verwendet.

Folgende Parameter können mit der Methode `CreateWithItem()` verwendet werden:

- default name, z. B. `project.Devices.CreateWithItem("OrderNumber:6FC5 317-5AA00-0AA0/V6.13", "NCU 1750", "TestDevice");`
- null, z. B. `project.Devices.CreateWithItem("OrderNumber:6FC5 317-5AA00-0AA0/V6.13", null, "TestDevice");`
- `string.Empty`, z. B. `project.Devices.CreateWithItem("OrderNumber:6FC5 317-5AA00-0AA0/V6.13", string.Empty, "TestDevice");`

Weitere Informationen zu Aufrufparametern der Methode `CreateWithItem()` finden Sie im Kapitel "Erstellen eines Geräts".

Die folgende Tabelle zeigt die Zuordnung der Geräte und ihrer Typkennungen:

SINUMERIK-Gerät	Typkennung
SINUMERIK ONE NCU 1750	Artikel-Nr.: 6FC5 317-5AA00-0Axx/Vy.z
SINUMERIK ONE NCU 1760	Artikel-Nr.: 6FC5 317-6AA00-0Axx/Vy.z
SINUMERIK ONE PPU 1740	Artikel-Nr.: 6FC5 317-4AA00-1xxx/Vy.z

Beim Anlegen der SINUMERIK-Geräte sind Platzhalter in der Typkennung zulässig. Die Platzhalter tauschen Sie später gegen passende gerätespezifische Zeichen aus.

Klassifikation des Geräteelements

Jedes Gerät oder Geräteelement besitzt obligatorische Attribute, die gelesen und geschrieben werden. Weitere Informationen zu den Attributen, die in Openness unterstützt werden, finden Sie im Kapitel "Funktionen auf Geräteelementen".

Die Klassifikationsattribute des Geräteelements sind schreibgeschützt und in der Benutzeroberfläche des TIA Portals nicht sichtbar.

Die Klassifikationsattribute haben den Wert "DeviceItemClassification":

Wert des Klassifikationsattributs	Beschreibung
DeviceItemClassifications.None (0)	Keine Klassifikation.
DeviceItemClassifications.CPU (1)	Das Geräteelement ist eine CPU.
DeviceItemClassifications.HM (2)	Das Geräteelement ist ein Kopfmodul.

Wenn Sie den Wert des Geräteelements über die integrierte PLC abfragen, ist es für ein SINUMERIK-Gerät der Wert "CPU (1)".

Als Kopfmodul fungieren im Openness-Objektmodell die folgenden Komponenten:

- SINAMICS Integrated
- NX-Baugruppe

In allen anderen Fällen ist der Wert des Klassifikationsattributs "DeviceItemClassification" "None" (0).

Geräteelemente über die Eigenschaft "Kopfmodul" finden

Die folgenden Beispiele zeigen, wie Sie die Geräteelemente mit der Eigenschaft "Kopfmodul" finden, bevor Sie z. B. ein Telegramm konfigurieren.

Sie können diese Eigenschaft nur lesen, aber nicht an Geräteelemente setzen.

Sinamics Integrated / NX über die Eigenschaft "Kopfmodul" finden

```
foreach (Device device in project.Devices)
{
    foreach (DeviceItem deviceItem in device.DeviceItems)
    {
        if (deviceItem.Classification == DeviceItemClassifications.HM)
        {
            var driveObjectContainer = deviceItem.GetService<DriveObjectContainer>();
            // do something
        }
    }
}
```

Alternativ verwenden Sie das folgende Code-Beispiel, um eine PLC unabhängig von konkreter Realisierung (integrierte SINUMERIK PLC, SIMATIC PLC, Software PLC im PC) anhand der Eigenschaft "CPU" zu finden:

PLCs finden

```
Device ncuDevice = ...  
DeviceItem plc = GetPlc(ncuDevice.DeviceItems);  
  
...  
DeviceItem GetPlc(DeviceItemComposition deviceItems)  
{  
    if (deviceItems.Count == 0)  
    {  
        return null;  
    }  
    foreach (var deviceItem in deviceItems)  
    {  
        if (deviceItem.Classification == DeviceItemClassifications.CPU)  
            return deviceItem;  
  
        DeviceItem plc = GetPlc(deviceItem.DeviceItems);  
        if (plc != null)  
            return plc;  
    }  
  
    return null;  
}
```

Siehe auch

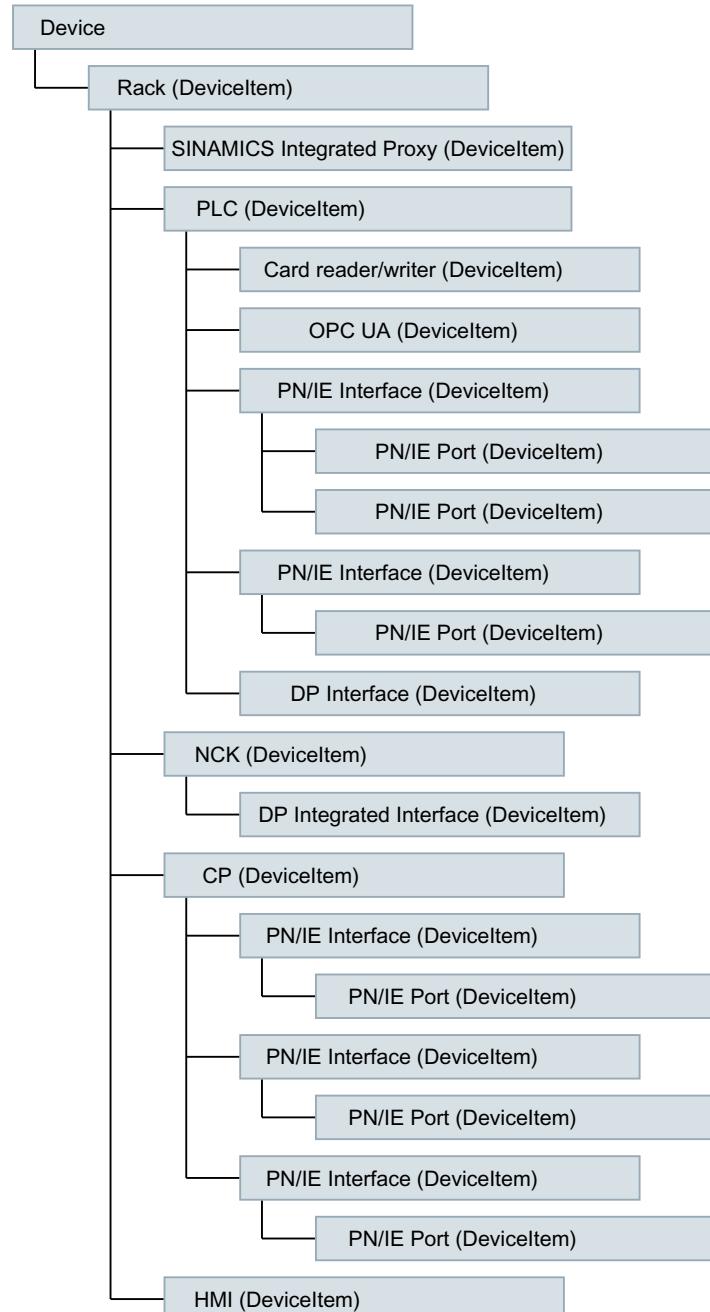
[NCU anlegen \(Seite 694\)](#)

[NX-Baugruppe anlegen \(Seite 694\)](#)

7.27.4 Objektmodell

Übersicht

Das folgende Diagramm beschreibt die Objekte, die sich bei SINUMERIK ONE unter "Device" befinden:



Weitere Informationen zu TIA Portal Openness-Objektmodell, finden Sie unter "TIA Portal Openness API".

Verfügbare Namespaces für Telegrammkonfiguration

In SINUMERIK Openess sind für die Telegrammkonfigurationsschnittstellen die folgenden Namespaces gültig:

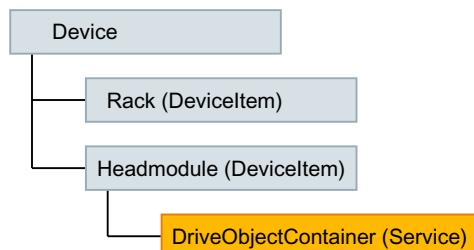
Namespace	Assembly
Siemens.Engineering.MC.DriveConfiguration (Seite 677)	Siemens.Engineering.MC.DriveConfiguration in Siemens.Engineering.dll
Siemens.Engineering.MC.Drives (Seite 683)	Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Hinweis

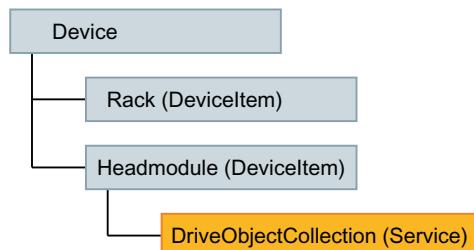
Mit dem Namespace `Siemens.Engineering.MC.DriveConfiguration` stehen Ihnen zusätzliche Funktionen zur Verfügung, z. B. können Sie Antriebsobjekte anlegen, löschen oder ihre Reihenfolge ändern.

Langfristig bleibt der Namespace `Siemens.Engineering.MC.DriveConfiguration` bestehen. Jedoch wird der Namespace `Siemens.Engineering.MC.Drives` aus Kompatibilitätsgründen in den nächsten Openess-Versionen weiterhin unterstützt.

Das folgende Diagramm beschreibt die Objekte, die sich bei SINUMERIK NX 10.3/15.3 unter "Device" im Namespace `Siemens.Engineering.MC.Drives` befinden:



Das folgende Diagramm beschreibt die Objekte, die sich bei SINUMERIK NX 10.3/15.3 unter "Device" im Namespace `Siemens.Engineering.MC.Drives` und `Siemens.Engineering.MC.DriveConfiguration` befinden:



7.27.5 Referenz

7.27.5.1 Namespace Siemens.Engineering.MC.DriveConfiguration

DriveObject

DriveObject

Die `DriveObject` Klasse ermöglicht den Zugriff auf das Antriebsobjekt. Über das Antriebsobjekt ist z. B. der Zugriff auf das Telegramm möglich.

Namespace: Siemens.Engineering.MC.DriveConfiguration
Assembly: Siemens.Engineering.MC.DriveConfiguration
 in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Zugriffsart	Zugriff	Beschreibung
Telegrams	TelegramComposition	read	-	Gibt eine Liste mit den verfügbaren Telegrammen des Antriebsobjekts zurück. Die Liste kann mit der <code>TelegramComposition</code> Klasse verändert werden.
Parent	IEngineeringObject	read	-	Gibt die Referenz auf die übergeordnete Klasse (<code>DriveObjectCollection</code> und <code>DriveObjectContainer</code>) zurück.
Category		read	-	Gibt die Referenz auf die übergeordnete Klasse (<code>DriveObjectCategory</code>) zurück
Name		read	-	Gibt den Namen des Antriebsobjekts zurück.

Die folgende Tabelle beschreibt die **Methoden** der Klasse:

Name	Beschreibung
GetAttribute	Greift lesend auf ein Attribut eines Antriebobjekts zu
GetAttributes	Greift lesend auf alle Attribute eines Antriebobjekts zu
GetAttributeInfos	Greift lesend auf Informationen zu Attributen eines Antriebsobjekts zu
SetAttribute	Greift schreibend auf ein Attribut eines Antriebobjekts zu
SetAttributes	Greift schreibend auf alle Attribute eines Antriebobjekts zu
Delete	Löscht die Antriebsobjekt-Instanz, an der "Delete" aufgerufen wird

DriveObjectCollection

DriveObjectCollection

Die `DriveObjectCollection` ist ein Service des Antriebsobjekts (`DeviceItem`) vom aktuellen Gerät (`Device`).

Namespace: Siemens.Engineering.MC.DriveConfiguration
Assembly: Siemens.Engineering.MC.DriveConfiguration
 in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Navigatoren** der `DriveObjectCollection`:

Name	Datentyp	Zugriffsart	Zugriff	Beschreibung
DriveObjects	DriveObjectComposition	read	-	Gibt eine Liste mit den verfügbaren Antriebsobjekten zurück. Die Antriebsobjekte ermöglichen den Zugriff auf Antriebsparameter und Telegramme.

Die folgende Tabelle beschreibt die **Methoden** der Klasse:

Name	Beschreibung
GetAttribute	Greift lesend auf ein Attribut von <code>DriveObjectCollection</code> zu
GetAttributes	Greift lesend auf alle Attribute von <code>DriveObjectCollection</code> zu
GetAttributeInfos	Greift lesend auf Informationen zu Attributen von <code>DriveObjectCollection</code> zu
SetAttribute	Greift schreibend auf ein Attribut von <code>DriveObjectCollection</code> zu
SetAttributes	Greift schreibend auf alle Attribute von <code>DriveObjectCollection</code> zu

DriveObjectComposition

DriveObjectComposition

Die `DriveObjectComposition` Klasse ermöglicht den Zugriff auf verfügbare Telegramme eines Antriebobjekts und enthält alle Antriebsobjekte einer NCU oder NX-Baugruppe.

Namespace: Siemens.Engineering.MC.DriveConfiguration
Assembly: Siemens.Engineering.MC.DriveConfiguration
 in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Zugriffsart	Zugriff	Beschreibung
Parent	IEngineeringObject	read	-	Gibt die Referenz auf die übergeordnete Klasse (DriveObjectContainer) zurück.
Count		read	-	
IsReadOnly		read	-	

Die folgende Tabelle beschreibt die **Methoden** der Klasse:

Name	Beschreibung
GetEnumerator	Ermöglicht die Iteration über die gegebene Menge der Elemente
Contains	Ermittelt, ob die angegebene Instanz in der Menge der Elemente enthalten ist. TRUE: Der Container enthält die Instanz FALSE: Der Container enthält die Instanz nicht
IndexOf	Gibt den Index in der Menge der Elemente für die angefragte Instanz zurück.
Create	Erstellt eine DriveObjectComposition-Klasse
Find	Sucht eine DriveObjectComposition-Klasse

DriveObjectCategory

DriveObjectCategory

Das Enum `DriveObjectCategory` enthält vordefinierte Kategorien der Antriebsobjekte. Auf `DriveObjectCategory` ist ein lesender Zugriff möglich.

Namespace: Siemens.Engineering.MC.DriveConfiguration

Assembly: Siemens.Engineering.MC.DriveConfiguration
in Siemens.Engineering.dll

Die folgende Tabelle enthält die vordefinierten Kategorien der Antriebsobjekte:

Unterstützte Antriebsobjekte in SINUMERIK	DriveObjectCategory
SinumerikDriveAxis	SERVO, ENC, HLA, TM41
SinumerikControlUnit	CU_I, CU_NX_CX
SinumerikInfeed	A_INF, B_INF, S_INF

TelegramComposition

TelegramComposition

Die `TelegramComposition` Klasse ermöglicht den Zugriff auf die Telegramme eines Antriebsobjekts (`DriveObject` (Seite 677)). Der Aufbau eines Telegramms kann über die `Telegram` (Seite 681) Klasse ausgelesen werden.

Änderungen an Telegrammobjekten (z. B. Wechsel des Safety-Telegrams), können dazu führen, dass in der `TelegramComposition` das entsprechende Telegrammobjekt gelöscht und ein neues angelegt wird. In diesem Fall müssen Sie die `TelegramComposition` erneut durchsuchen, um das neue Telegrammobjekt (Seite 681) nach der Änderung wieder zu finden.

Wenn der jeweilige Antriebsobjekttyp keine Telegramme unterstützt, wird der Wert für `TelegramComposition` leer zurückgegeben.

Namespace: Siemens.Engineering.MC.DriveConfiguration

Assembly: Siemens.Engineering.MC.DriveConfiguration
in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Zugriffsart	Zugriff	Beschreibung
Parent	IEngineeringObject	read	-	Gibt die Referenz auf die übergeordnete Klasse (<code>DriveObject</code>) zurück.
Count		read	-	
IsReadOnly		read	-	

Die folgende Tabelle beschreibt die **Methoden** der Klasse:

Name	Beschreibung
Create(enum TelegramId Id)	Erstellt ein Telegramm, der Telegrammtyp entspricht der ID.
int IndexOf(TelegramType)	Gibt den Index in der Menge der Elemente für die angefragte Instanz zurück.
bool Contains	Ermittelt, ob die angegebene Instanz in der Menge der Elemente enthalten ist. TRUE: Der Container enthält die Instanz FALSE: Der Container enthält die Instanz nicht
Contains	Ermittelt, ob die angegebene Instanz in der Menge der Elemente enthalten ist.TRUE: Der Container enthält die Instanz FALSE: Der Container enthält die Instanz nicht
IEnumerator GetEnumerator	Ermöglicht <code>IEnumerator<DriveObject></code> die Iteration über die gegebene Menge der Elemente

Telegram

Telegram

Die `Telegram` Klasse ermöglicht den Zugriff auf den Aufbau eines Telegramms von einem Antriebsobjekt.

Namespace: Siemens.Engineering.MC.DriveConfiguration
Assembly: Siemens.Engineering.MC.DriveConfiguration
 in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Zugriffsart	Zugriff	Beschreibung
<code>Id</code>	<code>Int32</code>	<code>read</code>	-	Gibt die ID des Telegramms zurück.
<code>Type</code>	<code>enum: TelegramType (Seite 682)</code>	<code>read</code>	-	Gibt den Typ des Telegramms als <code>Enum TelegramType</code> zurück.
<code>Addresses</code>	<code>AddressComposition (Seite 690)</code>	<code>read</code>	-	Gibt eine <code>AddressComposition</code> mit Informationen zur Adresse zurück.

Die folgende Tabelle beschreibt die **Methoden** der Klasse:

Name	Beschreibung
<code>GetSize (AddressIoType (Seite 692))</code>	Gibt die Größe der Eingänge oder Ausgänge des Telegramms zurück.
<code>CanChangeSize (AddressIoType (Seite 692), Int32, bool)</code>	Gibt <code>true</code> zurück, wenn die Größe des Telegramms wie parametriert geändert werden kann. Standardtelegramme können nur vergrößert werden. Das Beibehalten der bisherigen Telegrammadresse wird berücksichtigt, wenn die Option mit <code>true</code> parametriert ist.
<code>CanSetSize (AddressIoType (Seite 692) Int32, bool)</code>	Gibt <code>true</code> zurück, wenn die Größe des Telegramms wie parametriert geändert werden kann.
<code>SetSize</code>	Greift schreibend auf die Telegrammgröße zu
<code>Delete</code>	Löscht die Telegramm-Instanz, an der "Delete" aufgerufen wird
<code>GetAttribute</code>	Greift lesend auf ein Telegramm-Attribut zu
<code>GetAttributes</code>	Greift lesend auf alle Telegramm-Attribute zu
<code>GetAttributeInfos</code>	Greift lesend auf Informationen zu Telegramm-Attributen zu
<code>SetAttribute</code>	Greift schreibend auf ein Telegramm-Attribut zu
<code>SetAttributes</code>	Greift schreibend auf alle Telegramm-Attribute zu

Eigenschaften von Safety-Telegrammen

Die folgende Tabelle beschreibt die zusätzlichen **Eigenschaften** der Klasse "Telegram" vom Typ "SafetyTelegram".

Name	Datentyp	Zugriffsart	Zugriff	Beschreibung
Failsafe_FSourceAddress	UInt32	read	dynamisch	PROFIsafe-Quell-Adresse
Failsafe_FDestinationAddress	UInt32	read/write	dynamisch	PROFIsafe-Zieladresse
Failsafe_FIODBNumber	UInt32	read/write	dynamisch	Safety DB-Nummer, schreibender Zugriff nur möglich, wenn der Eigenschaft "Failsafe_ManualAssignmentFIODBNumber" der Wert "1" zugewiesen ist
Failsafe_FIODBName	string	read	dynamisch	Name Safety-DB
Failsafe_ManualAssignmentFIODBNu mber	bool	read/write	dynamisch	Einstellung für manuelle Zuweisung der Eigenschaft "Failsafe_FIODBNumber"
Failsafe_FMonitoringtime	UInt32	read/write	dynamisch	Safety-Überwachungszeit, schreibender Zugriff nur möglich, wenn die Eigenschaft "Failsafe_ManualAssignmentFMoni toringtime" auf "true" gesetzt ist
Failsafe_ManualAssignmentFMonito ringtime	bool	read/write	dynamisch	Einstellung für manuelle Zuweisung der Eigenschaft "Failsafe_FMonitoringtime"

Der Zugriff auf die zusätzliche Eigenschaften erfolgt über GetAttribute und SetAttribute z. B. GetAttribute("Failsafe_FSourceAddress"). Als Rückgabewert wird ein UInt32 erwartet.

TelegramType

TelegramType

Das Enum TelegramType enthält vordefinierte Telegrammtypen.

Namespace: Siemens.Engineering.MC.DriveConfiguration

Assembly: Siemens.Engineering.MC.DriveConfiguration
in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Enum-Einträge**:

Name	Beschreibung
MainTelegram	Haupttelegramm: z. B. Telegramm 136
SupplementaryTelegram	Zusatztrogramm: z. B. Telegramm 701
AdditionalTelegram	Erweiterung: freies Telegramm
SafetyTelegram	Safety-Telegramm: z. B. Telegramm 903

Hinweis

Das Drehmoment-Telegramm (`TorqueTelegram`) wird nicht unterstützt, obwohl es in der TIA Portal Openness-API im Kontext SINUMERIK verfügbar ist.

TelegramId

TelegramId

Das Enum `TelegramId` enthält die Telegrammnummern, die für die Kommunikation zwischen der PLC und dem Antrieb relevant sind. Die IDs sind über den PROFIdrive-Standard festgelegt.

Namespace: Siemens.Engineering.MC.DriveConfiguration

Assembly: Siemens.Engineering.MC.DriveConfiguration
in Siemens.Engineering.dll

7.27.5.2 Namespace Siemens.Engineering.MC.Drives

DriveObject

DriveObject

Die `DriveObject` Klasse ermöglicht den Zugriff auf das Antriebsobjekt. Über das Antriebsobjekt ist z. B. der Zugriff auf das Telegramm möglich.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Zugriffsart	Zugriff	Beschreibung
Telegrams	TelegramComposition	read	-	Gibt eine Liste mit den verfügbaren Telegrammen des Antriebsobjekts zurück. Die Liste kann mit der <code>TelegramComposition</code> Klasse verändert werden.
Parent	IEngineeringObject	read	-	Gibt die Referenz auf die übergeordnete Klasse (<code>DriveObjectContainer</code>) zurück.

Die folgende Tabelle beschreibt die **Methoden** der Klasse:

Name	Beschreibung
GetAttribute	Greift lesend auf ein Attribut eines Antriebobjekts zu
SetAttribute	Greift schreibend auf ein Attribut eines Antriebobjekts zu
GetEnumerator	Ermöglicht die Iteration über die gegebene Menge der Elemente

Siehe auch

[TelegramComposition \(Seite 685\)](#)

DriveObjectContainer

DriveObjectContainer

Der `DriveObjectContainer` ist ein Service des Antriebsobjekts (`DeviceItem`) vom aktuellen Gerät (`Device`).

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Navigatoren** des `DriveObjectContainer`:

Name	Datentyp	Zugriffsart	Zugriff	Beschreibung
DriveObjects	DriveObjectComposition	read	-	Gibt eine Liste mit den verfügbaren Antriebsobjekten zurück. Die Antriebsobjekte ermöglichen den Zugriff auf Antriebsparameter und Telegramme.
Parent	IEngineeringObject	read	-	Gibt die Referenz auf die übergeordnete Klasse (<code>DeviceItem</code>) zurück.

DriveObjectCollection

DriveObjectCollection

Die `DriveObjectCollection` ist ein Service des Antriebsobjekts (`DeviceItem`) vom aktuellen Gerät (`Device`).

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Navigatoren** der DriveObjectCollection:

Name	Datentyp	Zugriffsart	Zugriff	Beschreibung
DriveObjects	DriveObjectComposition	read	-	Gibt eine Liste mit den verfügbaren Antriebsobjekten zurück. Die Antriebsobjekte ermöglichen den Zugriff auf Antriebsparameter und Telegramme.
Parent	IEngineeringObject	read	-	Gibt die Referenz auf die übergeordnete Klasse (DeviceItem) zurück.

DriveObjectComposition

DriveObjectComposition

Die **DriveObjectComposition** Klasse ermöglicht den Zugriff auf verfügbare Telegramme eines Antriebobjekts.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Zugriffsart	Zugriff	Beschreibung
IndexOf	int32	read	-	Gibt den Index in der Menge der Elemente für die angefragte Instanz zurück.
Contains	bool	read	-	Ermittelt, ob die angegebene Instanz in der Menge der Elemente enthalten ist. TRUE: Der Container enthält die Instanz FALSE: Der Container enthält die Instanz nicht
GetEnumerator	IEnumerator<DriveObject>	read	-	Ermöglicht die Iteration über die gegebene Menge der Elemente

TelegramComposition

TelegramComposition

Die **TelegramComposition** Klasse ermöglicht den Zugriff auf die Telegramme eines Antriebsobjekts (DriveObject (Seite 683)). Der Aufbau eines Telegramms kann über die **Telegram** (Seite 687) Klasse ausgelesen werden.

Änderungen an Telegrammobjekten (z. B. Wechsel des Safety-Telegrams), können dazu führen, dass in der `TelegramComposition` das entsprechende Telegrammobjekt gelöscht und ein neues angelegt wird. In diesem Fall müssen Sie die `TelegramComposition` erneut durchsuchen, um das neue Telegrammobjekt (Seite 687) nach der Änderung wieder zu finden.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Methoden** der Klasse:

Name	Beschreibung
<code>CanInsertAdditionalTelegram(Int32, Int32)</code>	Gibt <code>true</code> zurück, wenn eine Erweiterung entsprechend den parametrierten Größen (Eingangs- und Ausgangsgröße) erstellt werden kann.
<code>InsertAdditionalTelegram(Int32, Int32)</code>	Erstellt für das Antriebsobjekt eine Erweiterung entsprechend den parametrierten Größen und gibt <code>true</code> zurück, wenn die Erweiterung eingefügt werden konnte. Im Fehlerfall wird eine <code>EngineeringTargetInvocationException</code> ausgelöst.
<code>CanInsertSupplementaryTelegram(Int32)</code>	Gibt <code>true</code> zurück, wenn ein Zusatztelegramm entsprechend der parametrierten Telegrammnummer erstellt werden kann.
<code>InsertSupplementaryTelegram(Int32)</code>	Erstellt das Zusatztelegramm mit der parametrierten Telegrammnummer und gibt <code>true</code> zurück, wenn das Telegramm eingefügt werden konnte. Im Fehlerfall wird eine <code>EngineeringTargetInvocationException</code> ausgelöst.
<code>CanInsertSafetyTelegram(Int32)</code>	Gibt <code>true</code> zurück, wenn ein Safety-Telegramm entsprechend der parametrierten Telegrammnummer erstellt werden kann.
<code>InsertSafetyTelegram(Int32)</code>	Erstellt das Zusatztelegramm mit der parametrierten Telegrammnummer und gibt <code>true</code> zurück, wenn das Telegramm eingefügt werden konnte. Im Fehlerfall wird eine <code>EngineeringTargetInvocationException</code> ausgelöst.

Name	Beschreibung
EraseTelegram(TelegramType)	Gibt true zurück, wenn das parametrierte Telegramm gelöscht werden konnte. Standardtelegramme können nicht gelöscht werden. Im Fehlerfall wird eine EngineeringTargetInvocationException ausgelöst.
Find(TelegramType)	Gibt das Objekt Telegram (Seite 687) zurück, wenn es über den parametrisierten Telegramp-typ gefunden werden konnte. null, wenn das Telegramm nicht gefunden wird. Beispiel Telegram telegram = telegrams.Find(TelegramType.MainTelegam);
int IndexOf(TelegramType)	Gibt den Index in der Menge der Elemente für die angefragte Instanz zurück.
Parent	Gibt die Referenz auf die übergeordnete Klasse (DriveObject) zurück.
bool Contains	Ermittelt, ob die angegebene Instanz in der Menge der Elemente enthalten ist. TRUE: Der Container enthält die Instanz FALSE: Der Container enthält die Instanz nicht
IEnumerator GetEnumerator	Ermöglicht Ienumerator<DriveObject> die Iteration über die gegebene Menge der Elemente

Telegram

Telegram

Die `Telegram` Klasse ermöglicht den Zugriff auf den Aufbau eines Telegramms von einem Antriebsobjekt.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Zugriffsart	Zugriff	Beschreibung
TelegramNumber	Int32	read	-	Gibt die Nummer des Haupttelegramms zurück oder legt die Nummer fest. Ein freies Telegramm hat die Nummer 999.
Type	enum: TelegramType (Seite 690)	read	-	Gibt den Typ des Telegramms als Enum TelegramType zurück.
Addresses	AddressComposition (Seite 690)	read	-	Gibt eine AddressComposition mit Informationen zur Adresse zurück.

Die folgende Tabelle beschreibt die **Methoden** der Klasse:

Name	Beschreibung
CanChangeTelegram (Int32)	Gibt true zurück, wenn das Telegramm in den parametrierten Standardtyp geändert werden kann.
GetSize (AddressIoType (Seite 692))	Gibt die Größe der Eingänge oder Ausgänge des Telegramms zurück.
CanChangeSize (AddressIoType (Seite 692), Int32, bool)	Gibt true zurück, wenn die Größe des Telegramms wie parametriert geändert werden kann. Standardtelegramme können nur vergrößert werden. Das Beibehalten der bisherigen Telegrammadresse wird berücksichtigt, wenn die Option mit true parametriert ist.
ChangeSize (AddressIoType (Seite 692), Int32, bool)	Gibt true zurück, wenn die Größe des Telegramms wie parametriert geändert werden konnte. Das Beibehalten der bisherigen Telegrammadresse wird berücksichtigt, wenn die Option mit true parametriert ist.
GetEnumerator	Ermöglicht die Iteration über die gegebene Menge der Elemente

Eigenschaften von Safety-Telegrammen

Die folgende Tabelle beschreibt die zusätzlichen **Eigenschaften** der Klasse "Telegram" vom Typ "SafetyTelegram".

Name	Daten-typ	Zu-griffsart	Zugriff	Beschreibung
Failsafe_FSourceAddress	UInt32	read	Zugriff nur über GetAttribute oder SetAttribute möglich	PROFIsafe-Quell-Adresse
Failsafe_FDestinationAddress	UInt32	read/write	Zugriff nur über GetAttribute oder SetAttribute möglich	PROFIsafe-Zieladresse
Failsafe_FIODBNumber	UInt32	read/write	Zugriff nur über GetAttribute oder SetAttribute möglich	Safety DB-Nummer, schreibender Zugriff nur möglich, wenn der Eigenschaft "Failsafe_ManualAssignmentFIODBNumber" der Wert "1" zugewiesen ist
Failsafe_FIODBName	string	read	Zugriff nur über GetAttribute oder SetAttribute möglich	Name Safety-DB
Failsafe_ManualAssignmentFIODBNumber	bool	read/write	Zugriff nur über GetAttribute oder SetAttribute möglich	Einstellung für manuelle Zuweisung der Eigenschaft "Failsafe_FIODBNumber"
Failsafe_FMonitoringtime	UInt32	read/write	Zugriff nur über GetAttribute oder SetAttribute möglich	Safety-Überwachungszeit, schreibender Zugriff nur möglich, wenn die Eigenschaft "Failsafe_ManualAssignmentFMonitoringtime auf "true" gesetzt ist
Failsafe_ManualAssignmentFMonitoringtime	bool	read/write	Zugriff nur über GetAttribute oder SetAttribute möglich	Einstellung für manuelle Zuweisung der Eigenschaft "Failsafe_FMonitoringtime"

Der Zugriff auf die zusätzlichen Eigenschaften erfolgt über GetAttribute und SetAttribute z. B. `GetAttribute("Failsafe_FSourceAddress")`. Als Rückgabewert wird ein UInt32 erwartet.

TelegramType

TelegramType

Das Enum TelegramType enthält vordefinierte Telegrammtypen.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Enum-Einträge**:

Name	Beschreibung
MainTelegram	Haupttelegramm: z. B. Telegramm 136
SupplementaryTelegram	Zusatztrogramm: z. B. Telegramm 701
AdditionalTelegram	Erweiterung: freies Telegramm
SafetyTelegram	Safety-Telegramm: z. B. Telegramm 903

Hinweis

Das Drehmoment-Telegramm (`TorqueTelegram`) wird nicht unterstützt, obwohl es in der TIA Portal Openness-API im Kontext SINUMERIK verfügbar ist.

AddressComposition

AddressComposition

Die AddressComposition Klasse repräsentiert die Adresse eines Telegramms.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Zugriffsart	Zugriff	Beschreibung
IoType	enum: AddressIoType	read	-	Gibt Informationen zum Typ der Adresse zurück.
Context	enum: AddressContext	read	Zugriff nur über GetAttribute oder SetAttribute möglich	Gibt Informationen zum Kontext der Adresse zurück.
StartAddress	Int32	read/write	-	Gibt die Startadresse des Telegramms zurück oder legt die Startadresse fest.
Length	Int32	read	-	Gibt die Länge des Telegramms zurück.
IndexOf	int32	read	-	Gibt den Index in der Menge der Elemente für die angefragte Instanz zurück.
Contains	bool	read	-	Ermittelt, ob die angegebene Instanz in der Menge der Elemente enthalten ist. TRUE: Der Container enthält die Instanz FALSE: Der Container enthält die Instanz nicht
GetEnumerator	IEnumerator<DriveObject>	read	-	Ermöglicht die Iteration über die gegebene Menge der Elemente

Hinweis

Wenn Sie über Address.Parent zurück zum Telegramm navigieren möchten, wird statt des Namespace `Siemens.Engineering.MC.Drives.Telegram` der Namespace `Siemens.Engineering.MC.DriveConfiguration.Telegram` aufgerufen.

Weitere Informationen zu TIA Portal Openness-Bibliotheken, finden Sie unter "Standard-Bibliotheken".

Siehe auch

[AddressIoType \(Seite 692\)](#)

[AddressContext \(Seite 692\)](#)

AddressContext

AddressContext

Das Enum AddressContext enthält Informationen zum Kontext der Adresse.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Enum-Einträge**:

Name	Beschreibung
AddressContext.None	Für die Adresse wurde kein Kontext gefunden
AddressContext.Device	Kontext ist eine Geräteadresse
AddressContext.Head	Kontext ist eine Kopfadresse

Weitere Informationen zu TIA Portal Openness-Bibliotheken, finden Sie unter "Standard-Bibliotheken".

AddressIoType

AddressIoType

Das Enum AddressIoType enthält Informationen zum Typ der Adresse.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Enum-Einträge**:

Name	Beschreibung
AddressIoType.None	Der IO-Typ ist nicht anwendbar
AddressIoType.Input	Typ ist eine Eingangsadresse
AddressIoType.Output	Typ ist eine Ausgangsadresse
AddressIoType.Diagnosis	Typ ist eine Diagnoseadresse
AddressIoType.Substitute	Typ ist eine Ersatzadresse

Weitere Informationen zu TIA Portal Openness-Bibliotheken, finden Sie unter "Standard-Bibliotheken".

7.27.5.3 ArchiveProvider

ArchiveProvider

Die Klasse ArchiveProvider dient zum Generieren der PLC-Archive.

Namespace: Siemens.Engineering.HW.Utilities
Assembly: Siemens.Engineering.HW.Utilities
 in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die Methoden der Klasse ArchiveProvider:

Name	Parameter	Beschreibung
Archive	DeviceItem plc FileInfo path Siemens.Engineering.MC.Sinumerik.SinumerikA rchivationMode String comment (optional) String author (optional)	Erstellt ein PLC-Archiv

Hinweis

Wenn Safety aktiviert ist, können Sie kein Archiv über Openness erzeugen.

7.27.6 Codebeispiel

7.27.6.1 Allgemein

Die folgenden Codebeispiele beschreiben die grundsätzliche Vorgehensweise für verschiedene Anwendungsfälle. Der Code ist nicht zwingend vollständig und kompilierbar.

7.27.6.2 Erste Schritte in SINUMERIK durchführen

- Verbinden Sie die TIA Portal Openness-Anwendung mit dem TIA Portal.
- Öffnen Sie Ihr Projekt.

Das folgende Beispiel zeigt, wie Sie ermitteln können, welche SINUMERIK Toolbox-Version installiert ist.

SINUMERIK Toolbox-Version ermitteln

```
using Siemens.Engineering;
```

SINUMERIK Toolbox-Version ermitteln

```
if (tiaProcess.InstalledSoftware.Any(sw => sw.Name.Equals("SINUMERIK Toolbox") &&
sw.Version.Equals("V16")))
{
    Console.WriteLine("SINUMERIK Toolbox is available");
}
// "V16" is the current SINUMERIK version started at December 2019.
```

7.27.6.3 NCU anlegen

Mit der Methode `CreateWithItem()` der `Devices Collection` legen Sie eine SINUMERIK-NCU an. Die SINUMERIK-NCUs werden über Parameter der Methode spezifiziert. Das Format der Parameter ist nachfolgend beschrieben.

NCU anlegen

Format der Parameter für SINUMERIK-NCU:

```
CreateWithItem(@"OrderNumber:mlfb/FirmwareVersion/",
"StandardSubrackName", "NameOfTheDevice")
```

Das folgende Beispiel zeigt, wie Sie eine Steuerung SINUMERIK ONE NCU 1750 anlegen.

SINUMERIK ONE NCU 1750 anlegen

```
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
Project tiaproject= portal.Projects.Open(...); //The path of the project

Device NCUDevice = tiaproject.Devices.CreateWithItem(@"OrderNumber:6FC5
317-5AA00-0AA0/6.13/", "NCU 1750", "TestDevice");
```

7.27.6.4 NX-Baugruppe anlegen

Mit der `Device CreateWithItem`-Methode legen Sie eine NX-Baugruppe an. Anschließend verbinden Sie die NX-Baugruppe mit einer NCU.

Folgende Typkennungen werden für SINUMERIK NX-Baugruppen verwendet:

SINUMERIK NX-Baugruppe	Typkennung
SINUMERIK NX10.3	OrderNumber:6SL3 040-1NC00-0Axx/Vy.z
SINUMERIK NX15.3	OrderNumber:6SL3 040-1NB00-0Axx/Vy.z

Das folgende Beispiel zeigt, wie Sie eine SINUMERIK NX-Baugruppe anlegen.

NX-Baugruppe anlegen

```
project.Devices.CreateWithItem("OrderNumber:6SL3040-1NC00-0AA0/
V5.2", "MyNXDevice", "TestDevice");
```

Versionskompatibilität

Die Firmware-Version der NX-Baugruppe muss mit der Firmware-Version von SINAMICS Integrated übereinstimmen und mit der Firmware-Version der NCU kompatibel sein.

Die von SINAMICS Integrated abweichende NX Firmware-Version kann über Openness nicht zugeordnet werden.

Die folgende Tabelle zeigt die Versionskompatibilität für SINUMERIK ONE:

NCU-Firmware (SINUMERIK ONE)	SINAMICS Integrated/NX-Firmware
V6.13	V5.2

7.27.6.5 NX-Baugruppe mit NCU verbinden

NX-Baugruppe mit einer NCU verbinden

Um eine NX-Baugruppe mit einer NCU über den Subnetztyp "ProfibusIntegrated" zu verbinden, muss der Dienst "NetworkInterface" über das Kopfmodul von NX geladen werden.

Das folgende Beispiel zeigt, wie Sie den Dienst "NetworkInterface" laden.

NetworkInterface-Dienst laden

```
foreach (Device device in project.Devices)
{
    foreach (DeviceItem deviceItem in device.DeviceItems)
    {
        if (deviceItem.Classification == DeviceItemClassifications.HM)
        {
            var networkInterface = deviceItem.GetService<NetworkInterface>();
            // do something
        }
    }
}
```

Das folgende Beispiel zeigt, wie Sie eine NX-Baugruppe über "ProfibusIntegrated" mit einer NCU verbinden:

NX-Baugruppe über ProfibusIntegrated verbinden

```
Subnet pbiSubnet = ...;
Node node = networkInterface.Nodes.FirstOrDefault();
node.ConnectToSubnet(pbiSubnet);
```

Die DP-Adresse wird in Openness über DRIVE-CLiQ Port der fest NX zugeordnet. Jedes Port-Label verfügt über eine feste DP-Integrated-Adresse.

Die DP-Adresse muss vor dem Verbinden mit der NCU der NX zugeordnet werden.

7.27.6.6 Auf NCK-Ereignisse zugreifen

NCK-Ereignisse

Das NCK-Modul ist laut dem SINUMERIK-Objektmodell (Seite 675) ein DeviceItem.

Mit den folgenden Attributen des NCK-Moduls greifen Sie auf die NCK-Ereignisse zu und können die NCK-Ereignisse in TIA Portal-Openness konfigurieren:

Name	Datentyp	Zugriffsart	Beschreibung
HardwareInterruptNckToPlcSignalExchangeActive	Boolean	r/w	Aktivieren/deaktivieren des Ereignisses
HardwareInterruptNckToPlcSignalExchangeEventName	String	r/w	Ereignisname
HardwareInterruptNckToPlcSignalExchangeInterrupt	Siemens.Engineering.SW.Blocks.OB	r/w	Der dem Ereignis zugewiesene OB
HardwareInterruptNckToPlcSignalExchangePriority	Int32	r/w	Ereignispriorität

Alle NCK-Ereignisse werden über `HardwareInterrupt` (Prozessalarm-OB) getriggert. Die Prozessalarm-OBs unterbrechen die zyklische Programmbearbeitung aufgrund eines Hardware-Ereignisses.

Das folgende Beispiel zeigt, wie Sie den Ereignisnamen für NCK-Modul ermitteln:

Ereignisnamen für NCK-Modul ermitteln

```
DeviceItem nck = ...;
string eventName =
(string)nck.GetAttribute("HardwareInterruptNckToPlcSignalExchangeEventName");
```

Das folgende Beispiel zeigt, wie Sie den `HardwareInterrupt` setzen:

HardwareInterrupt setzen

```
... DeviceItem nck = ...;
OB ob40 = ... try
{
  nck.SetAttribute("HardwareInterruptNckToPlcSignalExchangeInterrupt", ob40);
}
catch( (EngineeringException ex) )
{
// Handle setting failure
}
```

7.27.6.7 Archive erstellen

Archive für Serieninbetriebnahme erstellen

Mit TIA Portal-Openness erstellen und exportieren Sie SINUMERIK-Archive, um z. B. die Serieninbetriebnahme zu vereinfachen.

Hinweis

Während der Erstellung der Archive muss sich die PLC im Offline-Modus befinden. Der Safety-Modus darf nicht aktiv sein.

Sie erstellen SINUMERIK-Archive über die TIA Portal-Projekteigenschaft `HwUtilities` mit dem Service `SinumerikArchiveProvider`.

Das folgende Beispiel zeigt, wie Sie den Service `SinumerikArchiveProvider` aufrufen:

SinumerikArchiveProvider aufrufen

```
Project project = ...;
SinumerikArchiveProvider archiveProvider =
project.HwUtilities.Find("SinumerikArchiveProvider") as SinumerikArchiveProvider;

if (archiveProvider != null)
{
// Work with the provider
}
```

Das folgende Beispiel zeigt, wie Sie ein PLC-Archiv erstellen, das die Hardware-Informationen und alle Datenbausteine enthält:

PLC-Archiv erstellen

```
...
Siemens.Engineering.HW.DeviceItem plc = ...;

try {
    // The file extension is required
    string archivePath = string.Format(@"D:\some_path\{0}.dsf", plc.Name);

    // Comment and author arguments are optional
    archiveProvider.Archive(plc, new FileInfo(archivePath),
    SinumerikArchivationMode.HardwareAndAllProgramBlocks[, "Comment", "Author name"]);
}
catch (EngineeringTargetException ex)
{
// Handle archive failure
}
```

Das folgende Beispiel zeigt, wie Sie die Software-Anteile eines bereits erstellten PLC-Archivs aktualisieren:

Archivanteile aktualisieren

```
...
Siemens.Engineering.HW.DeviceItem plc_1 = ...;
Siemens.Engineering.HW.DeviceItem plc_1_copy = ...;
try {
    // The file extension is required
    string archivePath = @"D:\some_path\SinumerikArchive.dsf";

    // Create a Sinumerik archive with HardwareAndAllProgramBlocks
    archiveProvider.Export(plc_1, new FileInfo(archivePath),
    SinumerikArchivationMode.HardwareAndAllProgramBlocks);

    // Update the software part in the previously created archive using
    UpdateProgramBlocksOfArchive method
    archiveProvider.UpdateProgramBlocksOfArchive(plc_1_copy, new FileInfo(archivePath));
}
catch (EngineeringException ex)
{
    // Handle export failure
}
```

7.27.6.8 Safety Integrated aktivieren

Safety Integrated aktivieren

Mit TIA Portal Openness können Sie Safety Integrated (F-PLC) in den Eigenschaften der NCU aktivieren.

Hinweis

Auswirkungen auf Telegrammkonfiguration

Der verwendete Safety Integrated-Modus hat Auswirkungen auf die Telegrammkonfiguration, da im Modus Safety Integrated plus (F-PLC) andere Telegramme verwendet werden, als bei inaktivem Safety Integrated-Modus.

Hinzugefügte oder geänderte Telegramme bleiben jedoch erhalten, sofern diese mit dem neu gewählten Safety Integrated-Modus kompatibel sind.

Stellen Sie ggf. nach der Modusänderung in der Telegrammkonfiguration sicher, dass etwaige Anpassungen noch vorhanden sind.

Sie aktivieren und deaktivieren Safety Integrated (F-PLC) mit dem Service SafetyModeProvider.

Hinweis

Wenn Sie Safety Integrated (F-PLC) aktivieren oder deaktivieren, muss sich die PLC im Offline-Modus befinden.

Das folgende Beispiel zeigt, wie Sie den Service SafetyModeProvider aufrufen:

SafetyModeProvider aufrufen

```
...
Siemens.Engineering.HW.Device ncu = ...;
try
{
    SafetyModeProvider provider = ncu.GetService<SafetyModeProvider>();
    //Perform the safety mode change:
    provider.SetSafetyMode(SafetyMode.DbSI);
}
catch( (EngineeringException ex) )
{
    // Handle safety mode change failure
}
```

Das folgende Beispiel zeigt, wie Sie die aktuelle Safety Integrated-Einstellung auf dem Gerät abrufen:

Safety-Einstellung des Geräts aufrufen

```
...
Siemens.Engineering.HW.Device ncu = ...;
try
{
    SafetyModeProvider provider = ncu.GetService<SafetyModeProvider>();
    //Query the safety mode:
    SafetyMode safetyMode = provider.CurrentMode;
}
catch( (EngineeringException ex) )
{
    // Handle any failure
}
```

7.27.6.9 Beispiele für Namespace Siemens.Engineering.MC.DriveConfiguration

Telegramme vorbereiten

Die Antriebskommunikation einer SINUMERIK-NCU erfolgt mithilfe von Telegrammen über die Subkomponenten SINAMICS Integrated und ggf. über zusätzlich angeschlossene NX-Baugruppen.

Hinweis

Die SINUMERIK-NCU und ein SINAMICS Integrated befinden sich auf derselben Ebene im TIA Portal Openness-Objektmodell und erscheinen unter "DeviceComposition" als zwei unterschiedliche Geräte.

Um ein Telegramm zu konfigurieren, verwenden Sie "DriveObjectCollection". "DriveObjectCollection" ist ein Service des Antriebsobjekts (Geräteelement) vom aktuellen Kopfmodul (Geräteelement).

Um den Service "DriveObjectCollection" zu starten, navigieren Sie zu SINAMICS Integrated oder zum Kopfmodul der NX-Baugruppe. Die Hierarchie zwischen Geräten und Geräteelementen ist für SINAMICS Integrated und NX-Baugruppen identisch.

Das folgende Beispiel zeigt, wie Sie "DriveObjectCollection" über die Eigenschaft "Kopfmodul" finden:

DriveObjectCollection über Kopfmodul finden

```
foreach (Device device in project.Devices)
{
    foreach (DeviceItem deviceItem in device.DeviceItems)
    {
        if (deviceItem.Classification == DeviceItemClassifications.HM)
        {
            var driveObjectCollection = deviceItem.GetService<DriveObjectCollection>();
            // do something
        }
    }
}
```

Die SINUMERIK-NCU enthält ein SINAMICS Integrated Proxy-Objekt mit Referenzen auf SINAMICS Integrated.

Um auf ein SINAMICS Integrated-Gerät oder eine NX-Baugruppe zuzugreifen, navigieren Sie von der SINUMERIK-NCU über NCK zur DP Integrated-Schnittstelle. Anschließend ermitteln Sie das PROFIBUS-Mastersystem und navigieren zum verbundenen Slave.

Telegramme einfügen und löschen

Das folgende Beispiel zeigt, wie Sie ein Telegramm einfügen. Für den Zugriff benötigen Sie ein Antriebsobjekt.

Sie unterscheiden zwischen den Telegrammtypen mit Hilfe von ihren IDs.

Telegramm einfügen und auf Telegramm-Attribute zugreifen

```
using Siemens.Engineering.MC.DriveConfiguration;
```

Telegramm einfügen und auf Telegramm-Attribute zugreifen

```
TelegramComposition telegrams = drvObj.Telegrams;

//Create telegram
const int tgrmId = 136;
drvObj.Telegrams.CreateTelegram(tgrmId);

//Create safety telegram
const int tgrmId = 30;
drvObj.Telegrams.CreateTelegram(tgrmId);

// Get and set safety telegram attributes
uint watchDogTime =
(uint)safetyTgrm.GetAttribute("Failsafe_FMonitoringtime");

safetyTgrm.SetAttribute("Failsafe_FMonitoringtime", 300);

const int newSafetyTelegramNumber= 902;
if (safetyTgrm.CanChangeTelegram(newSafetyTelegramId)) {
safetyTgrm.TelegramId = newSafetyTelegramId; }
```

Das folgende Beispiel zeigt, wie Sie ein Telegramm löschen.

Telegramm löschen

```
using Siemens.Engineering.MC.DriveConfiguration;
//Delete telegram
const int tgrmId = 136;
drvObj.Telegrams.DeleteTelegram(tgrmId);
```

Safety-Telegramme einfügen und löschen

Das folgende Beispiel zeigt, wie Sie ein Safety-Telegramm einfügen. Für den Zugriff benötigen Sie ein Antriebsobjekt.

Safety-Telegramm einfügen und auf Telegramm-Attribute zugreifen

```
using Siemens.Engineering.MC.DriveConfiguration;
```

Safety-Telegramm einfügen und auf Telegramm-Attribute zugreifen

```

TelegramComposition telegrams = drvObj.Telegrams;

//Add safety telegram
const int tgrmId = 30;
drvObj.Telegrams.Create(tgrmId);

// Get and set safety telegram attributes
uint Failsafe_FDestinationAddress =
(uint)safetyTelegram.GetAttribute("Failsafe_FDestinationAddress");
uint Failsafe_FSourceAddress =
(uint)safetyTelegram.GetAttribute("Failsafe_FSourceAddress");
uint Failsafe_FIODBNumber = (uint)safetyTelegram.GetAttribute("Failsafe_FIODBNumber");
string Failsafe_FIODBName = safetyTelegram.GetAttribute("Failsafe_FIODBName").ToString();
uint Failsafe_FMonitoringtime =
(uint)safetyTelegram.GetAttribute("Failsafe_FMonitoringtime");
uint Failsafe_ManualAssignmentFIODBNumber =
(uint)safetyTelegram.GetAttribute("Failsafe_ManualAssignmentFIODBNumber");
bool Failsafe_ManualAssignmentFMonitoringtime =
(bool)safetyTelegram.GetAttribute("Failsafe_ManualAssignmentFMonitoringtime");

// Set safety telegram attributes
safetyTelegram.SetAttribute("Failsafe_ManualAssignmentFIODBNumber", 1);
safetyTelegram.SetAttribute("Failsafe_ManualAssignmentFMonitoringtime", true);
safetyTelegram.SetAttribute("Failsafe_FIODBNumber", 40000);
safetyTelegram.SetAttribute("Failsafe_FMonitoringtime", 200);
safetyTelegram.SetAttribute("Failsafe_FDestinationAddress", 15);

const int newSafetyTelegramId= 900;
if (safetyTgrm.CanChangeTelegram(newSafetyTelegramId)) {
safetyTgrm.TelegramId = newSafetyTelegramId; }

```

Das folgende Beispiel zeigt, wie Sie ein Safety-Telegramm löschen.

Safety-Telegramm löschen

```

using Siemens.Engineering.MC.DriveConfiguration;
//Remove Safety telegram
drvObj.Telegrams.DeleteTelegram(TelegramType.SafetyTelegram);

```

Telegramme erweitern

Das folgende Beispiel zeigt, wie Sie eine Erweiterung einfügen und die Größe eines Standardtelegramms verändern. Für den Zugriff benötigen Sie ein Antriebsobjekt.

Erweiterung einfügen und die Größe eines Standardtelegramms ändern

```

using Siemens.Engineering.MC.DriveConfiguration;

```

Erweiterung einfügen und die Größe eines Standardtelegramms ändern

```

TelegramComposition telegrams = drvObj.Telegrams;
Telegram telegram = telegrams.Find(TelegramType.MainTelegram);

Console.WriteLine("The Cu has the telegram: " + telegram.TelogramId);
Console.WriteLine("The Setpoint channel-specific size of the telegram is: "
+ telegram.GetOutputSize());

foreach (var address in telegram.Addresses)
{
    if (address.IoType == AddressIoType.Output)
    {
        Console.WriteLine("The Setpoint channel-specific IO start address of
the telegram on the connected PLC is: " + address.StartAddress);
    }
    else if (address.IoType == AddressIoType.Input)
    {
        Console.WriteLine("The Actual value channel-specific IO start address
of the telegram on the connected PLC is: " + address.StartAddress);
    }
}

// Create an additional telegram
if (drvObj.Telegrams.CreateAdditionalTelegram(2, 4))
{
    drvObj.Telegrams.CreateAdditionalTelegram(2, 4);
}

// Add a 3 word extension to the main telegram
Telegram mainTelegram == drvObj.Telegrams.Find(TelegramType.MainTelegram);
Int32 newSize = mainTelegram.GetSize(AddressIoType.Input) + 3;
if (mainTelegram.CanChangeSize(AddressIoType.Input, newSize, true))
{
    mainTelegram.ChangeSize(AddressIoType.Input, newSize, true)
}

```

7.27.6.10 Beispiele für Namespace Siemens.Engineering.MC.Drives

Telegramme vorbereiten

Die Antriebskommunikation einer SINUMERIK-NCU erfolgt mithilfe von Telegrammen über die Subkomponente SINAMICS Integrated und ggf. über zusätzlich angeschlossene NX-Baugruppen.

Hinweis

Die SINUMERIK-NCU und ein SINAMICS Integrated befinden sich auf derselben Ebene im TIA Portal Openness-Objektmodell und erscheinen unter "DeviceComposition" als zwei unterschiedliche Geräte.

Um ein Telegramm zu konfigurieren, verwenden Sie "DriveObjectContainer". "DriveObjectContainer" ist ein Service des Antriebsobjekts (Geräteelement) vom aktuellen Kopfmodul (Geräteelement).

Um den Service "DriveObjectContainer" zu starten, navigieren Sie zu SINAMICS Integrated oder zum Kopfmodul der NX-Baugruppe. Die Hierarchie zwischen Geräten und Geräteelementen ist für SINAMICS Integrated und NX-Baugruppen identisch.

Das folgende Beispiel zeigt, wie Sie "DriveObjectContainer" über die Eigenschaft "Kopfmodul" finden:

DriveObjectContainer über Kopfmodul finden

```
foreach (Device device in project.Devices)
{
    foreach (DeviceItem deviceItem in device.DeviceItems)
    {
        if (deviceItem.Classification == DeviceItemClassifications.HM)
        {
            var driveObjectContainer = deviceItem.GetService<DriveObjectContainer>();
            // do something
        }
    }
}
```

Die SINUMERIK-NCU enthält ein SINAMICS Integrated Proxy-Objekt mit Referenzen auf SINAMICS Integrated.

Um auf ein SINAMICS Integrated-Gerät oder eine NX-Baugruppe zuzugreifen, navigieren Sie von der SINUMERIK-NCU über NCK zur DP Integrated-Schnittstelle. Anschließend ermitteln Sie das PROFIBUS-Mastersystem und navigieren zum verbundenen Slave.

Telegramme einfügen und löschen

Das folgende Beispiel zeigt, wie Sie ein Telegramm einfügen. Für den Zugriff benötigen Sie ein Antriebsobjekt.

Telegramm einfügen und auf Telegramm-Attribute zugreifen

```
using Siemens.Engineering.MC.Drives;
```

Telegramm einfügen und auf Telegramm-Attribute zugreifen

```

TelegramComposition telegrams = drvObj.Telegrams;

//Add telegram
const int tgrmNumber = 136;
drvObj.Telegrams.InsertTelegram(tgrmNumber);

//Find telegram
Telegram telegram = drvObj.Telegrams.Find(TelegramType.MainTelegram);

//Add safety telegram
const int tgrmNumber = 30;
drvObj.Telegrams.InsertSafetyTelegram(tgrmNumber);

//Find safety telegram
Telegram safetyTgrm = drvObj.Telegrams.Find(TelegramType.SafetyTelegram);

// Get and set safety telegram attributes
uint watchDogTime =
(uint)safetyTgrm.GetAttribute("Failsafe_FMonitoringtime");

safetyTgrm.SetAttribute("Failsafe_FMonitoringtime", 300);

const int newSafetyTelegramNumber= 902;
if (safetyTgrm.CanChangeTelegram(newSafetyTelegramNumber)) {
safetyTgrm.TelegramNumber = newSafetyTelegramNumber; }

//Add supplementary telegram
const int tgrmNumber = 701;
drvObj.Telegrams.InsertSupplementaryTelegram(tgrmNumber);

Telegram telegram =
drvObj.Telegrams.Find(TelegramType.SupplementaryTelegram);

```

Das folgende Beispiel zeigt, wie Sie ein Telegramm löschen.

Telegramm löschen

```

using Siemens.Engineering.MC.Drives;
//Remove safety telegram
drvObj.Telegrams.EraseTelegram(TelegramType.SafetyTelegram);

//Remove supplementary telegram
drvObj.Telegrams.EraseTelegram(TelegramType.SupplementaryTelegram);

```

Hinweis

Sie können ein Haupttelegramm (MainTelegram) ändern, aber nicht löschen.

Mit Safety-Telegrammen arbeiten

Das folgende Beispiel zeigt, wie Sie ein Safety-Telegramm einfügen. Für den Zugriff benötigen Sie ein Antriebsobjekt.

Safety-Telegramm einfügen und auf Telegramm-Attribute zugreifen

```
using Siemens.Engineering.MC.Drives;
TelegramComposition telegrams = drvObj.Telegrams;

//Add safety telegram
const int tgrmNumber = 30;
drvObj.Telegrams.InsertSafetyTelegram(tgrmNumber);

//Find safety telegram
Telegram safetyTgrm = drvObj.Telegrams.Find(TelegramType.SafetyTelegram);

// Get and set safety telegram attributes
uint Failsafe_FDestinationAddress =
(uint)safetyTelegram.GetAttribute("Failsafe_FDestinationAddress");
uint Failsafe_FSourceAddress =
(uint)safetyTelegram.GetAttribute("Failsafe_FSourceAddress");
uint Failsafe_FIODBNumber = (uint)safetyTelegram.GetAttribute("Failsafe_FIODBNumber");
string Failsafe_FIODBNName = safetyTelegram.GetAttribute("Failsafe_FIODBNName").ToString();
uint Failsafe_FMonitoringtime =
(uint)safetyTelegram.GetAttribute("Failsafe_FMonitoringtime");
uint Failsafe_ManualAssignmentFIODBNumber =
(uint)safetyTelegram.GetAttribute("Failsafe_ManualAssignmentFIODBNumber");
bool Failsafe_ManualAssignmentFMonitoringtime =
(bool)safetyTelegram.GetAttribute("Failsafe_ManualAssignmentFMonitoringtime");

// Set safety telegram attributes
safetyTelegram.SetAttribute("Failsafe_ManualAssignmentFIODBNumber", 1);
safetyTelegram.SetAttribute("Failsafe_ManualAssignmentFMonitoringtime", true);
safetyTelegram.SetAttribute("Failsafe_FIODBNumber", 40000);
safetyTelegram.SetAttribute("Failsafe_FMonitoringtime", 200);
safetyTelegram.SetAttribute("Failsafe_FDestinationAddress", 15);

const int newSafetyTelegramNumber= 900;
if (safetyTgrm.CanChangeTelegram(newSafetyTelegramNumber)) {
safetyTgrm.TelegramNumber = newSafetyTelegramNumber; }
```

Das folgende Beispiel zeigt, wie Sie ein Safety-Telegramm löschen.

Safety-Telegramm löschen

```
using Siemens.Engineering.MC.Drives;
//Remove Safety telegram
drvObj.Telegrams.EraseTelegram(TelegramType.SafetyTelegram);
```

Telegramme erweitern

Das folgende Beispiel zeigt, wie Sie eine Erweiterung einfügen und die Größe eines Standardtelegramms verändern. Für den Zugriff benötigen Sie ein Antriebsobjekt.

Erweiterung einfügen und die Größe eines Standardtelegramms ändern

```
using Siemens.Engineering.MC.Drives;
```

Erweiterung einfügen und die Größe eines Standardtelegramms ändern

```

TelegramComposition telegrams = drvObj.Telegrams;
Telegram telegram = telegrams.Find(TelegramType.MainTelegram);

Console.WriteLine("The Cu has the telegram: " + telegram.TelegramNumber);
Console.WriteLine("The Setpoint channel-specific size of the telegram is: "
+ telegram.GetOutputSize());

foreach (var address in telegram.Addresses)
{
    if (address.IoType == AddressIoType.Output)
    {
        Console.WriteLine("The Setpoint channel-specific IO start address of
the telegram on the connected PLC is: " + address.StartAddress);
    }
    else if (address.IoType == AddressIoType.Input)
    {
        Console.WriteLine("The Actual value channel-specific IO start address
of the telegram on the connected PLC is: " + address.StartAddress);
    }
}

// Add an additional telegram
if (drvObj.Telegrams.CanInsertAdditionalTelegram(2, 4))
{
    drvObj.Telegrams.InsertAdditionalTelegram(2, 4);
}

// Add a 3 word extension to the main telegram
Telegram mainTelegram == drvObj.Telegrams.Find(TelegramType.MainTelegram);
Int32 newSize = mainTelegram.GetSize(AddressIoType.Input) + 3;
if (mainTelegram.CanChangeSize(AddressIoType.Input, newSize, true))
{
    mainTelegram.ChangeSize(AddressIoType.Input, newSize, true)
}

```

7.28 Funktionen für Startdrive

7.28.1 Einführung

Über TIA Portal-Openness automatisieren Sie das Engineering und steuern das TIA Portal über ein von Ihnen selbst erstelltes Programm.

Für dieses selbst erstellte Programm finden Sie in dieser Hilfe viele Informationen und Beispielcodes. Auch für die TIA Portal-Anwendung "Startdrive" können Sie eigene Programme zusammenstellen und anwenden.

Bevor Sie für Startdrive aus den nachfolgend gelisteten Beispielcodes ein eigenes Programm zusammenstellen, beachten Sie bitte die übergreifenden Informationen zu Openness, die Sie unter folgenden Stichworten in dieser Hilfe finden:

- Voraussetzungen für TIA Portal Openness
- TIA Portal Openness installieren
- Auf das TIA Portal zugreifen
- TIA Portal Openness-Objektmodell
- Programmierschritte

7.28.2 Typeldentifier - Kennzeichnung der Komponenten

Jede Ausprägung einer beliebigen Startdrive-Komponente besitzt eine eindeutige Nummer, die als Typeldentifier bezeichnet wird. Im Openness-Programmcode können Sie diesen Typeldentifier nutzen, um eine Komponente eindeutig zu identifizieren und zu benennen.

In Startdrive ist die Anzeige des Typeldentifiers optional und standardmäßig deaktiviert.

Anzeige des Typeldentifiers in Startdrive aktivieren

1. Wählen Sie in der Startdrive-Projektansicht das Menü "Extras > Einstellungen".
Der Konfigurationsbereich "Einstellungen" wird geöffnet.
2. Wählen Sie in der Sekundärnavigation den Eintrag "Hardware-Konfiguration".
3. Aktivieren Sie die Option "Anzeige des Type Identifiers für Geräte und Module aktivieren".
Die Anzeige des Typeldentifiers ist nun aktiv.

Typeldentifier in Startdrive auslesen

Der Typeldentifier kann in einem Startdrive-Projekt bei aktiverter Anzeige an folgenden Stellen ausgelesen werden:

- Für alle Komponenten (im Inspektorenfenster):
- Für Control Units (beim Anlegen eines Antriebsgeräts)

Um den Typeldentifier für eine Komponente im Inspektorenfenster auszulesen, gehen Sie so vor:

1. Doppelklicken Sie in der Gerätesicht des Startdrive-Projekts auf die gewünschte Komponente.
Das Inspektorenfenster wird geöffnet. Die aktive Ausprägung der Komponente wird in der Liste angezeigt.
2. Den dazu gehörenden Typeldentifier finden in der äußeren rechten Spalte.
Beispiel: OrderNumber: 6SL3131-7TE23-6Axx
Kopieren Sie diesen Typeldentifier in Ihre Openness-Anwendung.

Um den TypelIdentifier für eine Control Unit auszulesen, gehen Sie so vor:

1. Doppelklicken Sie in der Startdrive-Projektnavigation auf "Neues Gerät hinzufügen". Der gleichnamige Dialog wird geöffnet.
2. Selektieren Sie die gewünschte Regelungsbaugruppe aus der Auswahl. Rechts in der Detailanzeige wird nun der TypelIdentifier (unter der Artikelnummer und Firmware-Nummer) für die entsprechende Control Unit angezeigt.
Beispiel: OrderNumber: 6SL3040-1MA01-0Axx/V5.2/S120
3. Kopieren Sie diesen TypelIdentifier in Ihre Openness-Anwendung.

7.28.3 Referenzen

7.28.3.1 AddressComposition

AddressComposition

Die Klasse `AddressComposition` repräsentiert die Adresse eines Telegramms.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Syntax** der Klasse:

```
public sealed class AddressComposition
```

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Beschreibung
IoType	AddressIoType (Seite 710)	Gibt Informationen zum Typ der Adresse zurück.
Context	AddressContext (Seite 710)	Gibt Informationen zum Kontext der Adresse zurück.
StartAddress	Int32	Gibt die Startadresse des Telegramms zurück oder legt diese fest.
Length	Int32	Gibt die Länge des Telegramms zurück.

Siehe auch

[TelegramType \(Seite 723\)](#)

7.28.3.2 AddressContext

AddressContext

Das Enum AddressContext enthält Informationen zum Kontext der Adresse.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Syntax** der Klasse:

```
public enum AddressContext
```

Die folgende Tabelle beschreibt die **Enum-Einträge**:

Name	Beschreibung
AddressContext.None	Für die Adresse wurde kein Kontext gefunden
AddressContext.Device	Kontext ist eine Gerätedresse
AddressContext.Head	Kontext ist eine Kopfadresse

7.28.3.3 AddressIoType

AddressIoType

Das Enum AddressIoType enthält Informationen zum Typ der Adresse.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Syntax** der Klasse:

```
public enum AddressIoType
```

Die folgende Tabelle beschreibt die **Enum-Einträge**:

Name	Beschreibung
AddressIoType.None	Der IO-Typ ist nicht anwendbar
AddressIoType.Input	Typ ist eine Eingangsadresse
AddressIoType.Output	Typ ist eine Ausgangsadresse
AddressIoType.Diagnosis	Typ ist eine Diagnoseadresse
AddressIoType.Substitute	Typ ist eine Ersatzadresse

7.28.3.4 ConfigurationEntry

ConfigurationEntry

Die Klasse ConfigurationEntry dient zum Speichern von Parameterdaten, die aus den ConfigurationEntryCompositions einer Motor- oder Geberkonfiguration ermittelt werden können.

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Beschreibung
EnumValueList	IDictionary<it, string>	Gibt eine Liste mit möglichen Werten des enum-Parameters zurück.
MaxValue	object	Maximalwert des ConfigurationEntry.
MinValue	object	Minimalwert des ConfigurationEntry.
Name	string	Name des ConfigurationEntry.
Number	int	Numerische Darstellung des Namens für ConfigurationEntry.
Description	string	Beschreibung des ConfigurationEntry.
Parent	IEngineeringObject	Engineering Object Model-Elternelement dieses Objekts.
Unit	string	Einheit des ConfigurationEntry.
Value	object	Wert des ConfigurationEntry.

7.28.3.5 DriveDomainFunctions

DriveDomainFunctions

Die Klasse DriveDomainFunctions dient der Wiederherstellung der Werkseinstellungen bzw. der Sicherung des RAM-Inhalts auf das ROM.

Sie kann nur auf ein OnlineDriveFunctionInterface-Objekt angewandt werden.

Bei G120-Antrieben kann nur auf das DriveDomainFunctions-Objekt zugegriffen werden, wenn das Power Module mit dem Gerät verkabelt ist. Andernfalls wird null oder eine Exception zurückgegeben.

Die folgende Tabelle beschreibt die **Methoden** der Klasse:

Name	Beschreibung
PerformFactoryReset	Diese Methode ist für die Wiederherstellung der Werkseinstellungen zuständig.
PerformRAMtoROMCopyAllDriveObject	Die Daten aller Antriebsobjekte werden aus dem RAM auf die Speicherkarte/Festplatte geschrieben.

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Beschreibung
Parent	IEngineeringObject	Engineering Object Model-Elternelement dieses Objekts.

7.28.3.6 DriveObject

DriveObject

Die Klasse **DriveObject** ermöglicht den Zugriff auf das Antriebsobjekt. Über das Antriebsobjekt ist z. B. der Zugriff auf Antriebsparameter oder das Telegramm möglich.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Syntax** der Klasse:

```
public sealed class DriveObject
```

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Beschreibung
Parameters	DriveParameter-Composition (Seite 715)	Gibt eine Liste mit den verfügbaren Parametern des Antriebsobjekts zurück.
Telegrams	TelegramComposition (Seite 722)	Gibt eine Liste mit den verfügbaren Telegrammen des Antriebsobjekts zurück. Die Liste kann mit der Klasse TelegramComposition verändert werden.

Siehe auch

[Antriebsobjekt ermitteln \(Seite 727\)](#)

7.28.3.7 DriveObjectActivation

DriveObjectActivation

Die Klasse **DriveObjectActivation** dient der Aktivierung der Module bzw. der Ermittlung des Modulstatus. Sie kann auf **DriveObjectFunctions** von **DriveFunctionInterface** oder **OnlineDriveFunctionInterface** angewendet werden.

Die folgende Tabelle beschreibt die **Methoden** der Klasse:

Name	Beschreibung
ChangeActivationState (DriveObjectActivationState)	Ändert den Aktivierungsstatus des Antriebsobjekts. Gibt false zurück, wenn die Operation nicht abgeschlossen werden kann. Mögliche Statuswerte: <ul style="list-style-type: none">• Deactivate• Activate• DeactivateAndNotPresent

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Beschreibung
ActivationState	DriveObjectActivationState	Gibt den aktuellen Statuswert zurück.
IsActive	Boolean	Gibt bei aktivem Antriebsobjekt true zurück.

7.28.3.8 DriveObjectContainer

DriveObjectContainer

Der `DriveObjectContainer` ist ein Service des Antriebsobjekts (`DeviceItem`) vom aktuellen Gerät (`Device`).

Die folgende Tabelle beschreibt die **Navigatoren** des `DriveObjectContainer`:

Name	Datentyp	Beschreibung
DriveObjects	DriveObjectComposition (Seite 715)	Gibt eine Liste mit den verfügbaren Antriebsobjekten zurück. Die Antriebsobjekte ermöglichen den Zugriff auf Antriebsparameter und Telegramme.

7.28.3.9 DriveObjectTypeHandler

DriveObjectTypeHandler

Die Klasse `DriveObjectTypeHandler` dient der Umstellung des Antriebsobjekt-Typs für jedes Antriebsobjekts und der Ermittlung des Antriebsobjekt-Typs sowie aller möglichen Antriebsobjekt-Typen des aktuellen Antriebsobjekts. Sie kann nur auf `DriveFunctionInterface` angewendet werden.

Die folgende Tabelle beschreibt die **Methoden** der Klasse:

Name	Beschreibung
<code>ChangeDriveObjectType((target)DriveObjectType)</code>	Ändert den aktuellen Typ des Antriebsobjekts auf einen auswählbaren neuen Typ. Gibt false zurück, wenn die Operation nicht abgeschlossen werden kann.

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Beschreibung
<code>PossibleDriveObjectTypes</code>	<code>DriveObjectTypeComposition</code>	Die Verwendung von <code>targetDriveObjectType</code> führt auf dem aktuellen Antriebsobjekt zu einer Exception.
<code>CurrentDriveObjectType</code>	<code>DriveObjectType</code>	Zeigt den aktuell zugewiesenen Antriebsobjekt-Typ.

7.28.3.10 DriveParameter

DriveParameter

Die Klasse `DriveParameter` ermöglicht den Zugriff auf einen Antriebsparameter. Nicht alle Antriebsparameter sind für den Zugriff über Openness verfügbar.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Syntax** der Klasse:

```
public sealed class DriveParameter
```

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Beschreibung
ArrayIndex	Int32	Gibt den Index eines Array-Parameters zurück. Wertebereich: 0-7FFF Bei Parametern ohne Array ist der Index -1 Beispiel p108[4].15 par.ArrayIndex ergibt 4
ArrayLength	Int32	Gibt die Anzahl der Array Elemente zurück. Bei Parametern ohne Array ist der Wert 0
Bits	DriveParameter-Composition (Seite 715)	Gibt ein Objekt <code>DriveParameter</code> für ein Bit des Parameters zurück. Damit kann von einem Bitparameter z. B. der Wert oder der Name des Bitparameters gelesen werden. Beispiel DriveParameter param133 = cu.Parameters.Find(133, 0); DriveParameter param133Bit1 = param133.Bits[1]; String paramName = param133Bit1.Name;
EnumValueList	IDictionary<int, string>	Gibt eine Liste mit möglichen Werten des enum Parameters zurück. Z. B. <1, [1] Quick commissioning> null, wenn der Parameter nicht vom Typ <code>Enum</code> ist.
.MaxValue	Object	Gibt den maximalen Wert für die aktuell gewählte Einheit zurück.
.MinValue	Object	Gibt den minimalen Wert für die aktuell gewählte Einheit zurück.
Name	string	Gibt den Namen des Parameters zurück. Z. B. "p108[0].2"
ParameterText	string	Gibt den Text der Kurzbeschreibung für den Parameter zurück.

Name	Datentyp	Beschreibung
Number	Int32	Gibt die Nummer des Parameters zurück. Beispiel p108[0].2 Rückgabewert ist 108
Unit	string	Gibt die Einheit des Parameters als Text zurück.
Value	Object	Gibt den Offline/Online Wert des Parameters zurück oder schreibt einen Wert auf den Parameter. Bei Schreibfehlern wird eine <code>EngineeringTargetInvocationException</code> ausgelöst. Beispiele <ul style="list-style-type: none"> • <code>P2080Bit6.Value = 0;</code> • <code>P2080Bit6.Value = cu.Parameters.Find("r19");</code> BICO-Quelle Der Parameter einer BICO-Quelle kann nur gelesen werden BICO-Signalsenken Mögliche Werte sind 0, 1 oder ein Objekt <code>DriveParameter</code> . Ein Objekt <code>DriveParameter</code> wird zurückgegeben, wenn die BICO-Signalsenke mit einem anderen Parameter verbunden ist. Siehe auch Beispiel BICO-Parameter lesen und schreiben (Seite 728).

Siehe auch

Parameter lesen und schreiben (Seite 745)

7.28.3.11 DriveParameterComposition

DriveParameterComposition

Die Klasse `DriveParameterComposition` ermöglicht den Zugriff auf Parameter des Antriebs. Nicht alle Antriebsparameter sind für den Zugriff über Openness zugelassen.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Syntax** der Klasse:

```
public sealed class DriveParameterComposition
```

Die folgende Tabelle beschreibt die **Methoden** der Klasse:

Name	Beschreibung
Find(string)	<p>Gibt das Objekt DriveParameter (Seite 714) zurück, das über den Namen gesucht wird.</p> <p>null, wenn der Parameter nicht gefunden wird</p> <p>Beispiel</p> <pre>Find("P108[1]");</pre>
Find(UInt16, Int32)	<p>Gibt das Objekt DriveParameter (Seite 714) zurück, das über den Parameter- und Array-Index gesucht wird.</p> <p>null, wenn der Parameter nicht gefunden wird</p> <p>Beispiele</p> <ul style="list-style-type: none"> • cu.Find(108, 1); • cu.Find(51, -1);
WriteParameters(IEnumerable<string>, IEnumerable<string>, bool)	<p>Schreibt Werte in Parameter.</p> <p>Bei der Einstellung ignoreErrors = true wird im Fehlerfall versucht, alle Werte zu schreiben und am Ende eine EngineeringTargetInvocationException ausgelöst.</p> <p>Bei SINAMICS-G Antrieben können Parameterwerte nur bei konfiguriertem Power Module (PM) geschrieben werden.</p> <p>Beispiel</p> <pre>List<string> names = new List<string>(); List<string> values = new List<string>(); names.add("p300[0]"); values.add("17"); names.add("p5391[0]"); values.add("20"); cu.WriteParameters(names, values, true);</pre>

7.28.3.12 EncoderConfiguration

EncoderConfiguration

Die Klasse `EncoderConfiguration` speichert Daten von Nicht-Siemens-Gebern (-Encodern).

- Das Objekt `ConfigurationEntryComposition` muss vom Anwender ausgefüllt werden.
- Das Objekt `RequiredConfigurationEntries` muss ebenfalls ausgefüllt werden.

Namespace: Siemens.Engineering.MC.Drives

Siemens.Engineering.MC.Drives.DFI

Siemens.Engineering.MC.Drives.Enum

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Beschreibung
RequiredConfigurationEntries	ConfigurationEntryComposition	Zugängliche Konfigurationseingänge der Motorkonfiguration.
Parent	IEngineeringObject	Engineering Object Model-Elternelement dieses Objekts.

7.28.3.13 HardwareProjection

Die Klasse `HardwareProjection` ist verantwortlich für die Inbetriebnahme des Motors und des Gebers. Das Objekt kann bei `DriveFunctionInterface` und `OnlineDriveFunctionInterface` gefunden werden.

Die Projektierung kann bei G120-Antrieben für Motoren und Gebern online wie offline durchgeführt werden. Bei S120-Antrieben ist die Projektierung dagegen nur offline möglich.

Bei G120-Antriebsgeräten kann auf das Objekt `HardwareProjection` nur zugegriffen werden, wenn das Power Module im Antriebsgerät eingesteckt ist. Andernfalls wird beim Aufruf der Funktionen `HardwareProjection` eine null oder eine Exception zurückgegeben.

Bei einer Offline-Projektierung nutzen Sie die Hardware-Projektierung des `DriveFunctionInterface`. Bei einer Online-Projektierung nutzen Sie Hardware-Projektierung des `OnlineDriveFunctionInterface`.

Namespace: Siemens.Engineering.MC.Drives
 Siemens.Engineering.MC.Drives.DFI
 Siemens.Engineering.MC.Drives.Enums

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Methoden** der Klasse:

Name	Beschreibung
<code>SetMotorType(MotorType type, ushort driveDataSet)</code>	Stellt den Motor-Typ an der Control Unit ein (nur bei G120).
<code>GetCurrentMotorConfiguration(ushort driveDataSet)</code>	Ermittelt abhängig von der Datensatznummer des Antriebs den aktuell existierenden Konfigurationsbereich.
<code>ProjectMotorConfiguration(MotorConfiguration motConfig, ushort driveDataSet)</code>	Projektiert die Motorkonfiguration eines Antriebsgeräts abhängig von der Datensatznummer des Antriebs.
<code>SetEncoder(EncoderType type, EncoderInterface interfaceType, AbsoluteIncrementalFlag absIncFlag, RotaryLinearFlag rotLinFlag, ushort encDataSet)</code>	Stellt den Geber (Encoder) an der Control Unit ein (nur bei G120).
<code>GetCurrentEncoderConfiguration(ushort encDataSet)</code>	Ermittelt abhängig von der Datensatznummer des Gebers den aktuell existierenden Konfigurationsbereich.
<code>ProjectEncoderConfiguration(EncoderConfiguration encConfig, ushort encDataSet)</code>	Projektiert die Motorkonfiguration eines Antriebsgeräts abhängig von der Datensatznummer des Gebers (Encoders).

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Beschreibung
Parent	IEngineeringObject	Engineering Object Model-Elternelement dieses Objekts.

7.28.3.14 MotorConfiguration

MotorConfiguration

Die Klasse `MotorConfiguration` ist für die Inbetriebnahme von Motoren und Gebern verantwortlich.

Namespace: Siemens.Engineering.MC.Drives
 Siemens.Engineering.MC.Drives.DFI
 Siemens.Engineering.MC.Drives.Enums

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Verhalten bei Siemens-Motoren:

Um den Motor-Typ einzustellen muss die Aktion `SetMotortype` aufgerufen werden. Diese Aktion besteht aus der Enum `MotorType` und einer Nummer, die die `DriveDatasetNumber` repräsentiert.

Die folgende Tabelle beschreibt die Enums:

Enum-Name	Beschreibung
NoMotor	0: No motor
InductionMotor	1: Induction motor
SynchronousMotor	2: Synchronous motor
NoCodeNumber1LE1InductionMotor	10: 1LE1 induction motor (not a code number)
NoCodeNumber1LG6InductionMotor	13: 1LG6 induction motor (not a code number)
NoCodeNumber1xx1SIMOTICSFDInductionMotor	14: 1xx1 SIMOTICS FD induction motor (not a code number)
NoCodeNumber1LA7InductionMotorNoCodeNumber	17: 1LA7 induction motor (not a code number)
MotorSeriesNumber1LA81PQ8StandardInduction	18: 1LA8 / 1PQ8 standard induction motor series
NoCodeNumber1LA9InductionMotor	19: 1LA9 induction motor (not a code number)

Verhalten bei Nicht-Siemens-Motoren:

Die Klasse `MotorConfiguration` wird zum Speichern von Motordaten von Nicht-Siemens-Motoren benutzt. Es enthält 2 Objekte `ConfigurationEntryComposition`, die bei der Inbetriebnahme mit den entsprechenden Daten des Motors befüllt werden müssen. Das Objekt `RequiredConfigurationEntries` muss ebenfalls ausgefüllt werden. Das Objekt `OptionalConfigurationEntries` muss nicht ausgefüllt werden.

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Beschreibung
RequiredConfigurationEntries	ConfigurationEntryComposition	Zugängliche Konfigurationseingänge dieser Motorkonfiguration.
OptionalConfigurationEntries	ConfigurationEntryComposition	Zugängliche Konfigurationseingänge dieser Motorkonfiguration.
Parent	IEngineeringObject	Engineering Object Model-Elternelement dieses Objekts.

7.28.3.15 OnlineDriveObject

OnlineDriveObject

Die Klasse `OnlineDriveObject` ermöglicht den Online-Zugriff auf das Antriebsobjekt. Über das Antriebsobjekt ist der Zugriff auf Antriebsparameter möglich.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Syntax** der Klasse:

```
public sealed class OnlineDriveObject
```

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Beschreibung
Parameters	DriveParameter-Composition (Seite 715)	Gibt eine Liste mit den verfügbaren Parametern des Online-Antriebsobjekts zurück. null, wenn der Modus "Offline" ist. Im Offline-Modus wird beim Aufruf einer Methode oder bei einem Schreibzugriff auf einen Parameter eine Exception ausgelöst.

Siehe auch

Antriebsobjekt ermitteln (Seite 727)

Parameter lesen und schreiben (Seite 745)

Parameter online lesen und schreiben (Seite 747)

7.28.3.16 OnlineDriveObjectContainer

OnlineDriveObjectContainer

Der `OnlineDriveObjectContainer` ist ein Service des Antriebsobjekts (`DeviceItem`) vom aktuellen Gerät (`Device`).

Die folgende Tabelle beschreibt die **Navigatoren** des `OnlineDriveObjectContainer`:

Name	Datentyp	Beschreibung
<code>OnlineDriveObjects</code>	<code>OnlineDriveObjectComposition</code>	Gibt eine Liste mit den verfügbaren Online-Antriebsobjekten (<code>OnlineDriveObject</code> (Seite 719)) zurück. Die Antriebsobjekte ermöglichen den Zugriff auf Antriebsparameter.

7.28.3.17 StartDriveDownloadCheckConfiguration

StartDriveDownloadCheckConfiguration

Die Klasse `StartDriveDownloadCheckConfiguration` ist abgeleitet von der Klasse `DownloadCheckConfiguration` und hat die gleichen Eigenschaften. Die Klasse `DownloadCheckConfiguration` ist in der Standard-Openness-Hilfe beschrieben.

Diese Klasse stellt die Konfigurationseinstellungen über Optionskästchen vom Anwender zur Verfügung.

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Beschreibung
<code>Checked</code>	<code>bool</code>	Gibt die aktuelle Einstellung der Konfiguration zurück oder aktiviert/deaktiviert die Konfiguration.

Siehe auch

[Download \(Seite 729\)](#)

7.28.3.18 SafetyTelegram

SafetyTelegram

Die Klasse `SafetyTelegram` steht für ein Telegramm des Antriebsobjekts. Bei Fehlern in den Schreib-Attributen erscheint die Exception `EngineeringTargetInvocationException`.

Namespace: `Siemens.Engineering.MC.Drives`

Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Beschreibung
<code>TelegramNumber</code>	<code>Int32</code>	Nummer der Haupttelegramms. Freie Telegramme werden mit dem Wert 999 gekennzeichnet.
<code>Type</code>	<code>TelegramType</code> (Seite 723)	Telegrammtyp der als Enum "TelegramType" zurückgegeben wird.

Name	Datentyp	Beschreibung
Addresses	AddressComposition (Seite 709)	Zusammenstellung aller Adressen eines Telegramms.
PKW	Telegram (Seite 721)	Zusammenstellung aller PKW-Kanäle des Telegramms. null, wenn das Telegramm keinen PKW-Teil hat.

7.28.3.19 Telegram

Telegram

Die Klasse `Telegram` ermöglicht den Zugriff auf den Aufbau eines Telegramms von einem Antriebsobjekt.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Syntax** der Klasse:

```
public sealed class Telegram
```

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Beschreibung
TelegramNumber	Int32	Gibt die Nummer des Haupttelegramms zurück oder legt die Nummer fest. Ein freies Telegramm hat die Nummer 999.
Type	TelegramType (Seite 723)	Gibt den Typ des Telegramms als <code>Enum TelegramType</code> zurück.
Addresses	AddressComposition (Seite 709)	Gibt eine <code>AddressComposition</code> mit Informationen zur Adresse zurück.
PKW	Telegram	Gibt den Kanal <code>PKW</code> als <code>Telegram</code> zurück. null, wenn die Eigenschaft nicht verfügbar ist Ein Telegramm mit <code>PKW</code> ist z. B. Telegramm 353.

Die folgende Tabelle beschreibt die **Methoden** der Klasse:

Name	Beschreibung
CanChangeTelegram(Int32)	Gibt <code>true</code> zurück, wenn das Telegramm in den parametrierten Standardtyp geändert werden kann.
GetSize(AddressIoType (Seite 710))	Gibt die Größe der Eingänge oder Ausgänge des Telegramms zurück.

Name	Beschreibung
CanChangeSize(AddressIoType (Seite 710), Int32, bool)	Gibt true zurück, wenn die Größe des Telegramms wie parametriert geändert werden kann. Standardtelegramme können nur vergrößert werden. Das Beibehalten der bisherigen Telegrammadresse wird berücksichtigt, wenn die Option mit true parametriert ist.
ChangeSize(AddressIoType (Seite 710), Int32, bool)	Gibt true zurück, wenn die Größe des Telegramms wie parametriert geändert werden konnte. Das Beibehalten der bisherigen Telegrammadresse wird berücksichtigt, wenn die Option mit true parametriert ist.

7.28.3.20 TelegramComposition

TelegramComposition

Die Klasse `TelegramComposition` ermöglicht den Zugriff auf die Telegramme eines Antriebsobjekts. Der Aufbau eines Telegramms kann über die Klasse `Telegram` (Seite 721) ausgelesen werden.

Beachten Sie, dass durch die Klasse `TelegramComposition` referenzierte Objekte ungültig werden können. Z. B. wird nach Änderung der Telegrammgröße das Objekt `Telegram` (Seite 721) ungültig.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Syntax** der Klasse:

```
public sealed class TelegramComposition
```

Die folgende Tabelle beschreibt die **Methoden** der Klasse:

Name	Beschreibung
CanInsertAdditionalTelegram(Int32, Int32)	Gibt true zurück, wenn eine Erweiterung entsprechend den parametrierten Größen (Eingangs- und Ausgangsgröße) erstellt werden kann.
CanInsertTorqueTelegram(Int32, telegramNumber)	Prüft beim Hinzufügen eines neuen Torque-Telegramms mit einer bereits vorhandenen Telegramm-Nummer, ob dies möglich ist.
CanInsertSupplementaryTelegram(Int32)	Gibt true zurück, wenn ein Zusatztelegramm entsprechend der parametrierten Telegrammnummer erstellt werden kann.

Name	Beschreibung
EraseTelegram(TelegramType e)	<p>Löscht ein Telegramm mit bekanntem Telegrammtyp aus dem Antriebsobjekt.</p> <p>Gibt <code>true</code> zurück, wenn das parametrierte Telegramm gelöscht werden konnte.</p> <p>Wird ein Torque-Telegramm als Typ verwendet, so wird das Objekt "torqueTelegram" gelöscht.</p> <p>Wird ein Safety Integrated-Telegramm als Typ verwendet, so wird das Objekt "safetyTelegram" gelöscht.</p> <p>Standardtelegramme können nicht gelöscht werden.</p> <p>Im Fehlerfall wird eine <code>EngineeringTargetInvocationException</code> ausgelöst.</p>
Find(TelegramType)	<p>Gibt das Objekt <code>Telegram</code> (Seite 721) zurück, wenn es über den parametrierten Telegrammtyp oder Safety Integrated-Telegrammtyp gefunden werden konnte.</p> <p><code>null</code>, wenn das Telegramm nicht gefunden wird.</p> <p>Wird ein Torque-Telegramm als Typ verwendet, wird das Objekt <code>torqueTelegram</code> zurück gegeben, wenn es existiert.</p> <p>Wird ein Safety Integrated-Telegramm als Typ verwendet, wird das Objekt <code>safetyTelegram</code> zurück gegeben, wenn es existiert.</p> <p>Beispiel</p> <pre>Telegram telegram = telegrams.Find(TelegramType.MainTelegram);</pre>
InsertAdditionalTelegram(Int32, Int32)	<p>Erstellt für das Antriebsobjekt eine Erweiterung entsprechend den parametrierten Größen und gibt <code>true</code> zurück, wenn die Erweiterung eingefügt werden konnte.</p> <p>Im Fehlerfall wird eine <code>EngineeringTargetInvocationException</code> ausgelöst.</p>
InsertTorqueTelegram(Int32, telegramNumber)	Fügt einem Antriebsobjekt ein neues Torque-Telegramm mit einer bereits vorhandenen Telegramm-Nummer zu.
InsertSafetyTelegram(Int32, telegramNumber)	Fügt ein Safety Integrated-Telegramm mit seiner vorgegebenen Telegrammnummer einem Antriebsobjekt hinzu.
InsertSupplementaryTelegram(Int32)	<p>Erstellt das Zusatztelegramm mit der parametrierten Telegrammnummer und gibt <code>true</code> zurück, wenn das Telegramm eingefügt werden konnte.</p> <p>Im Fehlerfall wird eine <code>EngineeringTargetInvocationException</code> ausgelöst.</p>

7.28.3.21 **TelegramType**

TelegramType

Das Enum `TelegramType` enthält vordefinierte Telegrammtypen.

Namespace: Siemens.Engineering.MC.Drives

Assembly: Siemens.Engineering.MC.Drives in Siemens.Engineering.dll

Die folgende Tabelle beschreibt die **Syntax** der Klasse:

```
public enum TelegramType
```

Die folgende Tabelle beschreibt die **Enum-Einträge**:

Name	Beschreibung
MainTelegram	ID des Haupttelegramms
SupplementaryTelegram	ID des Zusatztelegramms
AdditionalTelegram	ID einer Erweiterung

7.28.3.22 TorqueTelegram

TorqueTelegram

Die Klasse `TorqueTelegram` steht für ein Telegramm des Antriebsobjekts. Bei Fehlern in den Schreib-Attributen erscheint die Exception `EngineeringTargetInvocationException`.

Namespace: `Siemens.Engineering.MC.Drives`

Assembly: `Siemens.Engineering.MC.Drives` in `Siemens.Engineering.dll`

Die folgende Tabelle beschreibt die **Eigenschaften** der Klasse:

Name	Datentyp	Beschreibung
TelegramNumber	Int32	Nummer der Telegramms.
Type	TelegramType (Seite 723)	Telegrammtyp der als Enum "TelegramType" zurückgegeben wird.
Addresses	AddressComposition (Seite 709)	Zusammenstellung aller Adressen eines Telegramms.
PKW	Telegram (Seite 721)	Zusammenstellung aller PKW-Kanäle des Telegramms. null, wenn das Telegramm keinen PKW-Teil hat.

7.28.4 Codebeispiele

Die folgenden Codebeispiele beschreiben die grundsätzliche Vorgehensweise für verschiedene Anwendungsfälle. Der Code ist nicht zwingend vollständig und kompilierbar.

7.28.4.1 Aktivierungsstatus ermitteln

Die folgenden Beispiele zeigen, wie Sie den Aktivierungs-Status für S120-Antriebe offline oder online ermitteln können:

Aktivierungs-Status eines S120-Antriebs offline ermitteln

```
using Siemens.Engineering.MC.Drives;
```

Aktivierungs-Status eines S120-Antriebs offline ermitteln

```
DriveFunctionInterface dfi = ...
DriveObjectActivation driveObjectActivation =
dfi.DriveObjectFunctions.DriveObjectActivation;
//driveObjectActivation can be null in case of the actual driveobject does not support
activation.

//change activation state
driveObjectActivation.ChangeActivationState(DriveObjectActivationState.Deactivate);

//get the activation state
DriveObjectActivationState activationState = driveObjectActivation.ActivationState;

//get the Is Active property
bool isActive = driveObjectActivation.IsActive;
```

Aktivierungs-Status eines S120-Antriebs online ermitteln

```
using Siemens.Engineering.MC.Drives;
OnlineDriveFunctionInterface onlinedfi = ...
DriveObjectActivation driveObjectActivation = onlinedfi.DriveObjectActivation;
//driveObjectActivation can be null in case of the actual driveobject does not support
activation in online.

//change activation state
driveObjectActivation.ChangeActivationState(DriveObjectActivationState.Deactivate);

//get the activation state
DriveObjectActivationState activationState = driveObjectActivation.ActivationState;

//get the IsActive property
bool isActive = driveObjectActivation.IsActive;
```

7.28.4.2 Antriebsfunktionen durchführen

Die folgenden Beispiele zeigen, wie Sie auf einfache Weise offline oder online auf Antriebsfunktionen zugreifen können:

Antriebsfunktionen offline durchführen

```
using Siemens.Engineering.MC.Drives;
DriveObject driveObject = ...
DriveFunctionInterface dfi = driveObject.GetService<DriveFunctionInterface>();

// dfi can be null in case of the actual driveobject does not support it.
```

Antriebsfunktionen online durchführen

```
using Siemens.Engineering.MC.Drives;
```

Antriebsfunktionen online durchführen

```
OnlineDriveObject onlineDriveObject = ...
OnlineDriveFunctionInterface onlineDfi =
onlineDriveObject.GetService<OnlineDriveFunctionInterface>();

// onlineDfi can be null in case of the actual onlineDriveObject
// does not support it or if the device is offline.
```

7.28.4.3 Antriebsgerät anlegen

Mit der Methode `CreateWithItem()` der Collection `Devices` legen Sie ein Antriebsgerät an.

Die Antriebsgeräte werden über Parameter der Methode spezifiziert. Das Format der Parameter ist nachfolgend beschrieben.

Antriebsgerät G120 anlegen

Format der Parameter für G120:

```
CreateWithItem(@"OrderNumber: mlfb / FirmwareVersion /",
"NameOfTheDevice", positionNumber)
```

Das folgende Beispiel zeigt, wie Sie ein G120-Antriebsgerät anlegen.

Antriebsgerät G120 anlegen

```
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
Project tiaproject= portal.Projects.Open("..."); //The path of the project

Device s120Device =
tiaproject.Devices.CreateWithItem(@"OrderNumber:6SL3246-0BA22-1FA0/4.7.6/"
, "Device_0", null);
```

Antriebsgerät S120, S150, MV, G130, G150 anlegen

Format der Parameter für S120, S150, MV, G130, G150:

```
CreateWithItem(@"OrderNumber: mlfb / FirmwareVersion /
AdditionalTypeIdentifier", "NameOfTheDevice", positionNumber)
```

Mögliche Werte für AdditionalTypeIdentifier:

- Leerstring (Z. B. für G120)
- S120
- S150
- MV
- G130
- G150

Das folgende Beispiel zeigt, wie Sie ein S120-Antriebsgerät anlegen.

Antriebsgerät S120 anlegen

```
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
Project tiaproject= portal.Projects.Open("..."); //The path of the project

Device s120Device =
tiaproject.Devices.CreateWithItem(@"OrderNumber:6SL3040-1MA01-0Axx/V4.8/
S120", "Device_0", null);
```

7.28.4.4 Antriebskomponente anlegen

Mit der Methode `PlugNew()` eines `Device` Objekts legen Sie eine Antriebskomponente für ein Antriebsgerät an.

Das folgende Beispiel zeigt, wie Sie eine Antriebskomponente anlegen.

Motormodul anlegen

```
DeviceItem subModul = sdrDevice.PlugNew(@"OrderNumber:6SL3xxx-xxxxx-xxxx",
"MotorModul", 65535);
```

7.28.4.5 Antriebsobjekt ermitteln

Die folgenden Beispiele zeigen, wie Sie Antriebsobjekte offline und online ermitteln.

Offline-Antriebsobjekt ermitteln

```
using Siemens.Engineering.MC.Drives;
//G device
Project project = portal.Projects.Open("..."); //Destination folder to
open the project
DeviceItem item = project.Devices[0].Items[0].Items[0];
DriveObject driveObject =
item.GetService<DriveObjectContainer>().DriveObjects[0];

//S device
Project project = portal.Projects.Open("..."); //Destination folder to
open the project
DeviceItem item = project.Devices[0].Items[0];
DriveObject driveObject =
item.GetService<DriveObjectContainer>().DriveObjects[0];
```

Online-Antriebsobjekt ermitteln

```
using Siemens.Engineering.MC.Drives;
```

Online-Antriebsobjekt ermitteln

```
//G device
Project project = portal.Projects.Open("..."); //Destination folder to
open the project
DeviceItem item = project.Devices[0].Items[0].Items[0];
OnlineDriveObject onlineDriveObject =
item.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];

//S device
Project project = portal.Projects.Open("..."); //Destination folder to
open the project
DeviceItem item = project.Devices[0].Items[0];
OnlineDriveObject onlineDriveObject =
item.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];
```

7.28.4.6 Antriebsobjekt-Typ ermitteln

Das folgende Beispiel zeigt, wie Sie den aktuellen Antriebsobjekt-Typ und die alternativ einstellbaren Typen ermitteln können.

Antriebsobjekt-Typen ermitteln

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;

DriveObject driveObject = ...
DriveFunctionInterface dfi = driveObject.GetService<DriveFunctionInterface>();
DriveObjectTypeHandler driveObjectTypeHandler =
dfi.DriveObjectFunctions.DriveObjectTypeHandler;

// dfi can be null in case of the actual driveobject does not support it.

// driveObjectTypeHandler can be null, if the actual driveObject does not support it.

// Get the possible drive object types on the drive object.
DriveObjectTypeComposition possibleDriveObjectTypes =
driveObjectTypeHandler.PossibleDriveObjectTypes;

// Get the current drive object type on the drive object.
DriveObjectType currentDriveObjectType = driveObjectTypeHandler.CurrentDriveObjectType;

//Call the ChangeDriveObjectType method with the current drive object type.
//The method parameter should be the target drive object type.
driveObjectTypeHandler.ChangeDriveObjectType(possibleDriveObjectTypes[0]);
```

7.28.4.7 BICO-Parameter lesen und schreiben

Das folgende Beispiel zeigt, wie Sie Werte von BICO-Parametern lesen und schreiben. Für den Zugriff benötigen Sie ein Antriebsobjekt.

BICO-Parameter lesen

```
using Siemens.Engineering.MC.Drives;
```

BICO-Parameter lesen

```
DriveParameter bicoSink= driveObject.Parameters.Find("p681");
if(bicoSink!=null)
{
    if(bicoSink.Value is DriveParameter)
    {
        DriveParamter bicoSourceValue = bicoSink.Value as DriveParameter;
        Console.WriteLine("The value of parameter " + bicoSink.Name + ":" + bicoSource.Name + " " + bicoSource.ParameterText);
    }
    else if (bicoSink.Value == null)
    {
        Console.WriteLine("Value contains an invalid connection or the source parameter is not accessible via Openness");
    }
    else
    {
        Console.WriteLine("The value of parameter " + bicoSink.Name + ":" + bicosink.Value.ToString());
    }
}
```

BICO-Parameter schreiben

```
using Siemens.Engineering.MC.Drives;

DriveParameter bicoSource= driveObject.Parameters.Find("r19");
DriveParameter bicoSink = driveObject.Parameters.Find("p738");
if(bicoSource != null)
{
    try
    {
        bicoSink.Value = bicoSource;
    }
    catch(UserException ex)
    {
        Console.WriteLine("Write failure :" + ex.Message);
    }
}
```

7.28.4.8 Download

Nach dem Start des Download müssen Sie die Konfigurationseinstellungen anpassen bzw. bestätigen. Die Konfigurationseinstellungen werden als Kindobjekte vom Objekt `DownloadConfiguration` zur Verfügung gestellt und gibt es in drei verschiedenen Typen:

- `StartDriveDownloadCheckConfiguration`
- `DownloadSelectionConfiguration`
- `DownloadPasswordConfiguration`

Das folgende Beispiele zeigt die Auswertung der verschiedenen Typen von Konfigurationseinstellungen im PreDownload Delegate.

Auswertung der Konfigurationseinstellungen nach Start des Download

```
using Siemens.Engineering.Download;
using Siemens.Engineering.Online;
static void PreDownload(DownloadConfiguration configuration)
{
    Console.WriteLine(configuration.Message);
    StartDriveDownloadCheckConfiguration sdcc = configuration as
StartDriveDownloadCheckConfiguration;
    if (sdcc != null)
    {
        sdcc.Checked = true;
        return;
    }

    DownloadPasswordConfiguration downloadPasswordConfiguration =
configuration as DownloadPasswordConfiguration;

    if (downloadPasswordConfiguration != null)
    {
        SecureString s = new SecureString();
        string passwordText = "password";
        foreach (var str in passwordText)
        {
            s.AppendChar(str);
        }

        downloadPasswordConfiguration.SetPassword(s);
        return;
    }

    DownloadSelectionConfiguration downloadSelectionConfiguration =
configuration as DownloadSelectionConfiguration;

    if (downloadSelectionConfiguration != null)
    {
        downloadSelectionConfiguration.SelectedIndex = 0;
        return;
    }
}
```

Das folgende Beispiele zeigt, wie Sie ein Projekt in das Gerät laden.

Download in das Gerät S120

```
using Siemens.Engineering.Download;
using Siemens.Engineering.Online;
```

Download in das Gerät S120

```

try
{
    DeviceItem item = ... //device item of the CU (e.g. :
project.Devices[0].Items[0].Items[0])
    DownloadProvider downloadProvider = item.GetService<DownloadProvider>();

    DownloadConfigurationDelegate pre = PreDownload;
    DownloadConfigurationDelegate post = PostDownload;

    ConnectionConfiguration connConfiguration =
downloadProvider.Configuration;
    ConfigurationMode configurationMode = connConfiguration.Modes.Find("PN/
IE");
    ConfigurationPcInterface pcInterface = configurationMode.PcInterfaces[0];
    ConfigurationSubnet subnet = pcInterface.Subnets.Find(/*subnet name*/);
    IConfiguration configuration = subnet.Addresses.Find(/*IP address of the
device*/);
    downloadProvider.Download(configuration, pre, post,
DownloadOptions.Software);
}
catch (Exception e)
{
    Console.WriteLine(e.Message);
}

//configuration handling before download
static void PreDownload(DownloadConfiguration configuration)
{
    Console.WriteLine(configuration.Message);
    StartDriveDownloadCheckConfiguration dcc = configuration as
StartDriveDownloadCheckConfiguration ;
    if (dcc != null)
    {
        dcc.Checked = true;
    }
}

//configuration handling after download
static void PostDownload(DownloadConfiguration configuration)
{
    Console.WriteLine(configuration.Message);
}

```

7.28.4.9 DRIVE-CLiQ-Verbindungen bearbeiten

Das folgende Beispiel zeigt, wie Sie mit Openness DRIVE-CLiQ-Verbindungen bearbeiten können.

DRIVE-CLiQ-Verbindungen bearbeiten

```
TiaPortal portal = new TiaPortal(TiaPortalMode.WithUserInterface);
```

DRIVE-CLiQ-Verbindungen bearbeiten

```

Project tiaproject = portal.Projects.Open(new FileInfo(@"C:\Users\testUser
\Documents\Automation\Project109\Project109.ap15")); //The path of the
project

Device device =
tiaproject.Devices.CreateWithItem(@"OrderNumber:6SL3040-1MA01-0Axx/V4.8/
S120", "Device_0", null); //Create the device

DeviceItem subModul = device.PlugNew(@"OrderNumber:6SL3x2x-1xxxx-xxxx",
"", 65535); //Plug a submodul (in this case : Motor modul)

DeviceItem cu = device.DeviceItems[1]; // The CU is always the 1
indexed in the DeviceItems of the Device

DeviceItem subModulDQInterface = subModul.DeviceItems[0]; //We need
the DQ interface from the submodul
DeviceItem cuDQInterface = cu.DeviceItems[3]; //This is the
DQ interface of the CU

NetworkPort subModulDQ =
((IEngineeringServiceProvider)subModulDQInterface.DeviceItems[0]).GetService<Net
workPort>(); //We need the DriveCliq port of the DQ interface from
the submodul
NetworkPort cuDQ =
((IEngineeringServiceProvider)cuDQInterface.DeviceItems[0]).GetService<Net
workPort>(); //We need the DriveCliq port of the DQ interface
from the CU

cuDQ.DisconnectFromPort(subModulDQ); //Delete the connection between the
two ports (automatically created when plugging a modul)
cuDQ.ConnectToPort(subModulDQ); //Create a new connection

```

7.28.4.10 Erste Schritte in Startdrive durchführen

Das folgende Beispiel zeigt, wie Sie einen laufenden TIA-Portal-Prozess lokalisieren oder erzeugen können.

TIA Portal-Prozess finden oder erzeugen

```

using Siemens.Engineering;

// Get the list of the running TIA Portal processes.
IList<TiaPortalProcess> procs = TiaPortal.GetProcesses();
TiaPortal portal;
// When there is at least one running TIA Portal, we will attach to the first from the list.
if (procs.Count != 0)
{
    portal = procs[0].Attach();
}
// When there is no running TIA Portal, we create one.
else
{
    portal = new TiaPortal(TiaPortalMode.WithUserInterface);
}

```

Das folgende Beispiel zeigt, wie Sie ein Startdrive-Projekt lokalisieren oder erzeugen können.

Startdrive-Projekt finden oder erzeugen

```
using Siemens.Engineering;
Project project;
// When the portal has one project, we save it in a variable.
if (portal.Projects.Count == 1)
{
    project = portal.Projects[0];
}
// When there is no existing project, we create one with a specific path, and the actual time
else
{
    project = portal.Projects.Create(
        new DirectoryInfo(@"C:\Projects\Project_" + DateTime.Now.Ticks),
        DateTime.Now.Ticks.ToString());
}
```

Das folgende Beispiel zeigt, wie Sie ermitteln können, ob eine bestimmte Startdrive-Variante (Package und Version) installiert ist.

Ermitteln, ob die gewünschte Startdrive-Version installiert ist

```
using Siemens.Engineering;
if (tiaProcess.InstalledSoftware.Any(sw => sw.Name.Equals("SINAMICS Startdrive Advanced") && sw.Version.Equals("V15"))){ Console.WriteLine("Startdrive is available");}
// "V15" is the current startdrive version started at December 2017.
// "V15.1" will be the current startdrive version beginning with December 2018.
// "SINAMICS Startdrive Basic" and "SINAMICS Startdrive Advanced" are the 2 possible
startdrive function packages.
```

7.28.4.11 Geber-Typ festlegen

Das folgende Beispiel zeigt, wie Sie offline den Geber-Typ bzw. die Geber-Datensatznummer einstellen können.

Geber-Typ bzw. Geber-Datensatznummer offline über Hardware-Projektierung einstellen

```
using Siemens.Engineering.MC.Drives
using Siemens.Engineering.MC.Drives.DFI
using Siemens.Engineering.MC.Drives.Enums;
```

7.28 Funktionen für Startdrive

Geber-Typ bzw. Geber-Datensatznummer offline über Hardware-Projektierung einstellen

```

DeviceItem cuDeviceItem = m_Device.DeviceItems[1];
DriveObject cuDriveObject =
cuDeviceItem.GetService<DriveObjectContainer>().DriveObjects[0];

DriveFunctionInterface cuDriveFunctionInterface =
cuDriveObject.GetService<DriveFunctionInterface>();
HardwareProjection hardwareProjection = cuDriveFunctionInterface.HardwareProjection;

// To enable the setting of an EncoderType, the value of p96 should
// be set to 0 (Application class == [0] Expert) if it is present on
// the current G drive.
DriveParameter p96 = cuDriveObject.Parameters.Find("p96");
if (p96 != null)
{
    p96.Value = 0;
}

// Setting Encoder 1 on the drive.
// In case of encoders, we have to set several enums to define an
// encoder type. These enums are: EncoderInterface, EncoderType,
// AbsoluteIncrementalFlag, and RotaryLinearFlag.

// There can be a problem, if the given enum combination is not valid.
// In that case, it has to give back a feedback.

hardwareProjection.SetEncoder(
    EncoderInterface.Terminal,
    EncoderType.HTLTTL,
    AbsoluteIncrementalFlag.Incremental,
    RotaryLinearFlag.Rotary,
    1);

// It is possible to set 2 encoders to a motor, one with encoderNumber == 1
// and the other with encoderNumber == 2

```

Das folgende Beispiel zeigt, wie Sie mit einer Onlineverbindung den Geber-Typ bzw. die Geber-Datensatznummer einstellen können.

Geber-Typ bzw. Geber-Datensatznummer online über Hardware-Projektierung einstellen

```

using Siemens.Engineering.MC.Drives
using Siemens.Engineering.MC.Drives.DFI
using Siemens.Engineering.MC.Drives.Enums;

```

Geber-Typ bzw. Geber-Datensatznummer online über Hardware-Projektierung einstellen

```
DeviceItem cuDeviceItem = m_Device.DeviceItems[1];
OnlineDriveObject cuOnlineDriveObject =
cuDeviceItem.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];
OnlineDriveFunctionInterface cuDriveFunctionInterface =
cuOnlineDriveObject.GetService<OnlineDriveFunctionInterface>();
HardwareProjection hardwareProjection = cuDriveFunctionInterface.HardwareProjection;

// To enable the setting of an Encoder, the value of p96 should
// be set to 0 (Application class == [0] Expert) if it is present
// the current G drive.
DriveParameter p96 = cuOnlineDriveObject.Parameters.Find("p96");
if (p96 != null)
{
    p96.Value = 0;
}

// Setting Encoder 1 on the drive.
// In case of encoders we have to set several enums to define an
// encoder type. These enums are: EncoderInterface, EncoderType,
// AbsoluteIncrementalFlag, and RotaryLinearFlag.

// There can be a problem, if the given enum combination is not valid.
// In that case, it has to give back a feedback.

hardwareProjection.SetEncoder(
    EncoderInterface.Terminal,
    EncoderType.HTLTTL,
    AbsoluteIncrementalFlag.Incremental,
    RotaryLinearFlag.Rotary,
    1);

// It is possible to set 2 encoders to a motor, one with encoderNumber == 1
// and the other with encoderNumber == 2
```

Das folgende Beispiel zeigt, wie Sie die aktuelle Konfiguration des Gebers aus dem Antrieb auslesen können.

Konfiguration des Gebers auslesen

```
DeviceItem cuDeviceItem = m_Device.DeviceItems[1];
DriveObject cuDriveObject =
cuDeviceItem.GetService<DriveObjectContainer>().DriveObjects[0];
DriveFunctionInterface cuDriveFunctionInterface =
cuDriveObject.GetService<DriveFunctionInterface>();

HardwareProjection encoderProjection = cuDriveFunctionInterface.HardwareProjection;

// Before setting an encoder, p96 should be 0
// (See section "Projecting EncoderConfiguration")

encoderProjection.SetEncoder(
    EncoderInterface.Terminal,
    EncoderType.HTLTTL,
    AbsoluteIncrementalFlag.Incremental,
    RotaryLinearFlag.Rotary,
    1);

EncoderConfiguration encoderConfiguration =
encoderProjection.GetCurrentEncoderConfiguration(1);
```

Das folgende Beispiel zeigt, wie Sie die aktuelle Konfiguration des Gebers offline projektieren können.

Geber offline konfigurieren

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;
```

Geber offline konfigurieren

```
// Project encoder configuration in Offline state
DeviceItem cuDeviceItem = m_Device.DeviceItems[1];

DriveObject cuDriveObject =
cuDeviceItem.GetService<DriveObjectContainer>().DriveObjects[0];
DriveFunctionInterface cuDriveFunctionInterface =
cuDriveObject.GetService<DriveFunctionInterface>();
HardwareProjection hardwareProjection = cuDriveFunctionInterface.HardwareProjection;

// Before setting an encoder type, p96 should be 0
// (See section "Projecting EncoderConfiguration")

// To Project an EncoderConfiguration, first set e.g. Encoder 1 with .SetEncoder(...).

hardwareProjection.SetEncoder(
    EncoderInterface.Terminal,
    EncoderType.HTLTTL,
    AbsoluteIncrementalFlag.Incremental,
    RotaryLinearFlag.Rotary,
    1);

// Get the current configuration of Encoder 1.
EncoderConfiguration config = hardwareProjection.GetCurrentEncoderConfiguration(1);

// Fill out Encoder 1's configuration values.
config.RequiredConfigurationEntries.ToList().ForEach(ce =>
{
    switch (ce.Name)
    {
        case "p405.0":
            ce.Value = 0;
            break;
        case "p405.1":
            ce.Value = 1;
            break;
        case "p408":
            ce.Value = 1024;
            break;
        case "p425":
            ce.Value = 2048;
            break;
        default:
            break;
    }
});

// Project the Encoder 1 configuration to the device.
bool result =
cuDriveFunctionInterface.HardwareProjection.ProjectEncoderConfiguration(config, 1);
```

Das folgende Beispiel zeigt, wie Sie die aktuelle Konfiguration des Gebers online projektieren können.

Geber online konfigurieren

```
using Siemens.Engineering.MC.Drives;  
using Siemens.Engineering.MC.Drives.DFI;  
using Siemens.Engineering.MC.Drives.Enums;
```

Geber online konfigurieren

```

// Project encoder configuration in Online state
DeviceItem cuDeviceItem = m_Device.DeviceItems[1];
DeviceItem cuDeviceItemForOnline = m_Device.DeviceItems[0];
//You need the rack for going online on G120 Device, which is .DeviceItems[0]

// GoOnline...
// To use these function in Online, you have to use the OnlineDriveFunctionInterface
OnlineDriveObject onlineDriveObject =
cuDeviceItem.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];
OnlineDriveFunctionInterface onlineDfi =
onlineDriveObject.GetService<OnlineDriveFunctionInterface>();
HardwareProjection hardwareProjection = onlineDfi.HardwareProjection;

// Before setting an encoder, p96 should be 0
// (See section "Projecting EncoderConfiguration")

// To Project an EncoderConfiguration, first set e.g. Encoder 1 with .SetEncoder(...).

hardwareProjection.SetEncoder(
    EncoderInterface.Terminal,
    EncoderType.HTLTTL,
    AbsoluteIncrementalFlag.Incremental,
    RotaryLinearFlag.Rotary,
    1);

// Get the current configuration of Encoder 1.
EncoderConfiguration config = hardwareProjection.GetCurrentEncoderConfiguration(1);

// Fill out Encoder 1's configuration values.
config.RequiredConfigurationEntries.ToList().ForEach(ce =>
{
    switch (ce.Name)
    {
        case "p405.0":
            ce.Value = 0;
            break;
        case "p405.1":
            ce.Value = 1;
            break;
        case "p408":
            ce.Value = 1024;
            break;
        case "p425":
            ce.Value = 2048;
            break;
        default:
            break;
    }
});

// Project the Encoder 1 configuration to the device.
bool result = onlineDfi.HardwareProjection.ProjectEncoderConfiguration(config, 1);

```

Das folgende Beispiel zeigt, wie Sie den Konfigurationseingang in die Konsole schreiben können.

Konfigurationseingang heraus schreiben

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
MotorConfiguration motorConfig = hardwareProjection.GetCurrentMotorConfiguration(0);

foreach (var configurationEntry in motorConfig.RequiredConfigurationEntries)
{
    Console.WriteLine(configurationEntry.Name);
}
EncoderConfiguration encConfig = hardwareProjection.GetCurrentEncoderConfiguration(1);

foreach (var configurationEntry in encConfig.RequiredConfigurationEntries)
{
    Console.WriteLine(configurationEntry.Name);
}
```

7.28.4.12 Geräte-Projektierung durchführen

Das folgende Beispiel zeigt, wie Sie für die S120- und G120-Antriebe eine Geräte-Projektierung offline durchführen können.

Geräte-Projektierung offline bei S120- bzw. G120-Antriebsgeräten durchführen

```
using Siemens.Engineering.MC.Drives
using Siemens.Engineering.MC.Drives.DFI
using Siemens.Engineering.MC.Drives.Enums;

DriveObject driveObject = ...
DriveFunctionInterface dfi = driveObject.GetService<DriveFunctionInterface>();
HardwareProjection hardwareProjection = dfi.HardwareProjection;

// dfi can be null in case of the actual driveobject does not support it.
// hardwareProjection can be null, if the actual driveObject does not support it.
// For example: On G120 drives, you have to use the CU as driveobject. On S120
// drives, you have to use the MotorModul as driveObject.
```

Das folgende Beispiel zeigt, wie Sie für G120-Antriebe eine Geräte-Projektierung online durchführen können.

Geräte-Projektierung online bei G120-Antriebsgeräten durchführen

```
using Siemens.Engineering.MC.Drives
using Siemens.Engineering.MC.Drives.DFI
using Siemens.Engineering.MC.Drives.Enums;
```

Geräte-Projektierung online bei G120-Antriebsgeräten durchführen

```
OnlineDriveObject onlineDriveObject = ...
OnlineDriveFunctionInterface onlineDfi =
onlineDriveObject.GetService<OnlineDriveFunctionInterface>();
HardwareProjection hardwareProjection = onlineDfi.HardwareProjection;

// onlineDfi can be null in case of the actual onlineDriveObject
// does not support it or if the device is offline.
// hardwareProjection can be null, if the actual onlineDriveObject
// does not support it.
// For example: On G120 drives, you have to use the CU as onlineDriveObject. On
// S120 drives, you have to use the MotorModul as onlineDriveObject.
```

7.28.4.13 Komponente für eine Antriebskomponente anlegen (nur S120)

Bei S120 haben Sie die Möglichkeit, eine Komponente unter einer Antriebskomponente anzulegen.

Für die Unterscheidung von Gebern geben Sie zusätzlich eine Typbezeichnung an. Die möglichen Typbezeichnungen und Einschränkungen bei Gebern (Encodern) sind in der Tabelle unten aufgelistet.

Das folgende Beispiel zeigt, wie Sie eine Komponente unter einer Antriebskomponente anlegen.

Motor und Geber unter einem Motormodul anlegen

```
DeviceItem subModul = sdrDevice.PlugNew(@"OrderNumber:6SL3xxx-xxxxx-xxxx",
"MotorModul", 65535);

//Plug a motor to the motor modul
subModul.Container.PlugNew(@"OrderNumber:1PH2092-4WG4x-xxxx",
"Motor_1", 65535);

//Plug an encoder to the motor modul
subModul.Container.PlugNew(@"OrderNumber:XExxxxx-xxxxx-xxxx//DRIVE-
CLIQ.202", "Encoder_1", 65535);
```

Typbezeichnungen für Geber und Einschränkungen

Beim Einfügen von Gebern über Openness gelten folgende Einschränkungen:

- Für manche Geber kann beim Einfügen über Openness nur ein unspezifisches Sensormodul angelegt. In diesen Fällen müssen Sie im TIA-Portal den konkreten Typ des Sensormoduls konfigurieren.
- Für ein Motormodul können maximal zwei Geber eingefügt werden.

7.28 Funktionen für Startdrive

Die folgende Tabelle listet die verfügbaren Typbezeichnungen für Geber auf.

DRIVE-CLiQ	Resolver	sin/cos	SSI	sin/cos+SSI	HTL/TTL	HTL/TTL +SSI	EnDat 2.1
DRIVE-CLiQ.202	Resolver.0	SIN_COS.0	SSI.0	SIN_COS_+SSI.0	HTL_TTL.0	HTL_TTL +SSI.0	EnD-at_2.1.2051
DRIVE-CLiQ.204	Resol-ver.1001	SIN_COS.2001	SSI.3081	SIN_COS_+SSI.2081	HTL_TTL.3001	HTL_TTL +SSI.3088	EnD-at_2.1.2052
DRIVE-CLiQ.212	Resol-ver.1002	SIN_COS.2002	SSI.3082	SIN_COS_+SSI.2082	HTL_TTL.3002	HTL_TTL +SSI.3090	EnD-at_2.1.2053
DRIVE-CLiQ.214	Resol-ver.1003	SIN_COS.2003	SSI.9999	SIN_COS_+SSI.2083	HTL_TTL.3003	HTL_TTL +SSI.9999	EnD-at_2.1.2054
DRIVE-CLiQ.242	Resol-ver.1004	SIN_COS.2004	-	SIN_COS_+SSI.2084	HTL_TTL.3005	-	EnD-at_2.1.2055
DRIVE-CLiQ.244	Resol-ver.9999	SIN_COS.2005	-	SIN_COS_+SSI.9999	HTL_TTL.3006	-	EnD-at_2.1.2151
DRIVE-CLiQ.9999	-	SIN_COS.2006	-	-	HTL_TTL.3007	-	EnD-at_2.1.9999
DRIVE-CLiQ.10100	-	SIN_COS.2007	-	-	HTL_TTL.3008	-	EnD-at_2.1.10100
-	-	SIN_COS.2008	-	-	HTL_TTL.3009	-	-
-	-	SIN_COS.2010	-	-	HTL_TTL.3011	-	-
-	-	SIN_COS.2012	-	-	HTL_TTL.3020	-	-
-	-	SIN_COS.2013	-	-	HTL_TTL.3109	-	-
-	-	SIN_COS.2110	-	-	HTL_TTL.9999	-	-
-	-	SIN_COS.2111	-	-	-	-	-
-	-	SIN_COS.2112	-	-	-	-	-
-	-	SIN_COS.9999	-	-	-	-	-

7.28.4.14 Motor-Typ und Motor-Konfiguration festlegen

Die folgenden Beispiele zeigen, wie Sie den Motortyp für G120-Antriebe über die Gerätekonfiguration einstellen können:

Motortyp offline über Hardware-Projektierung einstellen

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;
```

Motortyp offline über Hardware-Projektierung einstellen

```
//Offline (Only on G120 drives)
DriveFunctionInterface dfi = ...
HardwareProjection hardwareProjection = dfi.HardwareProjection;

// hardwareProjection can be null in case of the actual driveObject
// does not support activation.
// Setting the required MotorType and DriveDataSetNumber on the drive.
// It is only supported on G120 drives.

//First parameter is the MotorType, Second parameter is the DriveDataSetNumber
hardwareProjection.SetMotorType(MotorType.InductionMotor, 0);

// There can be a problem, if the selected MotorType is not available
// on the drive. In that case, it has to give back a feedback.
```

Motortyp online über Hardware-Projektierung einstellen

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;
//Online (Only on G120 drives)
OnlineDriveFunctionInterface onlineDfi = ...
HardwareProjection hardwareProjection = onlineDfi.HardwareProjection;

// hardwareProjection can be null in case of the actual
// onlineDriveObject does not support activation.
// Setting the required MotorType and DriveDataSetNumber on
// the drive. It is only supported on G120 drives.

// First parameter is the MotorType, Second parameter is the DriveDataSetNumber
hardwareProjection.SetMotorType(MotorType.InductionMotor, 0);

// There can be a problem, if the selected MotorType is not available on
// the drive. In that case, it has to give back a feedback.
```

Die folgenden Beispiele zeigen, wie Sie den Motortyp für G120-Antriebe aus dem Antrieb auslesen können:

Motorkonfiguration offline ermitteln

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;

// Offline
// WARNING: You have to set the MotorType on G drives before
// you would like to get the current configuration, otherwise
// you will get an exception/feedback, which informs you that
// there is no motor set.
// On S120 drives, you don't have to do that, but you have to
// plug a motor to motor modul.( later on)

HardwareProjection hardwareProjection = dfi.HardwareProjection;
// Get the current motor configuration
// It needs a datasetnumber, which is currently only supported on G120 drives.
MotorConfiguration motorConfiguration = hardwareProjection.GetCurrentMotorConfiguration(0);
```

Motorkonfiguration online ermitteln

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;

// Online
// WARNING: You have to set the MotorType on G drives before
// you would like to get the current configuration, otherwise
// you will get an exception/feedback, which informs you that
// there is no motor set.
// On S120 drives, it is not possible??

HardwareProjection hardwareProjection = onlineDfi.HardwareProjection;
// Get the current motor configuration
// It needs a datasetnumber, which is currently only supported on G120 drives.
MotorConfiguration motorConfiguration = hardwareProjection.GetCurrentMotorConfiguration(0);
```

Das folgende Beispiel zeigt, wie Sie eine Motorkonfiguration für G120-Antriebe durchführen können:

Motorkonfiguration projektieren

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;
```

Motorkonfiguration projektieren

```

DeviceItem cuDeviceItem = m_Device.DeviceItems[1];
DriveObject cuDriveObject =
cuDeviceItem.GetService<DriveObjectContainer>().DriveObjects[0];
DriveFunctionInterface cuDriveFunctionInterface =
cuDriveObject.GetService<DriveFunctionInterface>();

HardwareProjection hardwareProjection = cuDriveFunctionInterface.HardwareProjection;
hardwareProjection.SetMotorType(MotorType.InductionMotor, 0);

MotorConfiguration config = hardwareProjection.GetCurrentMotorConfiguration(0);

config.RequiredConfigurationEntries.ToList().ForEach(ce =>
{
    switch (ce.Number)
    {
        case 305:
            ce.Value = 20;
            break;
        case 307:
            ce.Value = 30;
            break;
        case 311:
            ce.Value = 200;
            break;
        case 304:
            ce.Value = 450;
            break;
        case 310:
            ce.Value = 50;
            break;
        case 335:
            ce.Value = 1;
            break;
        default:
            break;
    }
});
bool result =
cuDriveFunctionInterface.HardwareProjection.ProjectMotorConfiguration(config, 0);

```

7.28.4.15 Parameter lesen und schreiben

Das folgende Beispiel zeigt, wie Sie Werte von Antriebsparametern lesen und schreiben. Für den Zugriff benötigen Sie ein Antriebsobjekt.

Zugriff auf Parameter

```
using Siemens.Engineering.MC.Drives;
```

Zugriff auf Parameter

```

//Access a parameter via its name
DriveParameter parameter = driveObject.Parameters.Find("p5391[0]");

//Example of reading parameter attributes
if (parameter != null)
{
    Console.WriteLine("The Name of the parameter is : " + parameter.Name);
    Console.WriteLine("The value of the parameter is : " +
parameter.Value.ToString());
    Console.WriteLine("The minValue of the parameter is : " +
parameter.MinValue);
    Console.WriteLine("The MaxValue of the parameter is : " +
parameter.MaxValue);
    Console.WriteLine("The Unit of the parameter is : " + parameter.Unit);

    //Example for write:
    parameter.Value = 60;
}

// Access a parameter via Number and ArrayIndex// Note that
// - arrayless parameters (e.g. p96 on G120) are indexed by -1
// - parameters that consist of only a bit array do not count as
// array parameters, they are indexed by -1 as well and the
// further bit values can be accessed by the 'Bits' property
// of the given parameter (e.g. r2139 on G120). For further
// information about accessing bit parameters see the next
// section of this code snippet

DriveParameter r947_6 = driveObject.Parameters.Find(947, 6); // returns
r947[6]if (r947_6 != null)
{
    Console.WriteLine("The Name of the parameter is : " + r947_6.Name);
    Console.WriteLine("The value of the parameter is : " +
r947_6.Value.ToString());
}

//Accessing bit values
DriveParameter p2720 = driveObject.Parameters.Find(2720, 0);if (p2720 != null)
{
    // Note that in general, pXXX.Bits[YYY] is not necessarily equivalent to
    pXXX.YYY.
    // pXXX.Bits is an array of available bit values in ascending order by
    their names
    // and not an array of bit values indexed by their names.
    // For example: let the available bit values be [pXXX.0, pXXX.2, pXXX.3],
    then
    // pXXX.Bits[2] == pXXX.3, not pXXX.2

    DriveParameter p2720Bit1 = p2720.Bits[1]; // returns p2720[0].1
    if (p2720Bit1 != null)
    {
        Console.WriteLine("The name of the parameter is : " + p2720Bit1.Name);
        Console.WriteLine("The value of the second bit of the parameter is : " +
p2720Bit1.Value.ToString());
    }
}

```

```

Zugriff auf Parameter
}

//Get the enum values of a parameter
DriveParameter r47 = driveObject.Parameters.Find("r47");
foreach (var enumItem in r47.EnumValueList)
{
    Console.WriteLine("Enum value: " + enumItem.Key.ToString() + " = " +
enumItem.Value);
}

```

7.28.4.16 Parameter online lesen und schreiben

Das folgende Beispiel zeigt, wie Sie eine Liste mit den verfügbaren Online-Parametern erhalten. Für den Zugriff benötigen Sie ein Online-Antriebsobjekt.

Der Lese- und Schreibzugriff auf einzelne Online-Parameter aus der Parameterliste ist identisch wie im Beispiel Parameter lesen und schreiben (Seite 745).

Zugriff auf Online-Parameter

```

using Siemens.Engineering.MC.Drives;
DeviceItem item = ... //device item of the CU (e.g. :
project.Devices[0].Items[0].Items[0])
OnlineDriveObject onlineDriveObject =
item.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];
if (onlineDriveObject != null)
{
    var parameters = onlineDriveObject.Parameters;
}

```

7.28.4.17 Parametrierung speichern

Das folgende Beispiel zeigt, wie Sie die Parametrierung aus den S120, S210 oder G120-Antrieben ermitteln können.

Parametrierung online aus den Antrieben ermitteln

```

using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;

```

Parametrierung online aus den Antrieben ermitteln

```
OnlineDriveObject onlineDriveObject =
item.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];
OnlineDriveFunctionInterface onlineDfi =
onlineDriveObject.GetService<OnlineDriveFunctionInterface>();
DriveDomainFunctions driveDomainFunctions = onlineDfi.DriveDomainFunctions;

// onlineDfi can be null in case of the actual onlineDriveObject
// does not support it or if the device is offline.

// driveDomainFunctions can be null, if the actual onlineDriveObject
// does not support it.

// For example: On G120 and S210 drives, you have to use the CU to get the
onlineDriveObject.
// Please, pay attention at S120 drives. Here you can use any module (CU, MotorModul,
lineModul) as onlineDriveObject but preferable the CU, because it will run on CU!
```

Das folgende Beispiel zeigt, wie Sie die Parametrierung bei S120, S210 oder G120-Antrieben von RAM nach ROM kopieren und damit netzausfallsicher speichern können.

Sicherung RAM nach ROM durchführen

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;
//In case of G120 device.
DeviceItem cuDeviceItem = m_Device.DeviceItems[1];
OnlineDriveObject cuDriveObject =
cuDeviceItem.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];

//In case of S120 and S210 device you should generally get the CU driveobject.
DeviceItem cuDeviceItem = m_Device.DeviceItems[0];OnlineDriveObject cuDriveObject =
cuDeviceItem.GetService<OnlineDriveObjectContainer>().OnlineDriveObjects[0];

// To use the function, you have to use the OnlineDriveFunctionInterface.
OnlineDriveFunctionInterface cuDriveFunctionInterface =
cuDriveObject.GetService<OnlineDriveFunctionInterface>();
DriveDomainFunctions driveDomainFunctions = cuDriveFunctionInterface.DriveDomainFunctions;

// This function will perform a RAM to ROM copy on all device.
// It is important that the CU should be in online state before you call this function
anyway the program will give an exception.
// This call may take a few moments before be finished.
bool result = DriveDomainFunctions.PerformRAMtoROMCopyAllDriveObject();
```

7.28.4.18 Safety Integrated-Telegramme verwenden

Das folgende Beispiel zeigt, wie Sie Safety Integrated-Telegramme (z. B. 30) in Openness verwenden können.

Safety Integrated-Telegramme verwenden

```
using Siemens.Engineering.MC.Drives;
```

Safety Integrated-Telegramme verwenden

```
TelegramComposition telegrams = drvObj.Telegrams;

//Add safety telegram
const int tgrmNumber = 30;
drvObj.Telegrams.InsertSafetyTelegram(tgrmNumber);

//Find safety telegram
Telegram safetyTgrm = drvObj.Telegrams.Find(TelegramType.SafetyTelegram);

// Get and set safety telegram attributes
uint watchDogTime = (uint)safetyTgrm.GetAttribute("Failsafe_FMonitoringtime");

safetyTgrm.SetAttribute("Failsafe_FMonitoringtime", 300);

const int newSafetyTelegramNumber= 900;
if (safetyTgrm.CanChangeTelegram(newSafetyTelegramNumber))
{
    safetyTgrm.TelegramNumber = newSafetyTelegramNumber;
}

//Remove Safety telegram
drvObj.Telegrams.EraseTelegram(TelegramType.SafetyTelegram);
```

7.28.4.19 Telegramme einfügen und erweitern

Das folgende Beispiel zeigt, wie Sie eine Erweiterung einfügen und die Größe eines Standardtelegramms verändern. Für den Zugriff benötigen Sie ein Antriebsobjekt.

Erweiterung einfügen und die Größe eines Standardtelegramms ändern

```
using Siemens.Engineering.MC.Drives;
```

Erweiterung einfügen und die Größe eines Standardtelegrams ändern

```

TelegramComposition telegrams = drvObj.Telegrams;
Telegram telegram = telegrams.Find(TelegramType.MainTelegram);

Console.WriteLine("The Cu has the telegram: " + telegram.TelegramNumber);
Console.WriteLine("The Setpoint channel-specific size of the telegram is: " +
+ telegram.GetOutputSize());

foreach (var address in telegram.Addresses)
{
    if (address.IoType == AddressIoType.Output)
    {
        Console.WriteLine("The Setpoint channel-specific starting address of
the connected PLC is: " + address.StartAddress);
    }
    else if (address.IoType == AddressIoType.Input)
    {
        Console.WriteLine("The Actual value channel-specific starting address
of the connected PLC is: " + address.StartAddress);
    }
}

// Add an additional Telegram
if (drvObj.Telegrams.CanInsertAdditionalTelegram(2, 4))
{
    drvObj.Telegrams.InsertAdditionalTelegram(2, 4);
}

// Add a 3 word extension to the main telegram
Telegram mainTelegram == drvObj.Telegrams.Find(TelegramType.MainTelegram);
Int32 newSize = mainTelegram.GetSize(AddressIoType.Input) + 3;
if (mainTelegram.CanChangeSize(AddressIoType.Input, newSize, true))
{
    mainTelegram.ChangeSize(AddressIoType.Input, newSize, true)
}

```

7.28.4.20 Torque-Telegramme verwenden

Das folgende Beispiel zeigt, wie Sie Torque-Telegramme (z. B. 750) in Openness verwenden können.

Torque-Telegramme verwenden

```

using Siemens.Engineering.MC.Drives;
const int torqueTelegramNumber = 750;
TelegramComposition telegrams = drvObj.Telegrams;

// Add a new torque telegram
if (drvObj.Telegrams.CanInsertTorqueTelegram(torqueTelegramNumber))
{
    drvObj.Telegrams.InsertTorqueTelegram(torqueTelegramNumber);
}

// Find torque telegram
Telegram torqueTelegram = drvObj.Telegrams.Find(TelegramType.TorqueTelegram);

```

7.28.4.21 Werkseinstellungen wiederherstellen

Das folgende Beispiel zeigt, wie Sie für S120, S210 oder G120-Antriebe die Werkseinstellung wiederherstellen können.

Werkseinstellungen wiederherstellen

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DFI;
using Siemens.Engineering.MC.Drives.Enums;

OnlineDriveObject onlineDriveObject = ...
OnlineDriveFunctionInterface onlineDfi =
onlineDriveObject.GetService<OnlineDriveFunctionInterface>();
DriveDomainFunctions driveDomainFunctions = onlineDfi.DriveDomainFunctions;

// onlineDfi can be null in case of the actual onlineDriveObject
// does not support it or if the device is offline.
// saveParametrization can be null, if the actual onlineDriveObject
// does not support it.
// For example: On G120 and S210 drives, you have to use the CU to get the
onlineDriveObject.
// S120 drives, you can use any modul (CU, MotorModul, lineModul) as onlineDriveObject but
preferable the CU.
```

7.29 Funktionen für DCC

7.29.1 Einführung

Über TIA Portal-Openness automatisieren Sie das Engineering und steuern das TIA Portal über ein von Ihnen selbst erstelltes Programm.

Für dieses selbst erstellte Programm finden Sie in dieser Hilfe viele Informationen und Beispielcodes. Auch für die TIA Portal-Anwendung "DCC" können Sie eigene Programme zusammenstellen und anwenden.

Bevor Sie für DCC aus den nachfolgend gelisteten Beispielcodes ein eigenes Programm zusammenstellen, beachten Sie bitte die übergreifenden Informationen zu Openness, die Sie unter folgenden Stichworten in dieser Hilfe finden:

- Voraussetzungen für TIA Portal Openness
- TIA Portal Openness installieren
- Auf das TIA Portal zugreifen
- TIA Portal Openness-Objektmodell
- Programmierschritte

7.29.2 DCC Openness

Einleitung

Mit TIA Portal Openness V16 programmieren Sie Anwendungen, die das Engineering im TIA Portal automatisieren.

Funktionen von Openness in Verbindung mit SINAMICS DCC

Folgende Funktionen sind in der Version V16 implementiert:

- Pläne importieren (Seite 758)
- Einen Plan exportieren (Seite 758) oder alle Pläne (Ordner "Pläne") exportieren (Seite 759)
- DCB Extension-Bibliotheken in die Projektbibliothek importieren (Seite 761)
- Pläne anhand des Namens finden (Seite 759)
- Pläne aus dem Ordner "Pläne" löschen (Seite 760)
- Ablaufreihenfolge der Blöcke innerhalb eines Plans optimieren (Seite 760)

Hinweis

Subpläne

Pläne, die Subpläne enthalten, können mit Openness importiert werden. Auf die Subpläne kann aber mit Openness nicht zugegriffen werden.

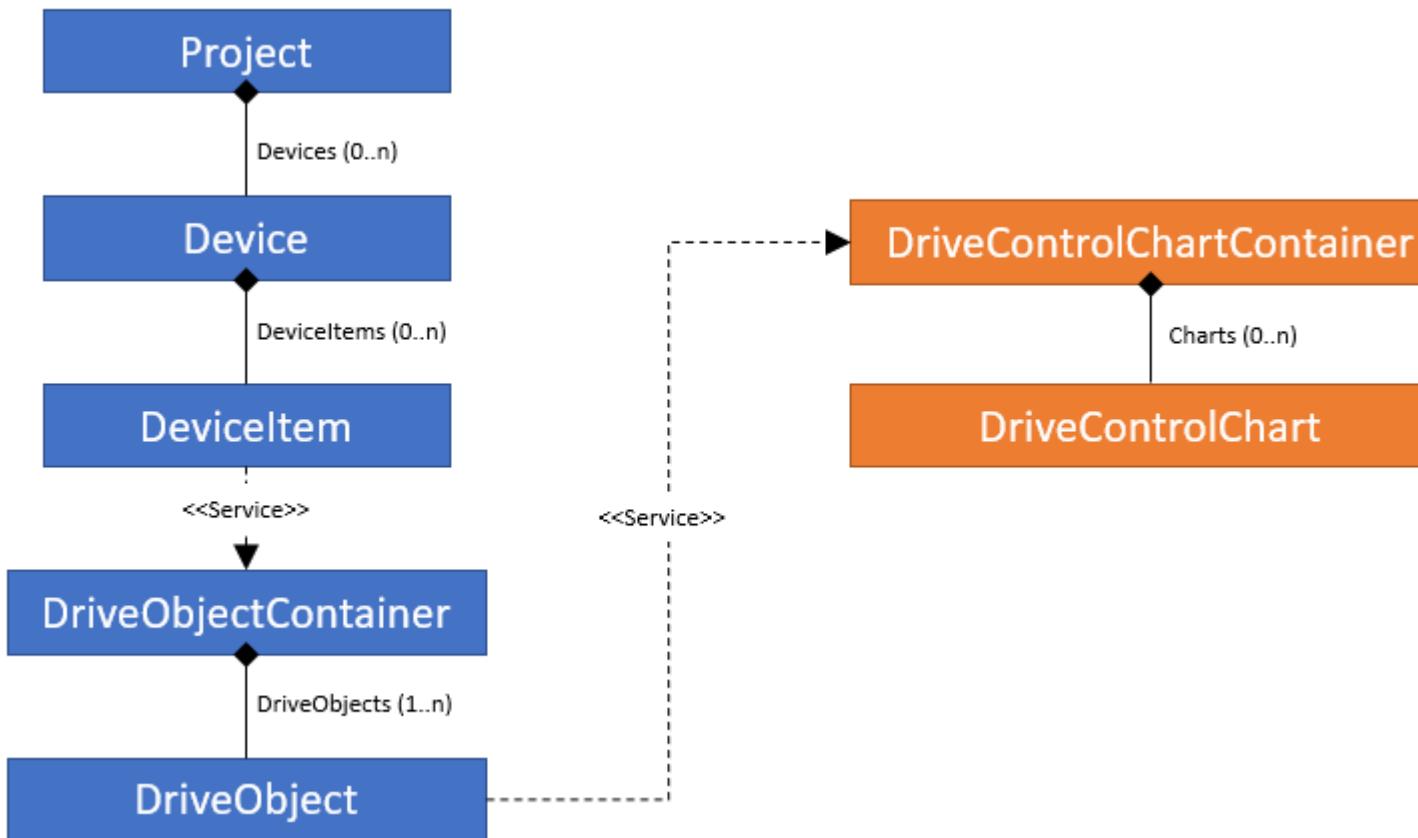
Weitere Informationen

Siehe TIA Informationssystem „Openness: Projekterstellung automatisieren“.

7.29.3 Objektmodell DCC Openness

Übersicht

Das folgende Diagramm beschreibt das DCC Openness-Objektmodell:



7.29.4 Referenzen

7.29.4.1 DriveControlChartContainer

DriveParameter

DriveControlChartContainer ist ein Dienst, der den DCC-Plan-Container unter einem DriveObject repräsentiert, sodass er aus dem entsprechenden DriveObject abgerufen werden kann. Wenn das DriveObject nicht von einem unterstützten Gerät stammt, ist der Dienst nicht verfügbar.

Die folgende Tabelle beschreibt die **Navigatoren** von DriveControlChartContainer:

Name	Datentyp	Schreibbar	Beschreibung
Charts	DriveControlChartComposition	Nur lesen	Liest das Plan-Layout im Plan-Container aus.

7.29.4.2 DriveControlChartComposition

DriveControlChartComposition features

DriveControlChartComposition zeigt eine Liste der verfügbaren Pläne.

Die folgende Tabelle beschreibt die **Methoden** von DriveControlChartComposition:

Name	Typ Rückgabewert	Parameter	Beschreibung	Exceptions
Import	DccImportResultData (Seite 755)	string path, DccImportOptions (Seite 755) import- Options	Beschreibung Methode: Importiert Pläne aus einer DCC-Exportdatei. Beschreibung Parameter: path: Vollständiger Pfad der zu importierenden Datei. importOptions: Beim Import verwendete Optionen, siehe DccImportOptions (Seite 755).	DccImportException (Seite 764)
Export	-	string path	Beschreibung Methode: Exportiert alle Pläne in eine DCC-Exportdatei. Beschreibung Parameter: path: Vollständiger Pfad der zu exportierenden Datei.	DccExportException (Seite 764)
GetChartSequence	DriveControlChart (Seite 756)	-	Liest alle Pläne im Container in der Ablaufreihenfolge aus.	-
Find	DriveControlChart (Seite 756)	string name	Identifiziert einen Plan anhand seines Namens im Plan-Container.	-

Siehe auch

[DriveControlChartComposition Exceptions \(Seite 764\)](#)

[DccImportResultData \(Seite 755\)](#)

[DriveControlChart \(Seite 756\)](#)

[DccImportOptions \(Seite 755\)](#)

7.29.4.3 DcclImportOptions

DcclImportOptions

Für den Import von Plänen sind diese Einstellungen möglich:

- DcclImportOptions.None:
Der Import ist erfolgreich abgeschlossen, wenn es keinen Namenskonflikt gibt. Ansonsten wird die Ausnahme DcclImportChartWithSameNameAlreadyAvailableException ausgegeben.
- DcclImportOptions.RenameOnConflict:
Wenn es einen Namenskonflikt bei den Plänen gibt, erhält der neu importierte Plan einen automatisch generierten Namen durch das CFC. Falls z. B. der Plan mit dem Namen CFC_1 während des Imports bereits vorhanden ist, erhält der neu importierte Plan den Namen CFC_2.

Siehe auch

[DriveControlChartComposition \(Seite 754\)](#)

7.29.4.4 DcclImportResultData

DriveParameter

DcclImportResultData enthält Informationen zum Ergebnis eines Imports von Plänen.

Die folgende Tabelle beschreibt die **Attribute** von DcclImportResultData:

Attributname	Datentyp	Schreibbar	Beschreibung
RemappedParameterNumbers	IDictionary<UInt16, UInt16>	Nur lesen	Während eines Imports kann es passieren, dass ein zu importierender DCC-Parameter bereits vorhanden ist. In diesem Fall erhält der zu importierende Parameter eine neue Nummer. Dieses Dictionary enthält dann alle neu zugeordneten Parameter. Der Schlüsselwert ist die vorherige Parameternummer, als sie exportiert wurde und der Wert ist der neu erstellte Parameter.

Siehe auch

[DriveControlChartComposition \(Seite 754\)](#)

7.29.4.5 DriveControlChart

DriveControlChart Features

DriveControlChart entspricht einem Plan innerhalb des Plan-Containers (Sub-Pläne werden von Openness nicht unterstützt).

Die folgende Tabelle beschreibt die **Attribute** von DriveControlChart:

Attributname	Datentyp	Schreibbar	Beschreibung
Name	string	Nur lesen	Name des Plans.

Die folgende Tabelle beschreibt die **Methoden** von DriveControlChart:

Name	Typ Rückgabewert	Parameter	Beschreibung	Exceptions
Export	-	string path	Beschreibung Methode: Exportiert ausschließlich diesen Plan. Beschreibung Parameter: path: Vollständiger Pfad der zu exportierenden Datei.	DccExportException (Seite 764)
Delete	-	-	Beschreibung Methode: Löscht den Plan.	-
OptimizeRunSequence	-	-	Beschreibung Methode: Optimiert die Ablaufreihenfolge des Plans. Der Optimierungsmechanismus wird von CFC bereitgestellt. Hinweis: Die Optimierung der Ablaufreihenfolge ist nur möglich, wenn eine DCC-Lizenz vorhanden ist.	DccLicenseUnavailableException (Seite 764)

Siehe auch

[DriveControlChart Exceptions \(Seite 764\)](#)

[DriveControlChartComposition \(Seite 754\)](#)

[Pläne löschen \(Seite 760\)](#)

7.29.4.6 Import DCB extension library

ImportDCBextensionlibrary

Die folgende Tabelle beschreibt die **Methoden** von ImportDCBextensionlibrary:

Name	Rückgabe-wert	Para-meter	Beschreibung
ImportDcbLibrary	-	string path	Argumente Methode: path: Der vollständige Pfad der Zip-Datei der DCB Extension-Bibliothek. Beschreibung Methode: Importiert eine DCB Extension-Bibliothek in die Projektbibliothek.

7.29.5 Codebeispiele

7.29.5.1 Allgemein

Die folgenden Codebeispiele beschreiben die grundsätzliche Vorgehensweise für verschiedene Anwendungsfälle. Der Code ist nicht zwingend vollständig und kompilierbar.

7.29.5.2 Zugriff auf DriveControlChartContainer über DriveObject

Das folgende Beispiel zeigt, wie Sie über ein Antriebsobjekt auf DriveControlChartContainer zugreifen.

Zugriff auf DriveControlChartContainer

```
using Siemens.Engineering.MC.Drives;
using Siemens.Engineering.MC.Drives.DCC;
Project project = portal.Projects.Open("..."); //Destination folder to
open the project
DeviceItem item = project.Devices[0].Items[0];
DriveObject
driveObject = item.GetService<DriveObjectContainer>().DriveObjects[0];
DriveControlChartContainer
chartContainer = driveObject.GetService<DriveControlChartContainer>();
//chartContainer can be null in case the DriveObject does not support DCC.
```

7.29.5.3 Pläne abrufen

Das folgende Beispiel zeigt, wie Sie Pläne abrufen können.

Pläne abrufen

```
DriveControlChartComposition charts = chartContainer.Charts;
```

7.29.5.4 Auf Pläne zugreifen

Das folgende Beispiel zeigt, wie Sie auf Pläne zugreifen können.

Auf Pläne zugreifen

```
DriveControlChartComposition charts = ...  
DriveControlChart firstChart = charts[0];  
  
// or looping...  
foreach(DriveControlChart chart in charts)  
{  
    ...  
}
```

7.29.5.5 Pläne importieren

Das folgende Beispiel zeigt, wie Sie Pläne importieren.

Pläne importieren

```
try  
{  
    DriveControlChartComposition charts = ...  
    DccImportResultData result = charts.Import(@"c:\Charts.dcc",  
    DccImportOptions.None);  
}  
catch (DccImportException exc)  
{  
}
```

Siehe auch

DCC Openness (Seite 752)

7.29.5.6 Pläne exportieren

Das folgende Beispiel zeigt, wie Sie Pläne exportieren können.

Pläne exportieren

```
try  
{  
    DriveControlChart chart;  
    ...  
    chart.Export(@"c:\CFC_1.dcc");  
}  
catch (DccExportException exc)  
{  
}
```

Siehe auch

[DCC Openness \(Seite 752\)](#)

7.29.5.7 Alle Pläne exportieren

Das folgende Beispiel zeigt, wie Sie alle Pläne exportieren können.

Alle Pläne exportieren

```
try
{
    charts.Export(@"c:\Charts.dcc");
}
catch (DccExportException exc)
{
}
```

Siehe auch

[DCC Openness \(Seite 752\)](#)

7.29.5.8 Pläne in der Ablaufreihenfolge abrufen

Das folgende Beispiel zeigt, wie Sie Pläne in der Ablaufreihenfolge abrufen können.

Pläne abrufen

```
DriveControlChartComposition charts = ...
IList<DriveControlChart> chartSequence = charts.GetChartSequence();
```

7.29.5.9 Pläne anhand des Namens finden

Das folgende Beispiel zeigt, wie Sie Pläne anhand des Namens suchen können.

Pläne finden

```
DriveControlChartComposition charts = ...
DriveControlChart chart = charts.Find("CFC_1");
```

Siehe auch

[DCC Openness \(Seite 752\)](#)

7.29.5.10 Importbericht anzeigen

Das folgende Beispiel zeigt, wie Sie den Bericht zu einem Importvorgang anzeigen können.

Importbericht anzeigen

```
DccImportResultData importResultData = charts.Import(@"d:\Charts.dcc",
DccImportOptions.None);
IDictionary<ushort, ushort> remappedParameters = importResultData.RemappedP
arameterNumbers;
foreach (KeyValuePair<ushort, ushort> remappedParameter in remappedParamete
rs)
{
    System.Console.WriteLine($"ParameterNumber <{remappedParameter.Key}> has
been changed to <{remappedParameter.Value}> during import.");
}
```

7.29.5.11 Pläne löschen

Das folgende Beispiel zeigt, wie Sie Pläne löschen können.

Pläne löschen

```
try
{
    DriveControlChart chart;
    ...
    chart.Delete();
}
```

Siehe auch

[DCC Openess \(Seite 752\)](#)

[DriveControlChart \(Seite 756\)](#)

7.29.5.12 Ablaufreihenfolge optimieren

Das folgende Beispiel zeigt, wie Sie die Ablaufreihenfolge der Pläne optimieren können.

Ablaufreihenfolge der Pläne optimieren

Ablaufreihenfolge der Pläne optimieren

```
try
{
    DriveControlChart chart;
    ...
    chart.OptimizeRunSequence();
}
catch (DccLicenseUnavailableException exc)
{
}
```

Siehe auch

[DCC Openness \(Seite 752\)](#)

7.29.5.13 DCB Extension-Bibliothek importieren

Das folgende Beispiel zeigt, wie Sie eine DCB Extension-Bibliothek importieren können.

DCB Extension-Bibliothek importieren

```
try
{
    Project myProject;
    ...

myProject.ProjectLibrary.ImportDcbLibrary(@"c:\GMcv5_1_sinamics5_1_(5.1.15
).zip");
}
catch (DccImportException exc)
{}
```

Siehe auch

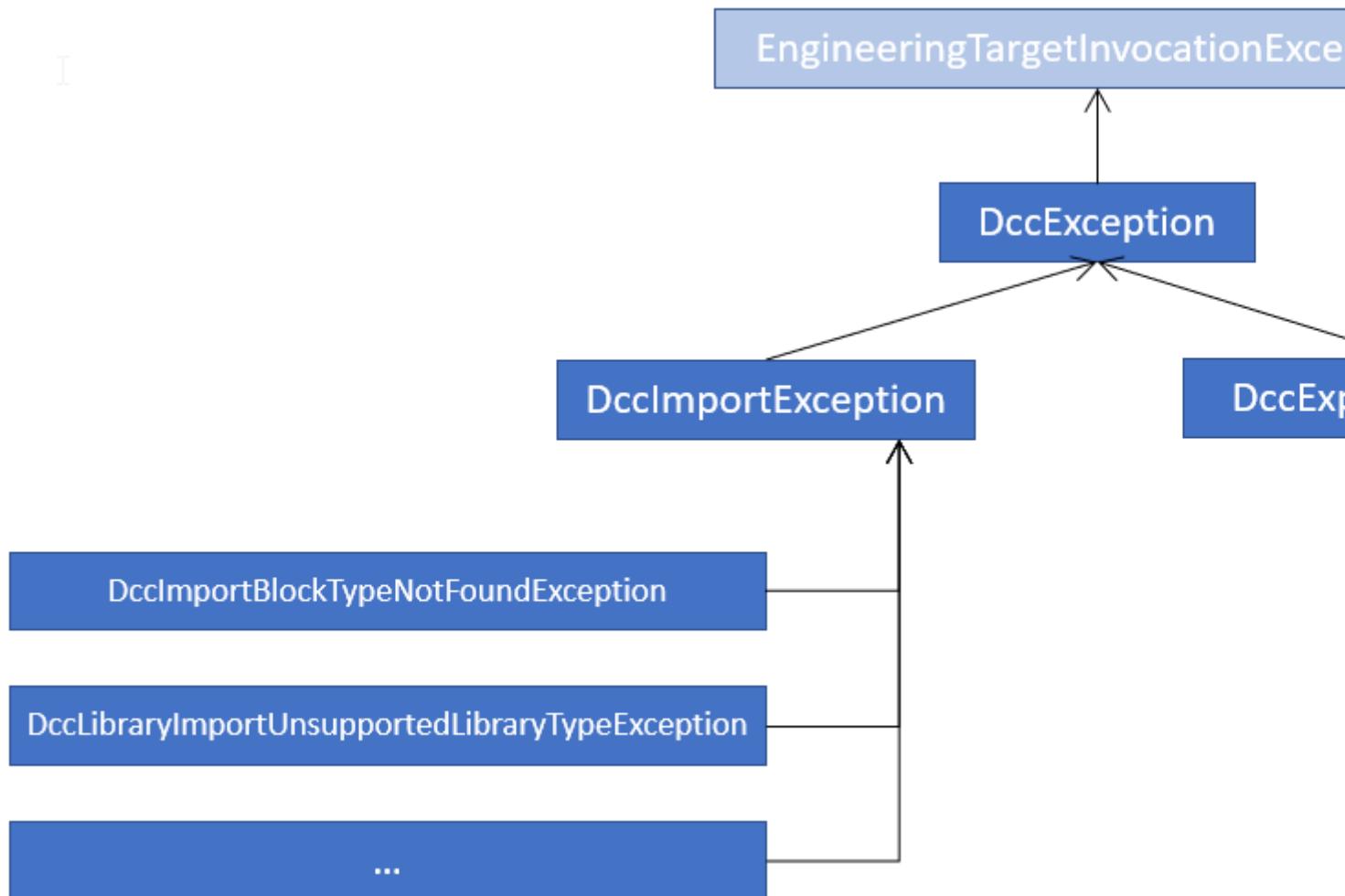
[DCC Openness \(Seite 752\)](#)

7.29.6 Ausnahmen DCC Openess

7.29.6.1 Umgang mit Exceptions

Umgang mit Exceptions

Für Anwendungen, die aufgrund von Benutzerfehlern fehlschlagen könnten, löst DCC Openess eine DCC-spezifische Ausnahme aus, die von `EngineeringTargetInvocationException` abgeleitet wird. Die Ausnahmen sind hierarchisch aufgebaut, wodurch alle spezifischen Ausnahmen von der übergeordneten Ausnahme `DccException` abgeleitet werden:



Wenn Sie nicht im Detail unterschiedliche Fehlerursachen untersuchen oder auswerten möchten, ist der einfachste Weg, die allgemeine DccException abzufangen:

```
try
{
    Project myProject;
    ...

    myProject.ProjectLibrary.ImportDcbLibrary(@"c:\GMCV5_1_sinamics5_1_(5.1.15
).zip");
    ...
    DriveControlChartContainer chartContainer = driveObject.GetService<DriveC
ontrolChartContainer>();
    chartContainer.Charts.Import(@"d:\Charts.dcc", DccImportOptions.None);
}
catch (DccException exc)
{
}
```

Fangen Sie alle relevanten Ausnahmen ab, wenn Sie je nach Fehlerart unterschiedlich reagieren möchten:

```
try
{
    DriveControlChartContainer
chartContainer = driveObject.GetService<DriveControlChartContainer>();
    chartContainer.Charts.Import(@"d:\Charts.dcc", DccImportOptions.None);
}
catch (DccImportLibraryIsMissingException missingLibExc)
{
}
catch (DccImportChartWithSameNameAlreadyAvailableException sameNameExc)
{
}
catch (DccImportException exc)
{
}
catch (DccException exc)
{
}
```

7.29.6.2 DriveControlChart Exceptions

Ausnahmen

Ausnahmen, die bei einem Export ausgegeben werden können:

- **DccExportException:**
Wird bei jeder Art von Exportfehler ausgegeben.

Ausnahmen, die beim Ausführen der Optimierungsfunktion für die Ablaufreihenfolge von Plänen ausgegeben werden können:

- **DccLicenseUnavailableException:**
Wird ausgegeben, wenn keine DCC-Lizenz vorhanden ist. In diesem Fall steht die Optimierungsfunktion für die Ablaufreihenfolge nicht zur Verfügung.

Siehe auch

[DriveControlChart \(Seite 756\)](#)

7.29.6.3 DriveControlChartComposition Exceptions

Ausnahmen

Ausnahmen, die beim Ausführen von Import ausgegeben werden:

- **DcclImportException:**
Allgemeine Importausnahme, die DCC im Fall eines Importfehlers ausgibt.
- **DcclImportBlockCreationException:**
Wird ausgegeben, wenn ein Block während eines Imports nicht erstellt werden kann.
- **DcclImportBlockTypeNotFoundException:**
Wird ausgegeben, wenn ein referenzierter DCB-Blocktyp weder in einer Standard-DCBLib oder einer Extension-Bibliothek gefunden werden kann.
- **DcclImportChartCreationException:**
- Wird ausgegeben, wenn ein Plan während eines Imports nicht erstellt werden kann.
- **DcclImportChartWithSameNameAlreadyAvailableException:**
Wird ausgegeben, wenn ein Plan mit demselben Namen bereits im Plan-Container vorhanden ist und DcclImportOptions.None verwendet wird.
- **DcclImportLibraryIsMissingException:**
Wird ausgegeben, wenn eine referenzierte DCB Extension-Bibliothek nicht im Projekt vorhanden ist.

- **DccImportFileAlreadyInUseException:**
Wird ausgegeben, wenn eine zu importierende Datei bereits durch einen anderen Prozess verwendet wird.
- **DccImportDcbTypeDifferentVersionAlreadyUsedException:**
Wird ausgegeben, wenn ein DCB-Typ in einer anderen Version der gleichen DCB Extension-Bibliothek bereits im Gerät verwendet wird.

Siehe auch

[DriveControlChartComposition \(Seite 754\)](#)

7.29.6.4 ImportDcbLibrary Exceptions

Ausnahmen

Ausnahmen, die von ImportDcbLibrary ausgegeben werden:

- **DccImportException:**
Wird bei jeder Art von Importfehler ausgegeben.
- **DccLibraryImportAlreadyAvailableException:**
Wird ausgegeben, wenn bereits eine DCB Extension-Bibliothek mit der gleichen Version im Projekt vorhanden ist.
- **DccLibraryImportCorruptedStudioLibraryException:**
Wird ausgegeben, wenn die Zip-Datei der DCB Extension-Bibliothek kompromittiert ist und nicht importiert werden kann.
- **DccLibraryImportIntegrityBrokenException:**
Wird ausgegeben, wenn die Integrität der Zip-Datei der DCB Extension-Bibliothek kompromittiert ist.
- **DccLibraryImportOverallPinLimitExceededException:**
Wird ausgegeben, wenn die DCB Extension-Bibliothek einen DCB-Typ enthält, dessen Pinanzahl die maximal Pinanzahl überschreitet.
- **DccLibraryImportStandardLibraryAlreadyAvailableException:**
Wird ausgegeben, wenn die zu importierende ausgewählte DCB Extension-Bibliothek die Standard-DCBLib ist, die bereits vorhanden ist.
- **DccLibraryImportUnsupportedLibraryTypeException:**
Wird ausgegeben, wenn die ausgewählte DCB Extension-Bibliothek nicht durch DCC und SINAMICS-Antriebe unterstützt wird.

7.30 Ausnahmen

7.30.1 Umgang mit Exceptions

Ausnahmen beim Zugriff auf das TIA Portal über TIA Portal Openness APIs

Bei der Ausführung einer TIA Portal Openness-Anwendung über die TIA Portal Openness API werden alle auftretenden Fehler als Ausnahmen gemeldet. Diese Ausnahmen enthalten Informationen, die Ihnen helfen, die aufgetretenen Fehler zu beheben.

Dabei wird zwischen zwei Arten von Ausnahmen unterschieden:

- Recoverable (Siemens.Engineering.EngineeringException)
Sie können mit dieser Ausnahme weiterhin ohne Unterbrechung auf das TIA Portal zugreifen. Alternativ können Sie die Verbindung zum TIA Portal abbrechen.
Die EngineeringExceptions beinhalten die folgenden Typen:
 - Sicherheitsbezogene Ausnahmen (EngineeringSecurityException), beispielsweise bei fehlenden Zugriffsrechten.
 - Ausnahmen beim Zugriff auf Objekte (EngineeringObjectDisposedException), beispielsweise beim Zugriff auf nicht mehr vorhandene Objekte.
 - Ausnahmen beim Zugriff auf Attribute (EngineeringNotSupportedException), beispielsweise beim Zugriff auf nicht mehr vorhandene Attribute.
 - Allgemeine Ausnahmen beim Aufruf (EngineeringTargetInvocationException), beispielsweise bei einem Fehler trotz gültigen Aufrufs der TIA Portal Openness API.
 - Ausnahmen beim Aufruf (EngineeringRuntimeException), beispielsweise bei einer ungültigen Belegung.
 - Ausnahmen, wenn es nicht genügend Ressourcen in der zugeordneten TIA Portal-Instanz gibt (EngineeringOutOfMemoryException)
 - Ausnahmen bei beendeten Aufrufen (EngineeringUserAbortException), beispielsweise beim Abbruch des Importvorgangs durch den Benutzer.
 - Ausnahmen beim API-Aufruf durch einen vom Kunden bereitgestellten Delegate (EngineeringDelegateInvocationException). Diese Ausnahme wird von der Ausnahme EngineeringTargetInvocationException abgeleitet.

Die EngineeringExceptions haben die folgenden Attribute:

- ExceptionMessageData messageData: Enthält den Grund, weshalb die Ausnahme ausgelöst wurde.
- ExceptionMessageData detailMessageData: Enthält zusätzliche Informationen über den Grund. Das Ergebnis wird als <IList> zurückgegeben.
- String message: Gibt das Ergebnis von MessageData und DetailMessageData zurück.

ExceptionMessageData gibt die folgenden Informationen zurück:

- String Text: Enthält den Grund, weshalb die Ausnahme ausgelöst wurde.
- NonRecoverable (Siemens.Engineering.NonrecoverableException)
Diese Ausnahme schließt das TIA Portal und die Verbindung zum TIA Portal wird getrennt. Sie müssen das TIA Portal mit der TIA Portal Openness-Anwendung erneut starten.

Programmcode

Das folgende Beispiel zeigt die Möglichkeiten, die Sie haben, um auf Ausnahmen zu reagieren:

```
try
{
    ...
}

catch(EngineeringSecurityException engineeringSecurityException)
{
    Console.WriteLine(engineeringSecurityException);
}

catch(EngineeringObjectDisposedException engineeringObjectDisposedException)
{
    Console.WriteLine(engineeringObjectDisposedException.Message);
}

catch(EngineeringNotSupportedException engineeringNotSupportedException)
{
    Console.WriteLine(engineeringNotSupportedException.MessageData.Text);
    Console.WriteLine();
    foreach(ExceptionMessageData detailMessageData in
engineeringNotSupportedException.DetailMessageData)
    {
        Console.WriteLine(detailMessageData.Text);
    }
}

catch (EngineeringTargetException)
{
    throw;
}

catch (EngineeringException)
{
    //Do not catch general exceptions
    throw;
}

catch(NonRecoverableException nonRecoverableException)
{
    Console.WriteLine(nonRecoverableException.Message);
}
```

7.30.2 Custom Exception

Einleitung

CustomException ist ein Mechanismus, durch den ein Openness API-Entwickler zahlreiche Ausnahmen zur Behandlung von Fehlerszenarien definieren und so eine langfristig stabile Fehlerbehandlung realisieren kann. Dank der Unterstützung von CustomException können Entwickler mit der Openness API ihre eigenen Custom Exceptions im EOM über EOM Designer erstellen. Die Benutzer der Openness API fangen dann diese Ausnahmen im Openness API-Code zusätzlich zu den vordefinierten Ausnahmen ab. Die im EOM definierten benutzerdefinierten Ausnahmen sollten an eine Openness-Aktion angehängt werden, die diese benutzerdefinierte Ausnahme auslösen kann, sodass Sie die Ausnahme abfangen können. Andernfalls wird eine EngineeringTargetException ausgegeben.

CustomException im EOM Designer

Der EOM Modeler erstellt eine CustomException mit dem EOM Designer. Die Ausnahme kann in einem beliebigen Namensraum erstellt werden.

CustomException in der Openness-Anwendung

Mithilfe der TIA Portal Openness-Anwendung können Sie eine CustomException direkt mit dem Openness-Code abfragen:

```
static void Main(string[] args)
{
try
{
SomeEOMType someEOMType = // Get some EOM type.
someEOMType.ActionThatThrowsException();
}
catch (AnEomCustomException ex)
{
//if AnEomCustomException is attached to someEOMType.ActionThatThrowsException() as
throwable exception.
}
catch (EngineeringTargetException ex)
{
}
```

Neuer EOM-Typ

CustomException ist ein neuer Typ im EOM, aber eine Klasse in der Siemens.Engineering.dll. CustomException sieht im EOM wie folgt aus:

```
<CustomException name="MaxCharactersExceededException" publicationLevel="Published"
createPublicationLevel="Published" accessModifier="Public">
<Product>Automation-Main</Product>
<Extends>
<Class name="Siemens.Engineering.EngineeringTargetException" />
</Extends>
<Mapping
ServerException="Siemens.Automation.CustomIdentity.BusinessLogic.Exceptions.MaxCharactersE
xceededException"> </CustomException>
```

Zuordnung

Jeder EOM Modeler sollte die Ausnahme ServerException, die zu einer Ausnahme CustomException gehört, einer Server-Ausnahme in der Clientkomponente zuordnen, sodass im Openess-Code für Benutzer der Openess API eine CustomException ausgelöst wird, wenn die Clientkomponente die Server-Ausnahme auslöst. Die Server-Ausnahme wird als vollständiger Name der Ausnahme im Server (Clientkomponente) angegeben. Die im Client definierte Server-Ausnahme muss von Siemens.Automation.CommonServices.UserExceptionBase abgeleitet sein.

Publikationsstufe

Die Publikationsstufe für CustomException darf nicht "Elevated" oder "System" lauten und die Publikationsstufe einer Openess-System-Ausnahme muss "Published" sein.

Erweiterungen

Die von Ihnen erstellte CustomException muss entweder von EngineeringTargetException oder von einer Ableitung von EngineeringTargetException abgeleitet sein. Der EOM Designer bietet die Möglichkeit, die Ausnahme zu erweitern.

Regeln

Für EOM Modeler zeigt der EOM Designer Fehler und Generator-Fehler beim Generieren der Engineering-Assembly an, falls Regelverstöße in folgenden Szenarien auftreten:

- Fehler:
 - Der Name der Custom Exception muss auf "Exception" enden. Beispiel: MaxCharactersExceededException
 - Der Zugriffsmodifikator für eine Custom Exception muss "Public" sein.
 - Er sollte entweder von EngineeringTargetInvocationException oder von einer anderen Ableitung von EngineeringTargetInvocationException abgeleitet sein.
 - Für ServerException ist eine Zuordnung der Ausnahme erforderlich, damit die Ausnahme clientseitig ausgelöst werden kann. Andernfalls ist die Ausnahme im EOM zwar verfügbar, kann aber von keiner Komponente ausgelöst werden.
 - Die bereitgestellte ServerException muss eindeutig sein.
 - Die Publikationsstufe der CustomException darf nicht "Elevated" oder "System" lauten.
- Info
 - Wenn die auslösbare Ausnahme ("Throwable") an eine Aktion "Create" angehängt wird, wird sie nur für die Dokumentation der "Create"-Aktion verwendet.

Diese Regeln gelten nur für CustomException-Ausnahmen, die von Openness-Anwendern erstellt wurden. Beispiel: SampleException.

Die oben genannten Regeln gelten nicht für die nachstehend aufgeführten Openness-System-Ausnahmen:

- EngineeringException
- EngineeringSecurityException
- EngineeringObjectDisposedException
- EngineeringNotSupportedException
- EngineeringTargetInvocationException
- EngineeringRuntimeException
- EngineeringOutOfMemoryException
- EngineeringUserAbortException
- EngineeringDelegateInvocationException
- NonRecoverableException

Auslösbare Ausnahme einer Aktion

Openness API-Entwickler können mit Openness Designer eine CustomException an eine Aktion anhängen. System-Ausnahmen in Openness, wie EngineeringTargetInvocationException, EngineeringObjectDisposedException usw., können nur an bestimmte Aktionen angehängt werden. Eine CustomException kann von Ihnen nur abgefangen werden, wenn die von der TIA-Portal-Server-Komponente ausgelöste Server-Ausnahme eine EOM CustomException-Zuordnung aufweist und die Ausnahme CustomException an die aufgerufene Aktion angehängt wird.

Wenn eine Custom Exception an eine spezielle Aktion wie eine "Create"- oder "Delete"-Aktion angehängt wird, wird die auslösbarer Ausnahme nur für die API-Dokumentation verwendet.

Wenn zu der Ausnahme von der Clientkomponente eine zugeordnete EOM Custom Exception auf der EOM-Seite vorhanden ist, haben Sie die Möglichkeit, die Ausnahme CustomException im Hinblick auf spezielle Aktionen (z. B. "Create"-Aktion) abzufangen.

Regeln für auslösbarer Ausnahmen

- Für eine Custom Exception sind nur die Publikationsstufen "Published", "Developer" und "Pilot" zulässig.
- Die Stufen "Elevated" und "System" sind für eine Custom Exception nicht zulässig.
- Eine Custom Exception kann nur an Aktionen mit folgenden Publikationsstufen angehängt werden:

Publikationsstufe der EOM-Aktion	Publikationsstufe der Custom Exception	Ist Custom Exception bei der Aktion zulässig?
Published	Published	Ja
Published	Developer	Nein
Published	Pilot	Nein
Developer	Developer	Ja
Developer	Published	Ja
Developer	Pilot	Nein
Pilot	Pilot	Ja
Pilot	Published	Ja
Pilot	Developer	Nein
Elevated	Pilot	Nein
Elevated	Published	Ja
Elevated	Developer	Nein
System	Published	Ja
System	Pilot	Nein
System	Developer	Nein

Hinweis

- Wenn die Publikationsstufen der EOM-Aktion und der Custom Exception nicht kompatibel sind, zeigt OPNS Designer einen Fehler an.
- Wenn es zu Kompatibilitätsfehlern bei Publikationsstufen kommt, schlägt die Generierung der Engineering-Assembly fehl.

Export/Import

8.1 Überblick

8.1.1 Grundlagen zum Import/Export

Einleitung

Sie können bestimmte Konfigurationsdaten exportieren und die Daten nach dem Editieren in dasselbe oder ein anderes Projekt reimportieren.

Hinweis

Im Zusammenhang mit dieser Beschreibung gibt es keinerlei Verpflichtungen oder Garantien, die Quelldatei manuell zu ändern und auszuwerten. Siemens übernimmt deshalb keine Haftung für den Gebrauch dieser Beschreibung als Ganzes oder in Teilen.

Exportierbare und importierbare Objekte

Die folgenden Konfigurationsdaten können auch mit TIA Portal Openess APIs importiert oder exportiert werden:

Tabelle 8-1 Projekte

Objekte	Export	Import
Grafiksammlung	X	X

Tabelle 8-2 PLC

Objekte	Export	Import
Bausteine	X	X
Knowhow-geschützte Bausteine	X	-
Fehlersichere Bausteine	X	X
Systembausteine	X	-
PLC-Variablenlisten	X	X
Technologieobjekte	X	X
PLC-Variablen und -Konstanten	X	X
Anwenderdatentypen	X	X

Tabelle 8-3 HMI

Objekte	Export	Import
Bilder	X	X
Bildvorlagen	X	X
Globale Bilder	X	X
Pop-up-Bilder	X	X
Slide-in-Bilder	X	X
Skripte	X	X
Textlisten	X	X
Grafiklisten	X	X
Zyklen	X	X
Verbindungen	X	X
Variablen	X	X
Variablen	X	X

Vollständiger Export oder Export offener Referenzen

Die oben aufgeführten Objekttypen werden mit allen Objekten exportiert oder importiert, wenn diese demselben Unterbaum angehören. Diese Regel gilt auch für referenzierte Objekte im selben Unterbaum.

Bei referenzierten Objekten in anderen Unterbäumen ist jedoch ein vollständiger Export oder Import nicht möglich. Stattdessen werden "offene Referenzen" zu diesen Objekten exportiert oder importiert.

Referenzierte Objekte aus demselben Unterbaum werden nur exportiert, wenn sie der Gruppe exportierbarer Objekte angehören. Alle Dynamisierungen an Objekten werden beim Importieren/Exportieren als Objekte behandelt und ebenfalls exportiert/importiert.

Der Export umfasst alle Objektattribute, die bei der Konfiguration geändert wurden. Das gilt unabhängig davon, ob das geänderte Attribut genutzt wird oder nicht.

Beispiel: Sie haben ein grafisches EA-Feld mit dem Modus "Eingabe/Ausgabe" konfiguriert und die Einstellung "Sichtbar nach Anklicken" für das Attribut "Rollbalken" ausgewählt. Im Laufe der Konfiguration haben Sie den Modus in "Zwei Zustände" geändert. Das Attribut "Rollbalken" ist in diesem Modus nicht verfügbar. Weil das Attribut "Rollbalken" geändert wurde, wird es in den Export einbezogen, selbst wenn das Attribut nicht genutzt wird.

Geöffnete Referenzen importieren

Sie können auch Objekte mit geöffneten Referenzen importieren (siehe Import von Projektierungsdaten (Seite 780)).

Wenn die referenzierten Objekte im Zielprojekt enthalten sind, werden die geöffneten Referenzen automatisch wieder mit den Objekttypen verknüpft. Diese Objekte müssen am gleichen Ort verfügbar sein und dem gleichen Namen wie für den Export zugeordnet sein. Wenn die referenzierten Objekte nicht im Zielprojekt enthalten sind, können die geöffneten

Referenzen nicht aufgelöst werden. Es wird kein zusätzliches Objekt erzeugt, um diese geöffneten Referenzen aufzulösen.

Dateiformat exportieren und importieren

Das Dateiformat zum Exportieren und Importieren ist XML. Nur CAx-Daten erfordern das AML-Format. Die unterschiedlichen Schemadefinitionen für sämtliche Formate werden in dem jeweiligen Abschnitt in diesem Handbuch beschrieben:

- XML-Format für die Daten eines HMI-Geräts (Seite 788)
- XML-Format für die Daten eines PLC-Geräts (Seite 841)
- AML-Format für CAx-Daten (Seite 919)

Schriftarten importieren und exportieren

Für Objekte definierte Schriftarten werden ebenfalls exportiert und importiert.

Wenn Sie nicht im Projekt enthaltene Schriftarten importieren, wird nach dem Import die Standardschriftart am Objekt angezeigt. Jedoch wird die importierte Schriftart in der Datenverwaltung gespeichert.

Wenn die Attribute für eine Schriftart nicht in einer Importdatei zuwiesen sind, werden die Attribute nach dem Import mit Standardwerten belegt.

Technologieobjekte importieren und exportieren

Die folgenden Technologieobjekte der Version \geq V5.0 können ab TIA Portal \geq V16 für S7-1500 und S7-1500T exportiert und importiert werden:

- SpeedAxis
- PositioningAxis
- SynchronousAxis
- ExternalEncoder
- Cam
- OutputCam
- CamTrack
- Kinematics
- LeadingAxisProxy

Die folgenden Technologieobjekte können ab TIA Portal \geq V16 für S7-1200 exportiert und importiert werden:

- PositioningAxis/CommandTable

Die folgenden Technologieobjekte können ab TIA Portal \geq V16 für S7-300, S7-400, S7-1200, S7-1500 exportiert und importiert werden:

- PID

Einschränkungen

Das Exportformat ist intern und ausschließlich für die aktuelle Version von TIA Portal Openness gültig. Das Exportformat kann sich in künftigen Versionen ändern.

Alle beim Import und Export auftretenden Fehler werden als Ausnahmen gemeldet. Weitere Informationen über Ausnahmen siehe Abschnitt Umgang mit Exceptions (Seite 766).

Siehe auch

[Einsatzgebiet von Import/Export \(Seite 776\)](#)

[Export von Projektierungsdaten \(Seite 778\)](#)

8.1.2 Einsatzgebiet von Import/Export

Einleitung

Die Funktionalität zum Importieren/Exportieren ermöglicht Ihnen, bestimmte Objekte gezielt zu exportieren.

Sie können die exportierten Daten in einem externen Programm editieren oder unverändert in anderen TIA-Portal-Projekten wiederverwenden.

Wenn Sie die Importdatei richtig strukturieren, können Sie auch extern erzeugte Konfigurationsdaten importieren, ohne zuerst einen Export durchführen zu müssen.

Hinweis

Wenn Sie extern erzeugte Konfigurationsdaten mit Codefehlern oder falscher Struktur importieren, kann dies unerwartete Fehler verursachen.

Anwendungsbereich

Exportieren und Importieren von Daten ist nützlich für folgende Aufgaben:

- Externes Editieren von Konfigurationsdaten.
- Importieren von extern erzeugten Konfigurationsdaten, z. B. Textlisten und Variablen.
- Verteilen von spezifizierten Konfigurationsdaten an verschiedene Projekte, z. B. ein geändertes Prozessbild, das in mehreren Projekten verwendet werden soll.
- Zum Replizieren und Anpassen der Hardware-Konfiguration zwischen dem TIA-Portal-Projekt und einem ECAD-Programm.

Siehe auch

[Grundlagen zum Import/Export \(Seite 773\)](#)

8.1.3 Versionsspezifischer SimaticML-Import

Anwendung

Ab TIA Portal Openness V14 SP1 ist der SimaticML-Import versionsübergreifend verwendbar. Sie können Ihre älteren Exportdateien mindestens in die nächsten zwei größeren Versionen importieren.

Jede Version der Openness API kann SIMATIC-ML-Dateien aus der entsprechenden Version und aus jeder unterstützten Version des vorherigen Release importieren; beispielsweise wird der Import von SIMATIC-ML-Dateien V14 SP1, V15 und V15.1 in Openness API V16 unterstützt.

Die folgende Tabelle zeigt ein Beispiel dafür, welche SIMATIC-ML-Version von welcher Openness API-Version importiert werden kann.

Openness API Versions-stand	SIMATIC-ML-Datei V14 SP1	SIMATIC-ML-Datei V15	SIMATIC-ML-Datei V15.1	SIMATIC ML V16
V14 SP1	Import unterstützt	Import nicht untersttzt	Import nicht untersttzt	Import nicht untersttzt
V15 SP1	Import unterstützt	Import unterstützt	Import nicht untersttzt	Import nicht untersttzt
V15.1	Import unterstützt	Import unterstützt	Import untersttzt	Import nicht untersttzt
V16	Import unterstützt	Import unterstützt	Import untersttzt	Import unterstützt

Die einzelnen Versionen der Openness API unterstützen den Export von SIMATIC-ML-Dateien. Die Version der exportierten SIMATIC-ML-Datei sollte jedoch mit der Version des TIA Portals und nicht mit der Version der verwendeten Openness API übereinstimmen.

Um dieses Merkmal zu unterstützen, enthalten SimaticML-Dateien jetzt die Modellversionsinformationen wie nachstehend dargestellt:

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
  <Engineering version="V14 SP1"/>
  <DocumentInfo>
    ...
  </DocumentInfo>
  <SW.DataBlock ID="0">
    ...
  </SW.DataBlock>
</Document>
```

Hinweis

Wenn die Versionsinformationen nicht in der SimaticML-Datei enthalten sind, verwendet das System die aktuelle Modellversion.

8.1.4 Bearbeiten der XML-Datei

Einleitung

Zur Bearbeitung einer XML-Datei für den Import von Projektierungsdaten verwenden Sie einen XML-Editor oder einen Texteditor.

Wenn Sie umfangreiche Änderungen durchführen oder selbst Objektstrukturen aufbauen, empfehlen wir die Verwendung eines XML-Editors mit Auto-Vervollständigungsfunktion.

Hinweis

Ändern des XML-Inhalts setzt umfangreiche Kenntnisse der Struktur und der Validierungsregeln in XML voraus. Vermeiden Sie Validierungsfehler und arbeiten Sie nur in Ausnahmefällen manuell in der XML-Struktur.

8.1.5 Export von Projektierungsdaten

Einleitung

Die Konfigurationsdaten jedes Startobjekts (Stamm) werden separat in eine XML-Datei exportiert.

Zum Editieren der Exportdatei sind angemessene Kenntnisse von XML erforderlich. Verwenden Sie für komfortables Editieren einen XML-Editor.

Beispiel

Sie haben ein Prozessbild, das ein EA-Feld enthält. In dem EA-Feld ist eine externe Variable konfiguriert. Ein Export des Prozessbilds enthält das Bild und das EA-Feld. Die Variable und die von der Variable verwendete Verbindung werden nicht exportiert. Stattdessen wird nur eine offene Referenz in den Export einzogen.

Inhalt der Exportdatei

Beginnend mit dem Startobjekt werden alle Objekte eines Unterbaums und deren Attribute in der Exportdatei gespeichert. Alle Verweise auf Objekte anderer Unterbäume werden nur als offene Verweise exportiert. Die entsprechenden Attribute der referenzierten Objekte in verschiedenen Unterbäumen werden nicht in die Exportdatei geschrieben.

Hinweis

Der Export von Objekttypen aus der Bibliothek wird nicht unterstützt.

Sie können Objekte als Typ in der Bibliothek erzeugen. Instanzen des im Projekt verwendeten Objekttyps können wie andere Objekte mit der TIA Portal Openness-Anwendung bearbeitet werden. Wenn Sie Objekte exportieren, werden die Instanzen ohne Typinformationen exportiert.

Wenn Sie diese Objekte wieder in das Projekt importieren, werden die Instanzen des Objekttyps überschrieben und die Instanz wird vom Objekttyp abgelöst.

Die Exportdatei enthält nicht notwendigerweise alle Attribute eines Objekts. Sie legen fest, welche Daten exportiert werden sollen:

- `ExportOptions.None`
Diese Einstellung exportiert nur die geänderten Daten oder die Daten, die von den Standardwerten abweichen.
Die Exportdatei enthält auch alle Werte, die für den anschließenden Datenimport obligatorisch sind.
- `ExportOptions.WithDefaults1`
Auch die Standardwerte werden exportiert.
- `ExportOptions.WithReadOnly1`
Auch die schreibgeschützten Werte werden exportiert.

¹: Sie können diese beiden Optionen mit der folgenden Syntax kombinieren:

```
Export(path,ExportOptions.WithDefaults |  
ExportOptions.WithReadOnly);
```

Sämtliche Inhalte der Exportdatei liegen auf Englisch vor. Unabhängig davon werden enthaltene Projekttexte in alle vorhandenen Sprachen exportiert und importiert.

Alle Konfigurationsdaten werden als XML-Objekte in der Exportdatei modelliert.

Siehe auch

[Grundlagen zum Import/Export \(Seite 773\)](#)

[Bausteine exportieren \(Seite 898\)](#)

8.1.6 Import von Projektierungsdaten

Einleitung

Konfigurationsdaten werden aus einer zuvor exportierten und editierten XML-Datei oder aus einer von Ihnen selbst erzeugten XML-Datei importiert. Die in dieser Datei enthaltenen Daten werden beim Import geprüft. Dieser Ansatz verhindert, dass die Konfigurationsdaten im TIA Portal infolge des Imports inkonsistent werden.

Einschränkungen

- Alle Stammobjekte in der Importdatei müssen vom gleichen Typ sein, z. B. Variablenklassen, Bausteine usw.
- Wenn eine Importdatei mehrere Stammobjekte enthält und eines davon nicht gültig ist, werden sämtliche Inhalte der Importdatei nicht importiert.
- Beim Importieren von Texten müssen die entsprechenden Projektsprachen im Zielprojekt eingerichtet worden sein, um ein Fehlschlagen des Imports zu vermeiden. Bei Bedarf können Sie die Spracheinstellungen über TIA Portal Openness ändern.
- Wenn Sie ungültige Attribute eines Objekts in der Importdatei angeben, die nicht in der grafischen Benutzeroberfläche des TIA Portals editiert werden können, wird der Import abgebrochen.
- Nur die im Feld "separately for each connection" aufgeführten Bereichszeiger können importiert oder exportiert werden.
- Der Import von Objekttypen aus der Bibliothek wird nicht unterstützt. Sie können Objekte als Typ in der Bibliothek erzeugen. Instanzen des im Projekt verwendeten Objekttyps können wie andere Objekte mit der TIA Portal Openness-Anwendung bearbeitet werden. Wenn Sie Objekte exportieren, werden die Instanzen ohne Typinformationen exportiert. Wenn Sie diese Objekte wieder in das Projekt importieren, werden die Instanzen des Objekttyps überschrieben und die Instanz wird vom Objekttyp abgelöst.
- Der Import fehlersicherer Bausteine wird nicht unterstützt.

Hinweis

Geräteabhängige Wertebereiche für grafische Attribute

Wenn Werte grafischer Attribute außerhalb des gültigen Wertebereichs liegen, werden diese Werte beim Import auf die möglichen Maximalwerte für das HMI-Gerät zurückgesetzt.

Unterschiedliches Importverhalten

Wenn bereits im Projekt vorhandene Objekte importiert werden sollen, müssen Sie das Importverhalten mit verschiedenen Programmcodes steuern. Andernfalls werden die Objekte beim Import wieder im Projekt erzeugt.

Die folgenden Einstellungen für das Importverhalten sind möglich:

- `ImportOptions.None`
Bei dieser Einstellung werden die Konfigurationsdaten ohne Überschreiben importiert. Wenn ein Objekt aus einer bereits im Projekt vorhandenen XML-Datei importiert wird, wird der Import unterbrochen und eine Ausnahme ausgelöst.
- `ImportOptions.Override`
Bei Verwendung dieser Einstellung werden die Konfigurationsdaten mit automatischem Überschreiben importiert.
Sie können angeben, dass im Projekt vorhandene Objekte beim Import überschrieben werden sollen. Relevante Objekte werden vor dem Import gelöscht und mit Standardwerten neu erzeugt. Diese Standardwerte werden beim Import mit den importierten Werten überschrieben. Wenn das vorhandene Objekt und das neue Objekt nicht in derselben Gruppe sind, kann kein Überschreiben stattfinden. Um Namenskonflikte zu vermeiden, wird der Import abgebrochen und eine Ausnahme ausgelöst.

Vorgehensweise zum Importieren

Wenn Sie eine XML-Datei importieren möchten, müssen die darin enthaltenen Daten bestimmten Regeln entsprechen. Die Inhalte der Importdatei müssen wohlgeformt sein. Es dürfen keine Syntaxfehler und keine Datenstrukturfehler vorhanden sein. Verwenden Sie bei umfangreichen Änderungen einen XML-Editor, der diese Kriterien vor dem Import überprüft.

Beim Import der XML-Datei in das TIA Portal werden die Daten in der Datei zuerst auf formale Fehler im XML-Code geprüft. Wenn bei dieser Prüfung Fehler erkannt werden, wird der Import abgebrochen und die Fehler werden in einer Ausnahme angezeigt (siehe Umgang mit Exceptions (Seite 766)).

Siehe auch

[Grundlagen zum Import/Export \(Seite 773\)](#)

[Anwenderdatentyp importieren \(Seite 917\)](#)

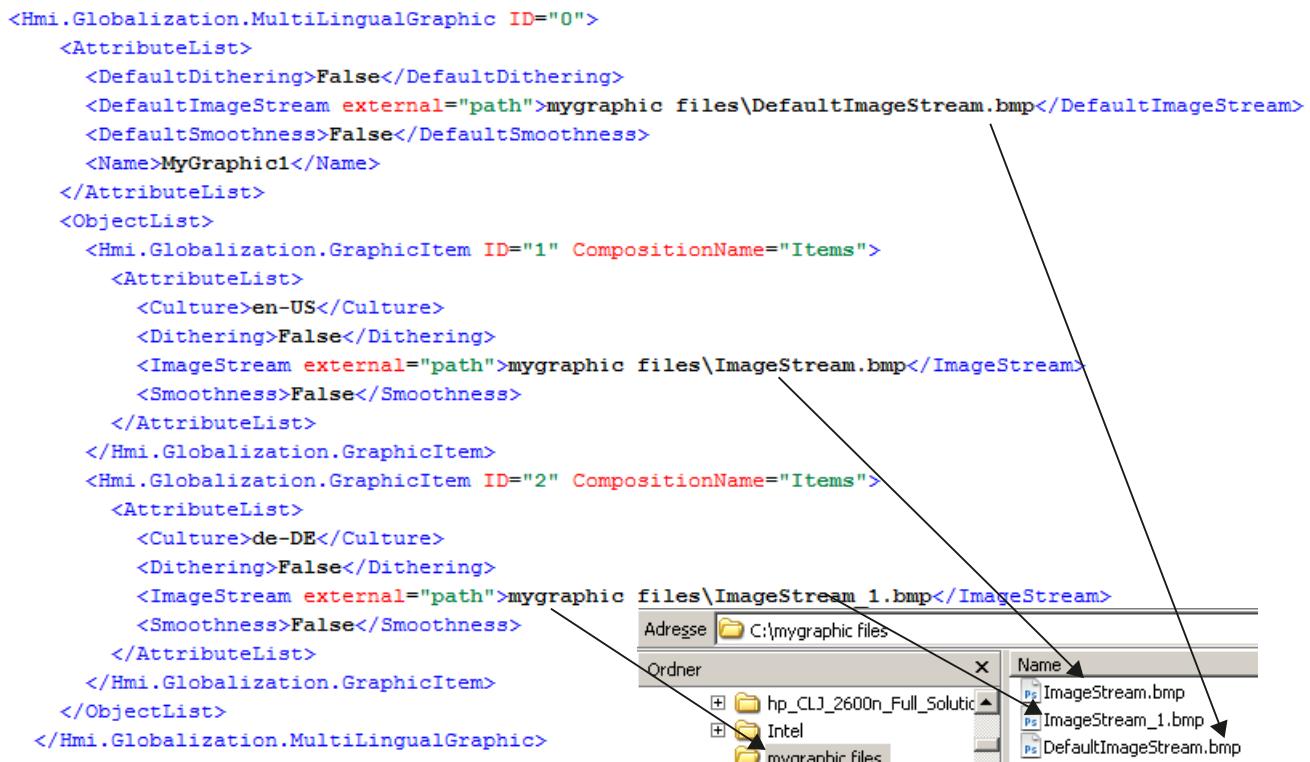
8.2 Import/Export von Projektdaten

8.2.1 Grafiksammlung

8.2.1.1 Export/Import von Grafiken

Einleitung

Der Export von Konfigurationsdaten aus dem TIA Portal in die XML-Datei beinhaltet keine ausgewählten Grafiken oder Grafiken, auf die durch ein Objekt verwiesen wird. Die Grafiken werden beim Export separat gespeichert. In der XML-Datei wird anhand eines relativen Pfads und die Dateinamen auf die Grafiken verwiesen. Eine Grafikreferenz wird in der XML-Datei als ein Objekt modelliert und enthält eine Attributliste und bei Bedarf eine Verknüpfungsliste wie bei den anderen Objekten.



Grafiken exportieren

Der Export von Konfigurationsdaten beinhaltet ausschließlich Grafiken, die für den Export direkt ausgewählt wurden. Die exportierbaren Grafiken werden für die jeweilige Sprache im TIA Portal gespeichert. Wird ein Projekt in mehreren Sprachen konfiguriert, dann werden alle verwendeten Sprachversionen exportiert.

Beim Grafikexport wird ein neuer Ordner im Exportordner erstellt. Der Dateiordnername wird durch Verknüpfen des XML-Dateinamens mit "Dateien" aufgebaut. Dieser Ordner enthält die exportierten Grafiken. Falls dieser Ordner bereits vorhanden ist, wird ein neuer Ordner erstellt und durch eine laufende Nummer ergänzt.

Die Grafiken werden in demselben Dateiformat wie im Projekt gespeichert. Das Datenformat wird nicht geändert oder konvertiert und die Auflösung und Farbtiefe bleiben unverändert.

Die Kennung "default" wird als Dateierweiterung für die als Standardsprache ausgewählte Sprache verwendet.

Wenn der Ordner bereits eine Datei mit demselben Namen enthält, wird der Dateiname der exportierten Grafik durch eine laufende Nummer ergänzt.

Grafiken importieren

Beim Importieren von Grafiken gelten die folgenden Voraussetzungen:

- Die Grafiken müssen ein vom TIA Portal unterstütztes Dateiformat besitzen.
- Auf die Grafiken muss in der XML-Datei durch Angabe eines relativen Pfads verwiesen werden.

Nach dem Export einer Grafik kann diese mithilfe eines Grafikprogramms außerhalb des TIA Portals bearbeitet und anschließend wieder importiert werden.

Siehe auch

Grundlagen zum Import/Export (Seite 773)

8.2.1.2 Grafiken eines Projektes exportieren

Voraussetzungen

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Sie können entweder eine einzelne Grafik oder alle Grafiken der Grafiksammlung eines Projekts in allen Sprachen exportieren. Während des Exports wird eine XML-Datei mit allen betroffenen Projektgrafikeinträgen erstellt und zusammen mit den exportierten Grafiken referenziert. Die entsprechenden Grafiken werden zusammen mit der XML-Datei im selben Verzeichnis des Dateisystems gespeichert.

Damit die exportierten Grafiken ("*.jpg", "*.bmp", "*.png", "*.ico", etc.) geändert werden können, sind diese Grafiken nicht schreibgeschützt.

Programmcode: Grafik exportieren

Um die erforderliche Grafik zu exportieren, ändern Sie den folgenden Programmcode:

```
//Exports all language variants of a single graphic
Project project = ...;
MultiLingualGraphicComposition graphicsComposition = project.Graphics;
MultiLingualGraphic graphic = graphicsComposition.Find("graphicName");
graphic.Export(new FileInfo(@"D:\ExportFolder\graphicName.xml"),
ExportOptions.WithDefaults);
```

Programmcode: Alle Grafiken exportieren

Um alle Grafiken einer Grafiksammlung zu exportieren, ändern Sie den folgenden Programmcode:

```
//Exports all graphics of a graphic library
Project project = ...;
MultiLingualGraphicComposition graphicsComposition = project.Graphics;
foreach(MultiLingualGraphic graphic in graphicsComposition)
{
    graphic.Export(new FileInfo(string.Format(@"D:\Graphics\{0}.xml", graphic.Name)),
ExportOptions.WithDefaults);
}
```

8.2.1.3 Grafiken in ein Projekt importieren

Voraussetzungen

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Eine XML-Datei wird zusammen mit den Sprachversionen einer Grafik in einem Verzeichnis Ihres Dateisystems gespeichert.

Sie können alle Grafiken in einem relativen Pfad in der XML-Datei referenzieren.

Sie können jetzt alle Sprachversionen einer in der XML-Datei enthaltenen Grafik in die Grafiksammlung importieren.

Beachten Sie auch Folgendes: Import von Projektierungsdaten (Seite 780).

Programmcode

Um eine oder mehrere Grafiken zu importieren, ändern Sie den folgenden Programmcode:

```
//Import all language variants of a single graphic
Project project = ...;
MultiLingualGraphicComposition graphicComposition = project.Graphics;
graphicComposition.Import(new FileInfo(@"D:\Graphics\Graphic1.xml"),
ImportOptions.Override);
```

8.2.2 Projekttexte

8.2.2.1 Export von Projekttexten

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Im TIA Portal finden Sie Projekttexte unter dem Knoten "Sprachen & Ressourcen" eines Projekts. Diese Texte werden in eine *.xlsx-Datei exportiert, die zum Beispiel für Übersetzungen genutzt werden kann. Die Einschränkungen beim Exportieren und Importieren von Projekttexten sind die gleichen wie in der Benutzeroberfläche. Diese Einschränkungen umfassen:

- Exportierte Texte können nur in das Projekt importiert werden, aus dem sie exportiert wurde.
- Texte können nur in Sprachen übersetzt werden, die im Projekt verfügbar sind. Bei Bedarf können Sie Projektsprachen über TIA Portal Openness hinzufügen.
- Nur vorhandene Texte können reimportiert werden. Wenn Texte aus dem ursprünglichen Projekt gelöscht oder neu erzeugt wurden, schlägt der Import dieser Texte fehl.

Sie müssen die folgenden Parameter festlegen:

Name	Beispiel	Beschreibung
pah	new FileInfo ("D:\Test\Project-Text.xlsx")	Pfad zur Exportdatei
sourceLanguage	new CultureInfo("en-US")	Referenzsprache, aus der Text zu übersetzen ist
targetLanguage	new CultureInfo("de-DE")	Zielsprache, in die Text zu übersetzen ist

Hinweis

Mehrsprachige Texte werden zusammen mit dem übergeordneten Objekt, zu dem sie gehören, exportiert. Mehrsprachige Texte können nicht explizit exportiert werden.

Programmcode: Aus dem Knoten "Sprachen & Ressourcen" exportieren

Die Verwendung der Beispielparameter führt zum folgenden Programmcode zum Exportieren von Projekttexten:

```
project.ExportProjectTexts(new FileInfo(@"D:\Test\ProjectText.xlsx"), new CultureInfo("en-US"), new CultureInfo("de-DE"));
```

XML-Struktur eines exportierten mehrsprachigen Textelements

```
...
<MultilingualText ID="2" CompositionName="Comment">
  <ObjectList>
    <MultilingualTextItem ID="3" CompositionName="Items">
      <AttributeList>
        <Culture>en-US</Culture>
        <Text>My super tag</Text>
      </AttributeList>
    <MultilingualTextItem ID="4" CompositionName="Items">
      <AttributeList>
        <Culture>ru-RU</Culture>
        <Text>Мої супер теги</Text>
      </AttributeList>
    </ObjectList>
  </MultilingualText>
...
...
```

8.2.2.2 Import von Projekttexten

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Im TIA Portal finden Sie Projekttexte unter dem Knoten "Sprachen & Ressourcen" eines Projekts. Sie können Projekttexte aus einer *.xlsx-Datei importieren, was zum Beispiel für Übersetzungen genutzt werden kann. Die Einschränkungen beim Exportieren und Importieren von Projekttexten sind die gleichen wie in der Benutzeroberfläche. Diese Einschränkungen umfassen:

- Exportierte Texte können nur in das Projekt importiert werden, aus dem sie exportiert wurde.
- Übersetzte Texte können nur in Sprachen importiert werden, die in dem Projekt, aus dem sie exportiert wurden, verfügbar sind.
- Nur vorhandene Texte können reimportiert werden. Wenn Texte aus dem ursprünglichen Projekt gelöscht oder neu erzeugt wurden, schlägt der Import dieser Texte fehl.

Sie müssen die folgenden Parameter festlegen:

Name	Beispiel	Beschreibung
path	new FileInfo(@"D:\Test\Project-Text.xlsx")	Pfad zur Importdatei
updateSourceLanguage	true	<p>Wenn "true", wird der Text der Referenzsprache anhand der Exportdatei aktualisiert.</p> <p>Wenn "false", wird der Text der Referenzsprache nicht aktualisiert.</p>

Hinweis

Mehrsprachige Texte werden zusammen mit dem übergeordneten Objekt, zu dem sie gehören, importiert. Mehrsprachige Texte können nicht explizit importiert werden.

Programmcode

Die Verwendung der Beispielparameter führt zum folgenden Programmcode zum Importieren von Projekttexten:

```
ProjectTextResult result = project.ImportProjectTexts(new FileInfo(@"D:\Test\ProjectText.xlsx"), true);
```

Der Import der Projekttexte gibt ein Objekt zurück, das den Status des Imports anzeigt und den Pfad angibt, in dem das Importprotokoll gespeichert wird. Auf diese Attribute kann mit dem folgenden Code zugegriffen werden:

```
ProjectTextResultState resultState = result.State;
FileInfo logFilePath = result.Path;
```

8.3 Import/Export von Daten eines HMI-Geräts

8.3.1 Aufbau einer XML-Datei

Einleitung

Die Daten in der Exportdatei vom Import/Export sind mit Verweis auf eine Grundstruktur gegliedert.

Grundstruktur einer Exportdatei

Die Exportdatei wird in einem XML-Format erzeugt.

Die XML-Datei beginnt mit einer Dokumentinformation. Sie beinhaltet die Daten der rechnerspezifischen Installation, mit der das Projekt exportiert wurde.

Die Exportdatei ist in die folgenden zwei Abschnitte unterteilt:

- Informationen über das Dokument

In diesem Abschnitt können Sie Ihre eigenen Informationen über den Export in gültiger XML-Syntax eingeben. Der Inhalt wird beim Import ignoriert.

Zum Beispiel können Sie einen Baustein mit `<IntegrityInformation>...</IntegrityInformation>` einfügen und darin zusätzliche Informationen über die Validierung unterbringen. Nach Weiterleitung der XML-Datei kann der Empfänger anhand dieses Bausteins vor dem Import prüfen, ob die XML-Datei geändert wurde.

- Objekt

Dieser Abschnitt enthält die zu exportierenden Elemente.

```

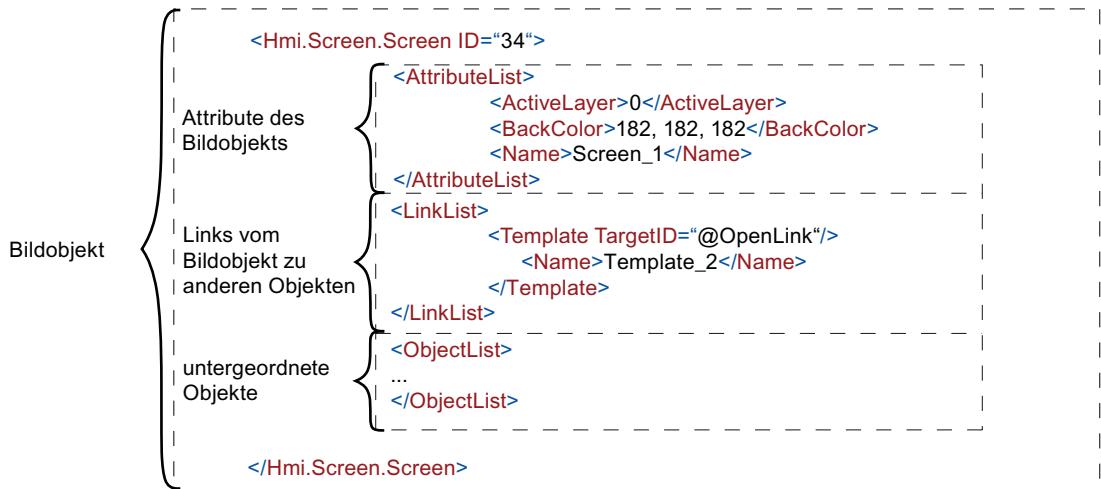
<?xml version="1.0" encoding="UTF-8" ?>
<Document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <DocumentInfo>
    <UserName>Jane Doe</UserName>
    <Company>Example_Inc</Company>
    <IntegrityInformation>...</IntegrityInformation>
    <Created>2016-04-28T18:05:42.179207Z</Created>
    <ExportSetting>WithDefaults</ExportSetting>
    <InstalledProducts>
      <Product>
        <DisplayName>Totally Integrated Automation Portal</DisplayName>
        <DisplayVersion>V14</DisplayVersion>
      </Product>
      <OptionPackage>
        <DisplayName>WinCC Professional</DisplayName>
        <DisplayVersion>V14</DisplayVersion>
      </OptionPackage>
      <OptionPackage>
        <DisplayName>Siemens TIA Openness</DisplayName>
        <DisplayVersion>V14</DisplayVersion>
      </OptionPackage>
    </InstalledProducts>
  </DocumentInfo>
  <Hmi.Screen.Screen ID ="0">
    <AttributeList>
      <ActiveLayer>0</ActiveLayer>
      <BackColor>189,190,0</BackColor>
      <Height>422</Height>
      <Name>Root screen</Name>
      <Number>1</Number>
      <Visible>True</Visible>
      <Width>640</Width>
    </AttributeList>
    <LinkList>
      <Template TargetID="@OpenLink">
        <Name>Template_1</Name>
      </Template>
    </LinkList>
    <ObjectList>
      <Name>dummy</Name>
    </ObjectList>
  </Hmi.Screen.Screen>
</Document>

```

The XML code represents a document structure. It includes sections for document information (User Name, Company, Integrity Information, Creation Date, Export Settings) and installed products (specifically mentioning the Totally Integrated Automation Portal, WinCC Professional, and Siemens TIA Openness). The main focus is a screen object with ID "0", which has attributes like Active Layer (0), Back Color (189,190,0), Height (422), Name (Root screen), Number (1), and Visible (True). It also contains a LinkList with a single template entry named "Template_1" and an ObjectList containing a single object named "dummy". Brackets on the left side group the code into two main sections: "Informationen über das Dokument" (covering DocumentInfo and InstalledProducts) and "Bildobjekt" (covering the Hmi.Screen.Screen section).

Bildobjekte einer Exportdatei

Die exportierten Elemente sind in zusätzlichen Elementen der XML-Datei verfügbar.



Siehe auch

Grundlagen zum Import/Export (Seite 773)

8.3.2 Struktur der Daten für Import/Export

Objekte

Die Grundstruktur ist für alle Objekte gleich.

Jedes Objekt in der XML-Datei beginnt mit seinem Typ, zum Beispiel "Hmi.Screen.Button", und einer ID. Die ID wird beim Export automatisch erzeugt.

```
<Hmi.Screen.Button CompositionName="ScreenItems" ID="60">
```

Jedes Objekt mit Ausnahme des Startobjekts enthält auch ein XML-Attribut "CompositionName". Der Wert für dieses Attribut ist voreingestellt. Es ist gelegentlich notwendig, dieses Attribut anzugeben, zum Beispiel zum Ändern der Beschriftung beim Drücken oder Loslassen einer Schaltfläche.

```

<MultilingualText ID="A" CompositionName="TextOff">
  <ObjectList>
    <MultilingualTextItem ID="B" CompositionName="Items">
      <AttributeList>
        <Culture>en-US</Culture>
        <Text>
          <body>
            <p>TextOff</p>
          </body>
        </Text>
      </AttributeList>
    </MultilingualTextItem>
  </ObjectList>
</MultilingualText>
<MultilingualText ID="C" CompositionName="TextOn">
  <ObjectList>
    <MultilingualTextItem ID="D" CompositionName="Items">
      <AttributeList>
        <Culture>en-US</Culture>
        <Text>
          <body>
            <p>TextOn</p>
          </body>
        </Text>
      </AttributeList>
    </MultilingualTextItem>
  </ObjectList>
</MultilingualText>

```

Attribute

Jedes Objekt enthält Attribute, die in einem Abschnitt "AttributeList" enthalten sind. Jedes Attribut ist als XML-Element modelliert, z. B. "BackColor". Der Wert eines Attributs ist als XML-Inhalt modelliert, z. B. "204, 204, 204".

```

<Hmi.Screen.Button ID="2" CompositionName="ScreenItems">
  <AttributeList>
    <BackColor>204, 204, 204</BackColor>
    <ObjectName>Button_1</ObjectName>
  </AttributeList>
</Hmi.Screen.Button>

```

Bei referenzierenden Objekten enthält jedes Objekt gegebenenfalls einen Abschnitt "LinkList". Dieser Abschnitt enthält Verknüpfungen mit anderen Objekten innerhalb oder außerhalb der XML-Datei. Jede Verknüpfung ist als XML-Element modelliert. Die Bezeichnung einer Verknüpfung wird vom Zielobjekt in der Schemadatei festgelegt. Jede Verknüpfung enthält auch das Attribut "TargetID". Wenn das Zielobjekt in der XML-Datei enthalten ist, ist der Wert des Attributs "TargetID" die ID des referenzierten Objekts mit vorangestellter Raute "#". Wenn das Zielobjekt nicht in der XML-Datei enthalten ist, hat das Attribut "TargetID" den Wert "@OpenLink". Die tatsächliche Referenz des Objekts ist als untergeordnetes XML-Element modelliert.

8.3 Import/Export von Daten eines HMI-Geräts

```
<Hmi.Tag.Tag ID="17">
  <AttributeList>
    <Name>Tag_1</Name>
  </AttributeList>
  <LinkList>
    <AcquisitionCycle TargetID="@OpenLink">
      <Name>2 s</Name>
    </AcquisitionCycle>
    <Connection TargetID="@OpenLink">
      <Name>HMI_connection</Name>
    </Connection>
  </LinkList>
</Hmi.Tag.Tag>
```

Beziehung zwischen Objekten und XML-Struktur

Die nachstehenden Bilder zeigen die Beziehung zwischen der exportierten XML-Struktur und den zugeordneten Objekten in WinCC.

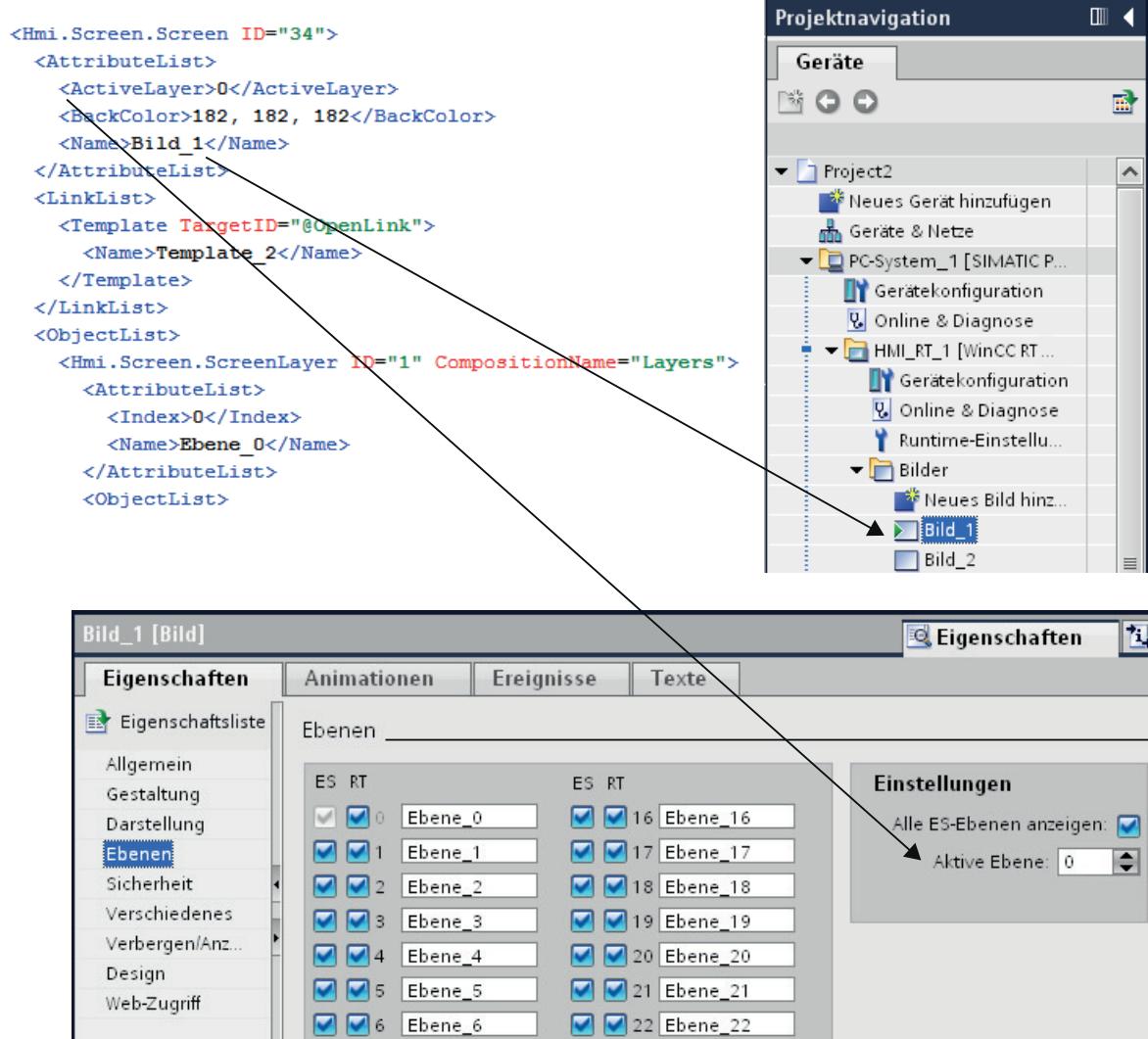


Bild 8-1 Beziehung zwischen WinCC-Benutzeroberfläche und XML-Struktur.

8.3 Import/Export von Daten eines HMI-Geräts

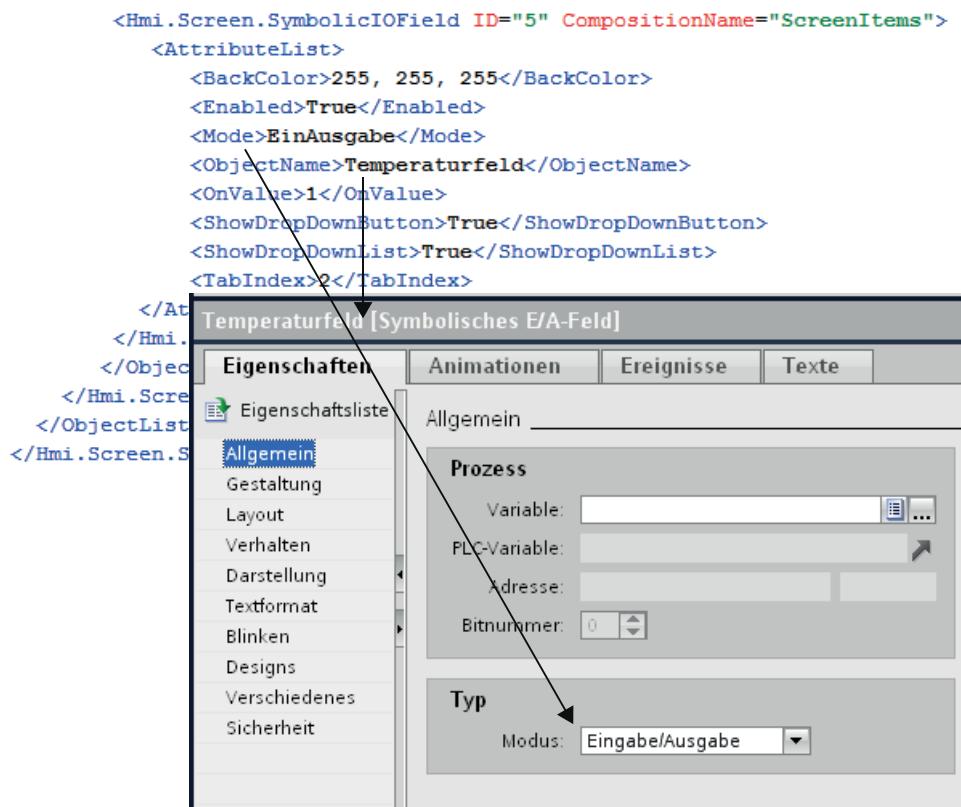


Bild 8-2 Beziehung zwischen den Einstellungen in WinCC und XML-Struktur.

8.3.3 Zyklen

8.3.3.1 Zyklen exportieren

Voraussetzungen

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Die Schnittstelle TIA Portal Openness API unterstützt den Export aller Zyklen eines bekannten HMI-Geräts in eine XML-Datei. Die Generierung der entsprechenden Exportdatei weist darauf hin, dass der Export abgeschlossen ist.

Programmcode

Um Zyklen aus einem HMI-Gerät in eine XML-Datei zu exportieren, ändern Sie den folgenden Programmcode:

```
//Exports cycles from an HMI device
private static void ExportCyclesFromHMITarget(HmiTarget hmitarget)
{
    CycleComposition cycles = hmitarget.Cycles;
    foreach(Cycle cycle in cycles)
    {
        cycle.Export(new FileInfo(string.Format(@"C:\Samples\{0}.xml", cycle.Name)), ExportOptions.WithDefaults);
    }
}
```

8.3.3.2 Zyklen importieren

Voraussetzungen

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Wenn Sie `ImportOptions.None` verwenden, können Sie anhand der Zusammensetzungszahl (Composition count) die Zyklen ermitteln, die tatsächlich importiert wurden. Sie haben Zugriff auf diese importierten Zyklen.

Hinweis

Standardzyklen haben Attribute, die in der Benutzeroberfläche nicht bearbeitet werden können. Wenn Sie in der Importdatei angeben, dass diese Attribute geändert werden sollen, verursacht der Importvorgang eine `NonRecoverableException` und schließt das TIA Portal.

8.3 Import/Export von Daten eines HMI-Geräts

Programmcode

Um einen Zyklus oder mehrere Zyklen aus einer XML-Datei in ein HMI-Gerät zu importieren, ändern Sie den folgenden Programmcode:

```
//Imports cycles to an HMI device
private static void ImportCyclesToHMITarget(HmiTarget hmitarget)
{
    CycleComposition cycles = hmitarget.Cycles;
    string dirPathImport = @"C:\OpennessSamples\Import\";
    string cycleImportFileName = "CycleImport.xml";
    string fullPath = Path.Combine(dirPathImport, cycleImportFileName);

    cycles.Import(new FileInfo(fullFilePath), ImportOptions.None);
}
```

Siehe auch

[Import von Projektierungsdaten \(Seite 780\)](#)

8.3.4 VariablenTabellen

8.3.4.1 HMI-VariablenTabellen exportieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Verwendung

Pro HMI-VariablenTabelle wird eine XML-Datei exportiert. Die API unterstützt diesen Exportvorgang. Der Export von VariablenTabellen ist auch in Unterordnern verfügbar.

Programmcode: Alle HMI-Variablenklassen aus einem angegebenen Ordner exportieren

Um alle HMI-Variablenklassen aus einem bestimmten Ordner zu exportieren, ändern Sie den folgenden Programmcode:

```
//Exports all tag tables from a tag folder
private static void ExportAllTagTablesFromTagFolder(HmiTarget hmitarget)
{
    TagSystemFolder folder = hmitarget.TagFolder;
    TagTableComposition tables = folder.TagTables;

    foreach (TagTable table in tables)
    {
        FileInfo info = new FileInfo(string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
        table.Name));
        table.Export(info, ExportOptions.WithDefaults);
    }
}
```

Programmcode: Eine HMI-Variablenklasse exportieren

Um eine einzelne HMI-Variablenklasse zu exportieren, ändern Sie den folgenden Programmcode:

```
//Exports a tag table from an HMI device
private static void ExportTagTableFromHMITarget(HmiTarget hmitarget)
{
    string tableName = "Tag table XYZ";
    TagSystemFolder folder = hmitarget.TagFolder;
    TagTableComposition tables = folder.TagTables;
    TagTable table = tables.Find(tableName);

    if (table != null)
    {
        FileInfo info = new FileInfo(string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
        table.Name));
        table.Export(info, ExportOptions.WithDefaults);
    }
}
```

Programmcode: Alle HMI-Variablenklassen exportieren

Um alle HMI-Variablenklassen zu exportieren, ändern Sie den folgenden Programmcode:

```
//Exports all tag tables from an HMI device
private static void ExportAllTagTablesFromHMITarget(HmiTarget hmitarget)
{
    TagSystemFolder sysFolder = hmitarget.TagFolder;

    //First export the tables in underlying user folder
    foreach (TagUserFolder userFolder in sysFolder.Folders)
    {
        ExportUserFolderDeep(userFolder);
    }

    //then, export all tables in the system folder
    ExportTablesInSystemFolder(sysFolder);
}

private static void ExportUserFolderDeep(TagUserFolder rootUserFolder)
{
    foreach (TagUserFolder userFolder in rootUserFolder.Folders)
    {
        ExportUserFolderDeep(userFolder);
    }
    ExportTablesInUserFolder(rootUserFolder);
}

private static void ExportTablesInUserFolder(TagUserFolder folderToExport)
{
    TagTableComposition tables = folderToExport.TagTables;
    foreach (TagTable table in tables)
    {
        string fullFilePath = string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name);
        table.Export(new FileInfo(fullFilePath), ExportOptions.WithDefaults);
    }
}

private static void ExportTablesInSystemFolder(TagSystemFolder folderToExport)
{
    TagTableComposition tables = folderToExport.TagTables;
    foreach (TagTable table in tables)
    {
        string fullFilePath = string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name);
        table.Export(new FileInfo(fullFilePath), ExportOptions.WithDefaults);
    }
}
```

8.3.4.2 HMI-Variablenliste importieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Programmcode

Um die HMI-Variablenliste einer XML-Datei in einen benutzerdefinierten Ordner oder in einen Systemordner zu importieren, ändern Sie den folgenden Programmcode:

```
//Imports a single HMI tag table from a XML file
private static void ImportSingleHMITagTable(HmiTarget hmitarget)
{
    TagSystemFolder folder = hmitarget.TagFolder;
    TagTableComposition tables = folder.TagTables;

    FileInfo info = new FileInfo(@"D:\Samples\Import\myExportedTagTable.xml");
    tables.Import(info, ImportOptions.Override);
}
```

Fehlerhafter Import von Variablen

Wenn Sie in den Namen von Variablen oder referenzierten Variablen die folgenden Symbole verwenden, kommt es beim Import der Variablen zu Fehlern:

- . (Punkt)
- \ (Backslash)

Abhilfe 1:

Vergewissern Sie sich vor einem Export, dass der Name der zu exportierenden Variable oder Bezugsvariable keinen Punkt und keinen Backslash enthält.

Abhilfe 2:

Schließen Sie in der Importdatei die Namen von Variablen oder referenzierten Variablen mit Anführungszeichen aus.

Beispiel

- Variablenname mit Symbol:
`<name>Siemens.Simatic.Hmi.Utah.Tag.HmiTag:41000_Options_Time_Date
|DB_SFX0908_HMI1.Actual_Date_Time.Hour</name>`
- Variablenname mit ausgeschlossenem Symbol in Anführungszeichen:
`<name>"Siemens.Simatic.Hmi.Utah.Tag.HmiTag:41000_Options_Time_Date
|DB_SFX0908_HMI1.Actual_Date_Time.Hour"</name>`

8.3.4.3 Einzelne Variable einer HMI-Variablenliste exportieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Verwendung

Die folgenden Objekttypen des Objektmodells können als untergeordnete Elemente einer HMI-Variable vorhanden sein und werden beim Export berücksichtigt:

MultilingualText	Für Kommentar, TagValue, DisplayName
TagArrayMemberTag	Für HMI-Array-Elemente
TagStructureMember	Für HMI-Strukturelemente
Event	Für konfigurierte Ereignisse
MultiplexEntry	Für konfigurierte Multiplex-Einträge von Variablen

Programmcode

Um eine einzelne Variable aus einer HMI-Variablenliste in eine XML-Datei zu exportieren, ändern Sie den folgenden Programmcode:

```
//Exports a selected tag from a tag table
private static void ExportSelectedTagFromTagTable(HmiTarget hmitarget)
{
    TagSystemFolder tagFolder = hmitarget.TagFolder;
    TagTable mytable = tagFolder.TagTables.Find("MyTagTable");

    TagComposition containingTags = mytable.Tags;
    Tag myTag = containingTags.Find("MyTag");

    if (myTag != null)
    {
        FileInfo info = new FileInfo(string.Format(@"C:\OpennessSamples\Tags\{0}.xml",
        myTag.Name));
        myTag.Export(info, ExportOptions.WithDefaults);
    }
}
```

8.3.4.4 Einzelne Variable in eine HMI-Variablenliste importieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Verwendung

Die folgenden Objektmodell-Objekttypen können als untergeordnete Elemente einer HMI-Variable vorhanden sein und werden beim Import berücksichtigt:

MultilingualText	Für Kommentar, TagValue, DisplayName
TagArrayMemberTag	Für HMI-Array-Elemente
TagStructureMember	Für HMI-Strukturelemente
Event	Für konfigurierte Ereignisse
MultiplexEntry	Für konfigurierte Multiplex-Einträge von Variablen

Programmcode

Um eine HMI-Variable aus einer XML-Datei in eine HMI-Variablenliste zu importieren, ändern Sie den folgenden Programmcode:

```
//Imports a tag into a tag table
private static void ImportTagIntoTagTable(HmiTarget hmitarget)
{
    TagSystemFolder tagFolder = hmitarget.TagFolder;
    TagTable myTable = tagFolder.DefaultTagTable;
    TagComposition tagComposition = myTable.Tags;

    FileInfo info = new FileInfo(@"D:\Samples\Import\myExportedTag.xml");
    tagComposition.Import(info, ImportOptions.Override);
}
```

8.3.4.5 Besonderheiten beim Export/Import von HMI-Variablen

Einleitung

Besondere Überlegungen gelten beim Export und Import der folgenden HMI-Variablen:

- Externe HMI-Variablen mit integrierter Verbindung
- HMI-Variablen mit dem Datentyp "UDT"

Ähnliche Programmcodes

Der Programmcode für die oben genannten HMI-Variablen ist fast identisch mit den folgenden Programmcodes:

- Programmcode: HMI-Variablen exportieren (Seite 800)
- Programmcode: HMI-Variablen importieren (Seite 801)

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Besondere Überlegungen für den Export/Import einer externen HMI-Variable mit integrierter Verbindung

Beim Exportieren einer externen HMI-Variable mit integrierter HMI-Verbindung wird statt der PLC-Variablenlizenzen nur die Verknüpfung der HMI-Variable mit der PLC-Variable in der Exportdatei gespeichert.

Vor dem Import müssen Sie sicherstellen, dass der PLC, die entsprechenden PLC-Variablen und die integrierte Verbindung zum entsprechenden PLC im Projekt vorhanden sind. Wenn das nicht der Fall ist, müssen diese Elemente vor dem Import erzeugt werden. Beim anschließenden Import der externen HMI-Variable wird die Verknüpfung zur PLC-Variable wieder aktiviert.

Namen externer HMI-Variablen müssen über alle Variablenlisten eines Projekts eindeutig sein. Wenn Sie beim Import nicht die passende Variablenliste für die HMI-Variable angeben, wird der Import abgebrochen.

Verwenden Sie zum Importieren einer externen HMI-Variable mit integrierter Verbindung die folgende XML-Struktur:

```
<Hmi.Tag.Tag ID="1" CompositionName="Tags">
  <AttributeList>
    <Name>MyIntegratedHmiTag_1</Name>
  </AttributeList>
  <LinkList>
    <AcquisitionCycle TargetID="@OpenLink">
      <Name>1_s</Name>
    </AcquisitionCycle>
    <Connection TargetID="@OpenLink">
      <Name>HMI_Connection_MP277_300400</Name>      <- Must exist in the project
    </Connection>
    <ControllerTag TargetID="@OpenLink">
      <Name>Datablock_1.DBElement1</Name>          <- Must exist in the project
    </ControllerTag>
  </LinkList>
</Hmi.Tag.Tag>
```

Besondere Überlegungen für den Export/Import einer HMI-Variable des Datentyps "UDT"

Die Verknüpfung wird zum Datentyp exportiert, wenn eine HMI-Variable des Datentyps "UDT" exportiert wird. Nur versionierte Datentypen werden für den Import unterstützt.

Die Datentypen müssen in der Projektbibliothek gespeichert werden. Datentypen in der globalen Bibliothek werden nicht unterstützt.

Die folgenden Regeln gelten beim Import:

- Der referenzierte Datentyp muss in der Projektbibliothek enthalten sein.
Der Import wird beendet, wenn der Datentyp nicht in der Projektbibliothek enthalten ist.
- Der referenzierte Datentyp muss versioniert sein. Versionierung wird ab TIA Portal V13 SP1 unterstützt.
Wenn der Datentyp nicht versioniert ist, wird eine Ausnahme ausgelöst.

Hinweis

Der erste gefundene Datentyp wird beim Import zum Auflösen der Referenz verwendet.

Hierbei gilt Folgendes: Zuerst wird das Wurzelverzeichnis der Projektbibliothek durchsucht, danach die Unterordner.

Verwenden Sie die folgende XML-Struktur, um eine HMI-Variable des Datentyps "UDT" zu importieren:

```
<Hmi.Tag.Tag ID="1" CompositionName="Tags">
    ...
    <LinkList>
        <DataType TargetID="@OpenLink">
            <Name>HmiUdt_1 V 1.0.0</Name>      <- Must exist in the project library
        </DataType>
    ...
</LinkList>
...
</Hmi.Tag.Tag>
```

8.3.5 VB-Skripte

8.3.5.1 VB-Skripte exportieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Alle untergeordneten benutzerdefinierten Ordner werden beim Export berücksichtigt. Für jedes exportierte VB-Skript wird eine eigene XML-Datei erstellt.

Programmcode: VB-Skript exportieren

Um ein ausgewähltes VB-Skript eines HMI-Geräts in eine XML-Datei zu exportieren, ändern Sie den folgenden Programmcode:

```
//Exports a single vbscript of an HMI device
private static void ExportSingleVBScriptOfHMITarget(HmiTarget hmitarget)
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptComposition vbScripts = vbScriptFolder.VBScripts;
    VBScript vbScript = vbScripts.Find("MyVBScript");

    FileInfo info = new FileInfo(string.Format(@"C:\OpennessSamples\Export\Scripts
\{0}.xml", vbScript.Name));
    vbScript.Export(info, ExportOptions.None);
}
```

8.3.5.2 VB-Skripte aus einem Ordner exportieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Für jedes exportierte VB-Skript wird eine eigene XML-Datei erstellt.

Programmcode: VB-Skript aus einem benutzerdefinierten Ordner exportieren

Um ein VB-Skript aus einem benutzerdefinierten Ordner in eine XML-Datei zu exportieren, ändern Sie den folgenden Programmcode:

```
//Exports vbscripts of a selected vbscript system folder
private static void ExportVBScriptOfSelectedVBScriptSystemFolder(HmiTarget hmitarget)
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptUserFolderComposition vbUserFolders = vbScriptFolder.Folders;
    VBScriptUserFolder vbUserFolder = vbUserFolders.Find("MyVBUserFolder");
    VBScriptComposition vbScripts = vbUserFolder.VBScripts;

    foreach (VBScript script in vbScripts)
    {
        FileInfo info = new FileInfo(String.Format(@"C:\OpennessSamples\Export\Scripts
\{0}.xml", script.Name));
        script.Export(info, ExportOptions.None);
    }
}
```

Programmcode: Alle VB-Skripte aus einem Systemordner exportieren

Um alle VB-Skripte aus dem Systemordner zu exportieren, ändern Sie den folgenden Programmcode:

```
//Exports all vbscripts by using a foreach loop
private static void ExportAllVBScripts(HmiTarget hmitarget)
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptComposition vbScripts = vbScriptFolder.VBScripts;
    if (vbScripts == null) return;

    foreach (VBScript script in vbScripts)
    {
        FileInfo info = new FileInfo(string.Format(@"C:\OpennessSamples\Export\Scripts
\{0}.xml", script.Name));
        script.Export(info, ExportOptions.None);
    }
}
```

8.3.5.3 VB-Skripte importieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

8.3 Import/Export von Daten eines HMI-Geräts

Verwendung

Sammelimporte werden unterstützt. Als Alternative können Sie einen Programmcode mit einer Foreach-Schleife verwenden (VB-Skripte exportieren (Seite 803)).

Programmcode

Um ein VB-Skript aus einer XML-Datei in ein HMI-Gerät zu importieren, ändern Sie den folgenden Programmcode:

```
private static void ImportSingleVBScriptToHMITarget(HmiTarget hmitarget)
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptComposition vbScripts = vbScriptFolder.VBScripts;
    if (vbScripts == null) return;
    {
        FileInfo info = new FileInfo(@"D:\Samples\Import\VBScript.xml");
        vbScripts.Import(info, ImportOptions.None);
    }
}
```

8.3.6 Textlisten

8.3.6.1 Textlisten aus einem HMI-Gerät exportieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Der Export von Text- und Grafiklisten umfasst alle Einträge der Listen. Text- und Grafiklisten können getrennt exportiert werden.

Die Textlisten eines HMI-Geräts werden exportiert. Für jede exportierte Textliste wird eine eigene XML-Datei erstellt.

Programmcode

Ändern Sie den folgenden Programmcode, um Textlisten aus einem HMI-Gerät zu exportieren:

```
//Export TextLists
private static void ExportTextLists(HmiTarget hmitarget)
{
    TextListComposition text = hmitarget.TextLists;
    foreach (TextList textList in text)
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Export\{0}.xml",
textList.Name);
        textList.Export(info, ExportOptions.WithDefaults);
    }
}
```

8.3.6.2 Textliste in ein HMI-Gerät importieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Die API-Schnittstelle unterstützt den Import einer Textliste aus einer XML-Datei in ein HMI-Gerät.

Programmcode

Um eine Textliste aus einer XML-Datei in ein HMI-Gerät zu importieren, ändern Sie den folgenden Programmcode:

```
//Imports a single TextList
private static void ImportSingleTextList(HmiTarget hmitarget)
{
    TextListComposition textListComposition = hmitarget.TextLists;
    IList<TextList> importedTextLists = textListComposition.Import(new
FileInfo(@"D:\SamplesImport\myTextList.xml"), ImportOptions.Override);
}
```

8.3.6.3 Erweiterte XML-Formate für Export/Import von Textlisten

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Standardexport von Textlisten
Siehe Textlisten aus einem Bediengerät exportieren (Seite 806)
- Standardimport von Textlisten
Siehe Textlisten in ein Bediengerät exportieren (Seite 807)

Verwendung

Eine Textliste kann auch formatierte Texte enthalten. Das betrifft hauptsächlich die folgende Formatierung:

- Textformatierung
- Verweise auf andere Objekte im Text

Reine Textformatierung in einer zu exportierenden Textliste führt zu einem erweiterten XML-Exportformat. Objektreferenzen werden als Open Links charakterisiert. Gleiches gilt für zu importierende Textlisten mit formatierten Texten.

Erweiterte XML-Exportformate können auch erheblich komplexer werden. Beispielsweise kann mehr als nur der Objektname in der Textliste verknüpft sein, möglicherweise durch ein Open Link zu einer PLC-Variable eines anderen Geräts. In diesem Fall müssen alle Informationen in einem String codiert werden, um das Open Link zu entfernen.

```

<?xml version="1.0" encoding="utf-8"?>
<Document>
<!-- ... -->
<MultilingualText ID="5" CompositionName="Text">
    <ObjectList>
        <MultilingualTextItem ID="6" CompositionName="Items">
            <AttributeList>
                <Culture>en-US</Culture>
                <Text>
                    <body>
                        <p>
                            <field ref="0" />
                        </p>
                    </body>
                <fieldinfos>
                    <fieldinfo name="0" domaintype="HMICommonTextList">
                        <reference TargetID="@OpenLink">
                            <name>Siemens.Simatic.Hmi.Utah.TextAndGraphicLists.HmiTextList:Empty Text_li
                        </reference>
                        <subreference TargetID="@OpenLink">
                            <name>Siemens.Simatic.Hmi.Utah.Tag.HmiTag:t1</name>
                        </subreference>
                        <domaindata>
                            <format length="9" />
                        </domaindata>
                    </fieldinfo>
                </fieldinfos>
            </Text>
        </AttributeList>
    </MultilingualTextItem>
    <MultilingualTextItem ID="7" CompositionName="Items">
        <AttributeList>
            <Culture>de-CH</Culture>
            <Text>
                <body>
                    <p>
                        <field ref="0" />
                    </p>
                </body>
            <fieldinfos>
                <fieldinfo name="0" domaintype="HMICommonTextList">
                    <reference TargetID="@OpenLink">
                        <name>Siemens.Simatic.Hmi.Utah.TextAndGraphicLists.HmiTextList:Empty Text_li
                    </reference>
                    <subreference TargetID="@OpenLink">
                        <name>Siemens.Simatic.Hmi.Utah.Tag.HmiTag:t1</name>
                    </subreference>
                    <domaindata>
                        <format length="9" />
                    </domaindata>
                </fieldinfo>
            </fieldinfos>
        </Text>
    </AttributeList>
    </MultilingualTextItem>
    </ObjectList>
</MultilingualText>

```

8.3.7 Grafiklisten

8.3.7.1 Grafiklisten exportieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Der Export von Text- und Grafiklisten umfasst alle Einträge der Listen. Text- und Grafiklisten können getrennt exportiert werden.

Pro Grafikliste wird eine XML-Datei erstellt. In den Grafiklisten enthaltene globale Grafikobjekte werden als Open Links exportiert.

Programmcode

Um Grafiklisten aus einem Bediengerät zu exportieren, ändern Sie den folgenden Programmcode:

```
//Exports GraphicLists
private static void ExportGraphicLists(HmiTarget hmitarget)
{
    GraphicListComposition graphic = hmitarget.GraphicLists;
    foreach (GraphicList graphicList in graphic)
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Export\{0}.xml",
graphicList.Name));
        graphicList.Export(info, ExportOptions.WithDefaults);
    }
}
```

8.3.7.2 Grafikliste importieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Die API-Schnittstelle unterstützt den Import einer Grafikliste aus einer XML-Datei in ein HMI-Gerät.

Alle referenzierten Grafikobjekte der Grafikliste werden in den Import einbezogen. Referenzen auf globale Grafiken werden nicht einbezogen. Wenn die referenzierten globalen Grafiken im Zielprojekt vorhanden sind, werden die Referenzen auf die globalen Grafiken während des Imports wiederhergestellt.

Programmcode

Um eine Grafikliste aus einer XML-Datei in ein HMI-Gerät zu importieren, ändern Sie folgenden Programmcode:

```
//Imports a single GraphicList
private static void ImportSingleGraphicList(HmiTarget hmitarget)
{
    GraphicListComposition graphicListComposition = hmitarget.GraphicLists;
    IList<GraphicList> importedGraphicLists = graphicListComposition.Import(new
FileInfo(@"D:\Samples\Import\myGraphicList.xml"), ImportOptions.Override);
}
```

8.3.8 Verbindungen

8.3.8.1 Verbindungen exportieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Verwendung

Die API-Schnittstelle unterstützt den Export aller Verbindungen eines HMI-Geräts in eine XML-Datei.

Hinweis

Integrierte Verbindungen exportieren

Der Export integrierter Verbindungen wird nicht unterstützt.

Für jede exportierte Verbindung wird eine eigene XML-Datei erstellt.

8.3 Import/Export von Daten eines HMI-Geräts

Programmcode

Um alle Verbindungen eines HMI-Geräts in eine XML-Datei zu exportieren, ändern Sie folgenden Programmcode:

```
//Exports communication connections from an HMI device
private static void ExportConnectionsFromHMITarget(HmiTarget hmitarget)
{
    ConnectionComposition connections = hmitarget.Connections;
    foreach(Connection connection in connections)
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Export\{0}.xml",
connection.Name));
        connection.Export(info, ExportOptions.WithDefaults);
    }
}
```

8.3.8.2 Verbindungen importieren

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Verwendung

Die API-Schnittstelle unterstützt den Import aller Verbindungen eines HMI-Geräts aus einer XML-Datei in ein HMI-Gerät. Wenn Sie mehrere Kommunikationsverbindungen importieren möchten, importieren Sie die jeweilige XML-Datei für die entsprechende Verbindung.

Hinweis

Wenn Sie eine Verbindung in ein Projekt importieren, in dem bereits eine integrierte Verbindung konfiguriert ist, wird diese Verbindung nicht überschrieben. Der Import wird abgebrochen und eine Exception wird ausgelöst.

Programmcode

Um eine einzelne Verbindung eines HMI-Geräts aus einer XML-Datei in ein HMI-Gerät zu importieren, ändern Sie folgenden Programmcode:

```
//Imports Communication connections to an HMI device
private static void ImportConnectionsToHMITarget(HmiTarget hmitarget)
{
    ConnectionComposition connections = hmitarget.Connections;
    IList<Connection> importedConnectionLists = connections.Import(new
FileInfo(@"D:\Samples\Import\myConnectionImport.xml"), ImportOptions.Override);
}
```

8.3.9 Bilder

8.3.9.1 Übersicht der exportierbaren Bild-Objekte

Verwendung

Sie können die nachstehenden Bilder mit TIA Portal Openess APIs exportieren und importieren:

Tabelle 8-4 Unterstützte Bilder

Objekt	Export/Import möglich
Bild	Ja
Globales Bild	Ja
Bildvorlage	Ja
Permanentfenster	Ja
Pop-up-Bild	Ja
Slide-in-Bild	Ja

8.3 Import/Export von Daten eines HMI-Geräts

Sie können die nachstehenden Bildobjekte mit TIA Portal Openess APIs exportieren und importieren:

Tabelle 8-5 Unterstützte Bildobjekte

Bereich	Objekttyp	Export/Import möglich
Grundlegende Objekte	Linie	Ja
	Polygonzug	Ja
	Polygon	Ja
	Ellipse	Ja
	Ellipsensegment	–
	Kreissegment	–
	Ellipsenbogen	–
	Kreisbogen	–
	Kreis	Ja
	Rechteck	Ja
	Verbinder	–
	Textfeld	Ja
	Grafikanzeige	Ja
	Rohr	–
	Doppel-T-Stück	–
	T-Stück	–
	Rohrkrümmer	–

Bereich	Objekttyp	Export/Import möglich
Elemente	EA-Feld	Ja
	Grafisches EA-Feld	Ja
	Editierbares Textfeld	–
	Listenfeld	–
	Kombinationsfeld	–
	Schaltfläche	Ja
	Rundbutton	–
	Leuchtdrucktaster	Ja
	Schalter	Ja
	Symbolisches EA-Feld	Ja
	Datum/Uhrzeit-Feld	Ja
	Balken	Ja
	Symbolbibliothek	Ja
	Schieberegler	Ja
	Bildlaufleiste	–
	Kontrollkästchen	–
	Optionsschaltfläche	–
	Zeigerinstrument	Ja
	Uhr	Ja
	Speicherplatzanzeige	–
	Funktionstasten (Softkeys)	Ja
	Gruppen	Ja
	Bildbausteininstanzen	Ja

8.3 Import/Export von Daten eines HMI-Geräts

Bereich	Objekttyp	Export/Import möglich
Bedienelemente	Bildfenster	–
	Benutzeranzeige	Ja
	Druckauftrag/Skriptdiagnose	–
	Kamera-Anzeige	–
	PDF-Anzeige	–
	Rezepturanzelge	–
	Meldeanzeige	–
	Meldeindikator	–
	Meldefenster	–
	f(x)-Kurvenanzeige	–
	f(t)-Kurvenanzeige	–
	Tabellenanzeige	–
	Wertetabelle	–
	HTML-Browser	–
	Media Player	–
	Kanaldiagnose	–
	WLAN-Empfang	–
	Zonen-Name	–
	Zonen-Signal	–
	Wirkbereichs-Name	–
	Wirkbereichs-Name (RFID)	–
	Wirkbereichs-Signal	–
	Ladezustand	–
	Handrad	–
	Hilfe-Indikator	–
	Sm@rtClient-Anzeige	–
	Status/Steuern	–
	Speicherplatzanzeige	–
	Anzeige von NC-Unterprogrammen	–
	Systemdiagnoseanzeige	–
	System-Diagnosefenster	–

Siehe auch

Grundlagen zum Import/Export (Seite 773)

8.3.9.2 Alle Bilder eines HMI-Geräts exportieren

Voraussetzungen

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Um alle aggregierten Bilder aller benutzerdefinierten Ordner eines HMI-Geräts zu exportieren, ist ein anderer Programmcode erforderlich.

Programmcode: Alle Bilder eines Geräts exportieren

Um die Bilder eines benutzerdefinierten Bilderordners eines HMI-Geräts und den Bildersystemordner zu exportieren, ändern Sie folgenden Programmcode:

```
private static void ExportScreensOfDayce(string rootPath, HmiTarget hmitarget)
{
    DirectoryInfo info = new DirectoryInfo(rootPath);
    info.Create();
    //export the ScreenFolder recursive

    string screenPath = Path.Combine(rootPath, "Screens");
    info = new DirectoryInfo(screenPath);
    info.Create();
    ExportScreens(screenPath, hmitarget);
}
```

Programmcode: Alle Bilder eines benutzerdefinierten Ordners exportieren

Um die Bilder eines benutzerdefinierten Bilderordners eines HMI-Geräts und den Bildersystemordner zu exportieren, ändern Sie folgenden Programmcode:

```
private static void ExportScreensOfDayce(HmiTarget hmitarget)
{
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Find("MyScreenFolder");
    //or ScreenSystemFolder folder = hmitarget.ScreenFolder;
    ScreenComposition screens = folder.Screens;
    foreach(Screen screen in screens)
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\{0}\{1}.xml",
        folder.Name, screen.Name));
        screen.Export(info, ExportOptions.WithDefaults);
    }
}
```

Programmcode: Alle Bilder eines Geräts unabhängig vom Benutzer exportieren

Um alle Bilder zu exportieren, ändern Sie folgenden Programmcode:

```
public static void ExportScreens(string screenPath, HmiTarget target)
{
    foreach(Screen screen in target.ScreenFolder.Screens)
    {
        screen.Export(new FileInfo(Path.Combine(screenPath, screen.Name + ".xml")),
ExportOptions.WithDefaults);
    }
    foreach(ScreenUserFolder subfolder in target.ScreenFolder.Folders)
    {
        ExportScreenUserFolder(Path.Combine(screenPath, folder.Name), subfolder);
    }
}

private static void ExportScreenUserFolder(string screenPath, ScreenUserFolder folder )
{
    foreach(Screen screen in folder.Screens)
    {
        screen.Export(new FileInfo(Path.Combine(screenPath, screen.Name + ".xml")),
ExportOptions.WithDefaults);
    }
    foreach(ScreenUserFolder subfolder in folder.Folders)
    {
        ExportScreenUserFolder(Path.Combine(screenPath, subfolder.Name), subfolder);
    }
}
```

8.3.9.3 Bild aus einem Bildordner exportieren

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Die folgenden Daten eines Bilds werden exportiert:

Bild	Daten
Attribute	ActiveLayer, BackColor, Height, Width, Name, Number, HelpText
Links öffnen	Template
Zusammensetzungen	<ul style="list-style-type: none"> • Layers • Animations Alle auf Runtime Advanced basierenden konfigurierten Animationen werden exportiert. • Events Alle auf Runtime Advanced basierenden konfigurierten Ereignisse werden exportiert. • Softkeys Alle konfigurierten Softkeys werden exportiert.

Für jede Schicht werden die folgenden Daten exportiert:

Hinweis

Standardmäßig besteht der Schichtname im TIA Portal aus leerem Text.

Wenn Sie den Schichtnamen im TIA Portal nicht ändern, ist der Name der exportierten Schicht leer. In diesem Fall ist der angezeigte Schichtname im TIA Portal von der Sprache der Benutzeroberfläche abhängig.

Wenn Sie den Schichtnamen im TIA Portal ändern, wird der geänderte Schichtname in allen entsprechenden Sprachen angezeigt.

Schicht	Daten
Attribute	Name, Index, VisibleES
Zusammensetzungen	ScreenItems (mit Bildelementen)

Nicht einbezogen werden in den Export:

- SCADA-spezifische Attribute
- Schichten, die keine Bildelemente enthalten und deren Attribute sich nicht von den Standardwerten unterscheiden.

8.3 Import/Export von Daten eines HMI-Geräts

Programmcode

Um ein einzelnes Bild aus dem Benutzerordner oder aus dem Systemordner eines HMI-Geräts zu exportieren, ändern Sie folgenden Programmcode:

```
//Exports a single screen from a screen folder
private static void ExportSingleScreenFromScreenFolder(HmiTarget hmitarget)
{
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Find("MyScreenFolder");
    //or ScreenSystemFolder folder = hmitarget.ScreenFolder;
    ScreenComposition screens = folder.Screens;
    Screen screen = screens.Find("Screen_1.xml");
    if (screen == null) return;
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\{0}\{1}.xml",
        folder.Name, screen.Name));
        screen.Export(info, ExportOptions.WithDefaults);
    }
}
```

8.3.9.4 Bilder in ein HMI-Gerät importieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Die Bilder können nur in eine bestimmte Art von HMI-Gerät importiert werden. Das HMI-Gerät und das Gerät, aus dem die Bilder exportiert wurden, müssen den gleichen Gerätetyp besitzen.

Die folgenden Daten eines Bilds werden exportiert:

Bild	Daten
Attribute	ActiveLayer, BackColor, Height, Width, Name, Number, HelpText
Links öffnen	Templates
Zusammensetzungen	<ul style="list-style-type: none">• Layers• Animations Alle für Bilder konfigurierbaren Animationen werden importiert.• Events Alle für Ereignisse konfigurierbaren Animationen werden importiert.• Softkeys Alle für Softkeys konfigurierbaren Animationen werden importiert.

Für jede Schicht werden die folgenden Daten importiert:

Hinweis

Wenn Sie vor dem Import für den Schichtnamen leeren Text angegeben haben, ist der angezeigte Schichtname im TIA Portal nach dem Import von der Sprache der Benutzeroberfläche abhängig.

Wenn Sie einen Schichtnamen zugewiesen haben, wird der angegebene Schichtname nach dem Import in allen entsprechenden Sprachen angezeigt.

Schicht	Daten
Attribute	Name, Index
Zusammensetzung	ScreenItems

Einschränkungen

- Wenn die Breite und Höhe eines Bilds nicht den Abmessungen des Geräts entsprechen, wird der Importvorgang abgebrochen und eine Exception ausgelöst. Die Anpassung der enthaltenen Bildelemente wird nicht unterstützt. Aus diesem Grund können sich bestimmte Bildelemente außerhalb der Bildgrenzen befinden. In diesem Fall wird eine Compilerwarnung ausgegeben.
- Die Bildnummer muss für alle Bilder des Geräts jeweils eindeutig sein. Ein Bildimport wird abgebrochen, wenn ein Bild mit einer Bildnummer gefunden wird, die bereits im Gerät erstellt wurde. Wenn Sie noch keine Bildnummer zugewiesen haben, wird dem Bild während des Importvorgangs eine eindeutige Bildnummer zugeordnet.
- Die Anordnung der Bildelemente innerhalb der Z-Reihenfolge muss für jede Schicht im Bild eindeutig und lückenlos sein. Aus diesem Grund wird nach dem Import des Bilds eine Konsistenzprüfung durchgeführt, bei der die Anordnung gegebenenfalls repariert wird. Dieser Vorgang kann bei bestimmten Bildelementen zu geänderten „Tabindizes“ führen. Sie können die Z-Reihenfolge der Bildelemente in der XML-Datei manuell ändern. Das Bildelement an erster Stelle befindet sich ganz am Ende der Z-Reihenfolge.

Hinweis

Sie können die Werte für Breite und Höhe eines Bildelements in der XML-Datei ändern, wenn für das Bildelement das Attribut „Größe an Inhalt anpassen“ aktiviert ist.

Hinweis

Import von Bildtypen aus der Bibliothek wird nicht unterstützt

Ab WinCC V12 SP1 können Sie in der Bibliothek ein Bild als Typ erstellen. Instanzen des im Projekt verwendeten Bildtyps können wie andere Bilder mit der TIA Portal Openness-Anwendung bearbeitet werden. Wenn Sie Bilder exportieren, werden die Instanzen von Bildtypen ohne die Typinformationen exportiert.

Wenn Sie diese Bilder wieder in das Projekt importieren, werden die Instanzen des Bildtyps überschrieben und die Instanz wird vom Bildtyp abgelöst.

Programmcode: Bilder in ein HMI-Gerät importieren

Um mit der For each-Schleife Bilder in ein HMI-Gerät zu importieren, ändern Sie folgenden Programmcode:

```
//Imports all screens to an HMI device
private static void ImportScreensToHMITarget(HmiTarget hmitarget)
{
    FileInfo[] exportedScreens = new FileInfo[] {new FileInfo(@"D:\Samples\Import\Screen_1.xml"), new FileInfo(@"D:\Samples\Import\Screen_2.xml")};
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Find("MyScreenFolder");
    foreach (FileInfo screenFileInfo in exportedScreens)
    {
        folder.Screens.Import(screenFileInfo, ImportOptions.Override);
    }
}
```

Programmcode: In einen neu erstellten Benutzerordner importieren

Um ein Bild in einen neu erstellten Benutzerordner eines HMI-Geräts zu importieren, ändern Sie folgenden Programmcode:

```
//Imports a single screen to a new created user folder of an HMI device
private static void ImportSingleScreenToNewFolderOfHMITarget(HmiTarget hmitarget)
{
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Create("MyFolder");
    folder.Screens.Import(new FileInfo(@"D:\Samples\Import\myScreens.xml"),
ImportOptions.Override);
}
```

8.3.9.5 Permanentfenster exportieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Die folgenden Daten des Permanentfensters werden exportiert:

Permanentfenster	Daten
Attribute	ActiveLayer, BackColor, Height, Width, Name
Zusammensetzungen	Layers

Für jede Schicht werden die folgenden Daten exportiert:

Schicht	Daten
Attribute	Name, Index
Zusammensetzungen	ScreenItems (mit Bildelementen)

Programmcode

Um ein Permanentfenster eines HMI-Geräts in eine XML-Datei zu exportieren, ändern Sie folgenden Programmcode:

```
//Exports a permanent area
private static void ExportScreenoverview(HmiTarget hmitarget)
{
    ScreenOverview overview = hmitarget.ScreenOverview;
    if (overview == null) return;

    FileInfo info = new FileInfo(@"D:\Samples\Screens\ExportedOverview.xml");
    overview.Export(info, ExportOptions.WithDefaults);
}
```

8.3.9.6 Permanentfenster importieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

8.3 Import/Export von Daten eines HMI-Geräts

Verwendung

Die folgenden Daten des Permanentfensters werden importiert:

Permanentfenster	Daten
Attribute	ActiveLayer, BackColor, Height, Width, Name, Visible, Number
Zusammensetzungen	Layers

Für jede Schicht werden die folgenden Daten importiert:

Schicht	Daten
Attribute	Name, Index
Zusammensetzungen	ScreenItems (mit Bildelementen)

Wenn die Breite und Höhe eines Bilds nicht den Abmessungen des Geräts entsprechen, wird der Importvorgang abgebrochen und eine Exception ausgelöst. Die Anpassung der enthaltenen Geräteelemente (Bildelemente) wird nicht unterstützt. Aus diesem Grund können sich einige Geräteelemente außerhalb der Bildgrenzen befinden. In diesem Fall wird eine Compilerwarnung ausgegeben.

Die Anordnung der Geräteelemente im Permanentfenster muss eindeutig und lückenlos sein. Aus diesem Grund wird nach dem Import des Permanentfensters eine Konsistenzprüfung durchgeführt, bei der die Anordnung gegebenenfalls repariert wird. Dieser Vorgang kann bei bestimmten Geräteelementen zu geänderten „Tabindizes“ führen.

Programmcode

Um ein Permanentfenster aus einer XML-Datei in ein HMI-Gerät zu importieren, ändern Sie folgenden Programmcode:

```
//Imports a permanent area
private static void ImportScreenOverview(HmiTarget hmiTarget)
{
    FileInfo info = new FileInfo(@"D:\Samples\Screens\ExportedOverview.xml");
    hmiTarget.ImportScreenOverview(info, ImportOptions.Override);
}
```

8.3.9.7 Alle Bildvorlagen eines HMI-Geräts exportieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Pro Bildvorlage wird eine XML-Datei erstellt.

Da Sammelexporte nicht unterstützt werden, müssen Sie alle Bildvorlagen enumerieren und separat exportieren. Achten Sie dabei darauf, dass die verwendeten Bildvorlagennamen den Dateibenennungskonventionen Ihres Dateisystems entsprechen.

Programmcode: Alle Bildvorlagen eines Geräts exportieren

Um alle Bildvorlagen aus einem bestimmten Ordner zu exportieren, ändern Sie folgenden Programmcode:

```
public static void ExportScreenTemplatesOfDevice(string rootPath ,
ScreenTemplateUserFolder folder)
{
    string screenPath = Path.Combine(rootPath, "Screens");
    DirectoryInfo info = new DirectoryInfo(screenPath);
    info.Create();

    //export the ScreenTemplateFolder recursive
    ExportScreenTemplates (screenPath, hmitarget);
}
```

Programmcode: Alle Bildvorlagen eines bestimmten Ordners exportieren

Um alle Bildvorlagen zu exportieren, ändern Sie folgenden Programmcode:

```
//Exports all screen templates of a selected folder
private static void ExportScreenTemplates(string templatePath, HmiTarget hmitarget)
{
    foreach (ScreenTemplate screen in hmitarget.ScreenTemplateFolder.ScreenTemplates)
    {
        screen.Export(new FileInfo(Path.Combine(templatePath, screen.Name + ".xml")),
ExportOptions.WithDefaults);
    }
    foreach (ScreenTemplateUserFolder folder in hmitarget.ScreenTemplateFolder.Folders)
    {
        ExportScreenTemplates(Path.Combine(templatePath, folder.Name), hmitarget);
    }
}
```

8.3.9.8 Bildvorlagen aus einem Ordner exportieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

8.3 Import/Export von Daten eines HMI-Geräts

Verwendung

Die folgenden Daten der Bildvorlage werden exportiert:

Bildvorlagen	Daten
Attribute	ActiveLayer, BackColor, Height, Width, Name
Zusammensetzungen	<ul style="list-style-type: none">• Layers• Animations Alle konfigurierten Animationen werden exportiert. SCADA-Animationen werden nicht exportiert.• Softkeys Alle konfigurierten Softkeys werden exportiert.

Für jede Schicht werden die folgenden Daten exportiert:

Schicht	Daten
Attribute	Name, Index
Zusammensetzungen	ScreenItems (mit Bildelementen)

Programmcode: Bildvorlage eines benutzerdefinierten Ordners exportieren

Um eine einzelne Bildvorlage aus dem Systemordner oder einem benutzerdefinierten Ordner zu exportieren, ändern Sie folgenden Programmcode:

```
private static void ExportSingleScreenTemplate(string templatePath, HmiTarget hmiTarget)
{
    ScreenTemplateUserFolder folder =
hmiTarget.ScreenTemplateFolder.Folders.Find("MyTemplateFolder");
    //or ScreenTemplateSystemFolder folder = hmiTarget.ScreenTemplateFolder;
    ScreenTemplateComposition templates = folder.ScreenTemplates;
    ScreenTemplate template = templates.Find("templateName");
    if(template == null) return;

    FileInfo info = new FileInfo(string.Format(@"D:\Samples\Templates\{0}\{1}.xml",
folder.Name, template.Name));
    template.Export(info, ExportOptions.WithDefaults);
}
```

Programmcode: Alle Bildvorlagen eines benutzerdefinierten Ordners exportieren

Um alle Bildvorlagen aus einem bestimmten Ordner zu exportieren, ändern Sie folgenden Programmcode:

```
public static void ExportScreenTemplateUserFolder(string rootPath,
ScreenTemplateUserFolder folder)
{
    DirectoryInfo info = new DirectoryInfo(rootPath);
    info.Create();

    foreach (ScreenTemplate screen in folder.ScreenTemplates)
    {
        screen.Export(new FileInfo(Path.Combine(info.FullName, screen.Name + ".xml")), 
ExportOptions.WithDefaults);
    }
    foreach (ScreenTemplateUserFolder subfolder in folder.Folders)
    {
        ExportScreenTemplateUserFolder(Path.Combine(info.FullName, subfolder.Name), 
subfolder);
    }
}
```

Programmcode: Alle Bildvorlagen eines bestimmten Ordners exportieren

Um alle Bildvorlagen zu exportieren, ändern Sie folgenden Programmcode:

```
//Exports all screen templates of a selected folder
private static void ExportScreenTemplates(string templatePath, ScreenTemplateUserFolder
folder)
{
    foreach (ScreenTemplate screen in folder.ScreenTemplates)
    {
        screen.Export(new FileInfo(Path.Combine(templatePath, screen.Name + ".xml")), 
ExportOptions.WithDefaults);
    }
    foreach (ScreenTemplateUserFolder subfolder in folders.Folders)
    {
        ExportScreenTemplates(Path.Combine(templatePath, subfolder.Name), subfolder);
    }
}
```

8.3.9.9 Bildvorlagen importieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Die folgenden Daten einer Bildvorlage werden importiert:

Bildvorlage	Daten
Attribute	ActiveLayer, BackColor, Height, Width, Name, SetTabOrderInFront
Zusammensetzungen	<ul style="list-style-type: none">• Layers• Animations Alle für Bilder konfigurierbaren Animationen werden importiert.• Softkeys Alle für Softkeys konfigurierbaren Animationen werden importiert.

Für jede Schicht werden die folgenden Daten importiert:

Schicht	Daten
Attribute	Name, Index
Zusammensetzungen	ScreenItems (mit Bildelementen)

Wenn die Breite und Höhe einer Bildvorlage nicht den Abmessungen des Geräts entsprechen, wird der Importvorgang abgebrochen und eine Exception ausgelöst. Die Anpassung der enthaltenen Bildelemente wird nicht unterstützt. Aus diesem Grund können sich bestimmte Bildelemente außerhalb der Bildgrenzen befinden. In diesem Fall wird eine Compilerwarnung ausgegeben.

Die Anordnung der Geräteelemente in der Bildvorlage muss eindeutig und lückenlos sein. Aus diesem Grund wird nach dem Import der Bildvorlage eine Konsistenzprüfung durchgeführt, bei der die Anordnung gegebenenfalls repariert wird. Dieser Vorgang kann bei bestimmten Bildelementen zu geänderten „Tabindizes“ führen.

Programmcode: Allgemeiner Import

Um mit der For each-Schleife alle Bildvorlagen in ein HMI-Gerät zu importieren, ändern Sie folgenden Programmcode:

```
//Imports screen templates to an HMI device
private static void ImportScreenTemplatesToHMITarget(HmiTarget hmitarget)
{
    ScreenTemplateUserFolder folder =
hmitarget.ScreenTemplateFolder.Folders.Find("MyTemplateFolder");
    // or ScreenTemplateSystemFolder folder = hmitarget.ScreenTemplateFolder;
    FileInfo[] exportedTemplates = {new FileInfo[] { new FileInfo(@"D:\Samples\Import
\Template_1.xml"), new FileInfo(@"D:\Samples\Import\Template_n.xml") };};
    foreach (FileInfo templateFileName in exportedTemplates)
    {
        folder.ScreenTemplates.Import(templateFileName, ImportOptions.Override);
    }
}
```

Programmcode: In einen neu erstellten Benutzerordner importieren

Um eine Bildvorlage in einen neu erstellten Benutzerordner eines HMI-Geräts zu importieren, ändern Sie folgenden Programmcode:

```
//Imports screen templates to a user folder of an HMI device
private static void ImportScreenTemplatesToFolderOfHMITarget(HmiTarget hmitarget)
{
    ScreenTemplateUserFolder screenTemplateFolder =
        hmitarget.ScreenTemplateFolder.Folders.Find("MyTemplateFolder");
    ScreenTemplateUserFolder folder = screenTemplateFolder.Folders.Create("MyNewFolder");
    folder.ScreenTemplates.Import(new FileInfo(@"D:\Samples\Import\ScreenTemplate.xml"),
ImportOptions.Override);
}
```

8.3.9.10 Exportieren eines Pop-up-Bilds

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Die folgenden Daten des Pop-Up-Bilds werden exportiert:

Bildvorlagen	Daten
Attribute	ActiveLayer, BackColor, GridColor, Height, Name, ScrollbarBackgroundColor, ScrollbarForegroundColor, Width
Zusammensetzungen	<ul style="list-style-type: none"> Layers Events Alle konfigurierten Ereignisse werden exportiert.

Für jede Schicht werden die folgenden Daten exportiert:

Schicht	Daten
Attribute	Name, Index, VisibleES
Zusammensetzungen	ScreenItems Alle exportierbaren Bildobjekte werden exportiert.

Programmcode: Exportieren eines Pop-up-Bilds aus einem Ordner

Um ein einzelnes Pop-up-Bild aus dem Systemordner oder einem benutzerdefinierten Ordner zu exportieren, ändern Sie folgenden Programmcode:

```
//Exports a single pop-up screen
private static void ExportSinglePopUpScreen(HmiTarget hmitarget)
{
    ScreenPopupUserFolder folder =
hmitarget.ScreenPopupFolder.Folders.Find("MyPopupFolder");
//or ScreenPopupSystemFolder folder = hmitarget.ScreenPopupFolder;
ScreenPopupComposition popups = folder.ScreenPopups;
ScreenPopup popup = popups.Find("popupName");
if(popup == null) return;

FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\{0}\{1}.xml",
folder.Name, popup.Name));
popup.Export(info, ExportOptions.WithDefaults);
}
```

8.3.9.11 Importieren eines Pop-up-Bildes

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Die folgenden Daten eines Pop-up-Bilds werden importiert:

Bildvorlagen	Daten
Attribute	ActiveLayer, BackColor, GridColor, Height, Name, ScrollbarBackgroundColor, ScrollbarForegroundColor, Width
Zusammensetzungen	<ul style="list-style-type: none">• Layers• Events Alle konfigurierten Ereignisse werden exportiert.

Die Existenz folgender Attribute ist für das Importieren verpflichtend:

- Name
- Height
- Width

Für jede Schicht werden die folgenden Daten importiert:

Schicht	Daten
Attribute	Name, Index, VisibleES
Zusammensetzungen	ScreenItems Alle importierbaren Bildobjekte werden importiert.

Einschränkungen

Wenn das Gerät keine Pop-up-Bilder unterstützt, wird der Kopiervorgang abgebrochen und eine Ausnahme ausgelöst.

Wenn die Breite und Höhe eines Pop-up-Bilds nicht den Abmessungen des Geräts entsprechen, wird der Importvorgang abgebrochen und eine Exception ausgelöst.

- Minimale Höhe = 1 Pixel
- Minimale Breite = 1 Pixel
- Max. Höhe = sechsfache Höhe des Gerätebildschirms
- Max. Breite = zweifache Breite des Gerätebildschirms
- Bei Geräten mit Runtime-Version V13 SP1 entspricht die maximale Höhe und die maximale Breite der Höhe und Breite des Gerätebildschirms.

Programmcode: Importieren eines Pop-up-Bilds in einen Ordner

Um ein Pop-up-Bild in einen Pop-up-Bild-Systemordner oder in einen benutzerdefinierten Ordner zu importieren, ändern Sie den folgenden Programmcode:

```
//Imports a pop-up screen to an HMI device
private static void ImportPopupScreenToHMITarget(HmiTarget hmitarget)
{
    FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\PopupScreen.xml"));
    hmitarget.ScreenPopupFolder.ScreenPopups.Import(info, ImportOptions.None);
}
```

8.3.9.12 Exportieren eines Slide-in-Bilds

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Die folgenden Daten und Werte des Slide-in-Bilds werden exportiert:

Bildvorlagen	Daten	
Attribute	Activate	false
	ActiveLayer	0
	BackColor	(182; 182; 182)
	GridColor	(0; 0; 0)
	Dimension	427 Das Attribut "Dimension" gibt entweder die Breite oder die Höhe des Slide-in-Bilds an, abhängig von der Art des Slide-in-Bilds.
	LineColor1	(223; 223; 223)
	LineColor2	(32; 32; 32)
	OperableAreaColor	(128; 128; 128)
	SlideinType	Oben, Unten, Links, Rechts Slide-in-Bilder haben keinen Namen, sondern einen SlideinType.
	Visibility	FadeOut
Zusammensetzungen	Layers	

Hinweis

Slide-in-Bilder haben keinen Namen, sondern einen SlideinType.

Für jede Schicht werden die folgenden Daten exportiert:

Schicht	Daten	
Attribute	Name,	
	Index	
	VisibleES	
Zusammensetzungen	ScreenItems	Alle exportierbaren Bildobjekte werden exportiert.

Programmcode: Exportieren eines Slide-in-Bilds

Um ein einzelnes Slide-in-Bild aus dem Systemordner zu exportieren, ändern Sie folgenden Programmcode:

```
//Exports a single slide-in screen
private static void ExportSingleSlideinScreen(HmiTarget hmitarget)
{
    ScreenSlideinSystemFolder systemFolder = hmitarget.ScreenSlideinFolder;
    var screens = systemFolder.ScreenSlideins;
    ScreenSlidein slidein = screens.Find(SlideinType.Bottom);
    if (slidein == null) return;

    FileInfo info = new FileInfo(string.Format(@"D:\Samples\Screens\{0}\{1}.xml"));
    slidein.Export(info, ExportOptions.WithDefaults);
}
```

8.3.9.13 Importieren eines Slide-in-Bilds

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Die folgenden Daten und Werte eines Slide-in-Bilds werden importiert:

Bildvorlagen	Daten
Attribute	Activate = false ActiveLayer = 0 Authorization BackColor = (182; 182; 182) Dimension = 427 Das Attribut „Dimension“ gibt entweder die Breite oder die Höhe des Slide-in-Bilds an, abhängig davon, welches der zwei Attribute für die Art des Slide-in-Bilds geändert werden kann. GridColor = (0; 0; 0) LineColor1 = (223; 223; 223) LineColor2 = (32; 32; 32) OperateableAreaColor = (128; 128; 128) SlideinType = Top, Bottom, Left, Right Visibility = FadeOut
Zusammensetzungen	Layers

8.3 Import/Export von Daten eines HMI-Geräts

Die Existenz des folgenden Attributs ist für das Importieren verpflichtend:

- SlideinType

Für jede Schicht werden die folgenden Daten importiert:

Schicht	Daten
Attribute	Name, Index, VisibleES
Zusammensetzungen	ScreenItems Alle importierbaren Bildobjekte werden importiert.

Einschränkungen

- Wenn das Gerät keine Slide-in-Bilder unterstützt, wird der Import abgebrochen und eine Ausnahme ausgelöst.
 - Wird ein Slide-in-Bild von einem anderen Element referenziert, ist das Slide-in-Bild über openlink zu referenzieren und nicht über SlideinType, z. B. in der Systemfunktion „ShowSlideinScreen“).
- Die folgende Tabelle zeigt die Abbildung des Attributs "SlideinType" mit dem entsprechenden openlink:

SlideinType	Openlink-Name
Top	GraphX_Slidein_Top
Right	GraphX_Slidein_Right
Bottom	GraphX_Slidein_Bottom
Left	GraphX_Slidein_Left

Programmcode: Importieren eines Slide-in-Bilds in einen Ordner

Um ein Slide-in-Bild in einen Slide-in-Bild-Systemordner zu importieren, ändern Sie folgenden Programmcode:

```
//Imports a slide-in screen to an HMI device
private static void ImportSlideinScreenToHMITarget(HmiTarget hmitarget)
{
    FileInfo info = new FileInfo(@"D:\Samples\Screens\SlideInScreen.xml");
    hmitarget.ScreenSlideinFolder.ScreenSlideins.Import(info, ImportOptions.None);
}
```

8.3.9.14 Bild mit Eingabemaske exportieren

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Die folgenden Daten einer Bildbausteininstanz in einem Bild werden exportiert:

Bild	Daten
Attribute	Left, Top, Width, Height, ObjectName, Resizing,TabIndex,FaceplateTypeName
Schnittstellenattribute	Alle konfigurierten Schnittstellenattribute einer Bildbausteininstanz werden als exportierbare Bildelemente exportiert.
Zusammensetzungen	<ul style="list-style-type: none"> • Animationen Alle bewegten Animationen werden exportiert. Variablenanimationen basieren auf den Attributen der Schnittstelle. • Ereignisse Alle konfigurierten Ereignisse werden exportiert.

Beachten Sie die folgenden Vorgaben für exportierte Attribute von Bildbausteininstanzen:

- Resizing
Das Attribut „Resizing“ wird in jedem Fall exportiert, unabhängig von den Exportoptionen.
- FaceplateTypeName
Das Attribut "FaceplateTypeName" identifiziert den entsprechenden Typ und die Version des Bildbausteins, z. B. „Faceplate_1 V 0.0.2“.

Hinweis

Bildbausteintyp in einem Bibliotheksordner

Befindet sich ein Bildbausteintyp in einem Bibliotheksordner, benötigen Sie den gesamten Pfad und Namen zur Identifizierung des Bildbausteintyps. Das Schlüsselwort „@\$@“ wird verwendet, um Ordner und/oder Namen von Bildbausteintypen zu trennen, z. B. „Folder_1@\$@SubFolder_1@\$@Faceplate_1 V 0.0.2“.

Die folgenden Daten von Bildelementen innerhalb einer Bildbausteininstanz sind vom Export ausgeschlossen:

Bildelement	Attribut
E/A-Feld	Flashing on limit violation
Grafisches E/A-Feld	Fit embedded graphic object to screen size

Programmcode

Um ein einzelnes Bild mit einer Bildbausteininstanz zu exportieren, ändern Sie den folgenden Programmcode:

```
//Exports a single screen including a faceplate instance
private static void ExportSingleScreenWithFaceplateInstance(HmiTarget hmitarget)
{
    ScreenFolder folder = hmitarget.ScreenFolder.Folders.Find("MyScreenFolder");
    ScreenComposition screens = folder.Screens;
    Screen screen = screens.Find("ScreenWithFaceplateName");
    if (screen == null) return;
    {
        FileInfo info = new FileInfo(string.Format(@"D:\Samples\Faceplates\{0}\{1}.xml",
        folder.Name, screen.Name));
        screen.Export(info, ExportOptions.WithDefaults);
    }
}
```

8.3.9.15 Bild mit Eingabemaske importieren

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Die folgenden Daten einer Bildbausteininstanz in einem Bild werden importiert:

Bild	Daten
Attribute	Left, Top, Width, Height, ObjectName, Resizing, TabIndex, FaceplateTypeName
Schnittstellenattribute	Alle konfigurierten Schnittstellenattribute einer Bildbausteininstanz werden als importierbare Bildelemente importiert.
Zusammensetzungen	<ul style="list-style-type: none">• Animationen Alle bewegten Animationen werden importiert. Variablenanimationen basieren auf den Attributen der Schnittstelle.• Ereignisse Alle konfigurierten Ereignisse werden importiert.

Die Existenz folgender Attribute ist für das Importieren verpflichtend:

- ObjectName
- FaceplateTypeName

Die folgenden Daten von Bildelementen innerhalb einer Bildbausteininstanz sind von Export und Import ausgeschlossen:

Bildelement	Attribut
E/A-Feld	Flashing on limit violation
Grafisches E/A-Feld	Fit embedded graphic object to screen size

Einschränkungen

- Bildbaustein, Ereignis oder Schnittstellenattribut unbekannt
Wenn ein faceplate type name, ein event name oder ein interface attribute name in der Importdatei angegeben ist, der nicht im Projekt existiert, wird der Import abgebrochen und eine Ausnahme ausgegeben.
- Skalierungsverhalten einer Bildbausteininstanz
Das Attribut „Resizing“ wird in jedem Fall importiert, unabhängig von den Exportoptionen.
Beispiele:
Ist "Resizing" auf "KeepRatio" gesetzt, dann wird das Attribut "Height" verwendet, um den Attributwert "Width" zu berechnen.
 - Die Größe eines Bildbausteintyps beträgt 100 x 100 Pixel. Wenn eine Bildbausteininstanz mit einer Größe von 300 x 100 Pixel importiert wird und der Wert "FixedSize" auf das Attribut "Resizing" eingestellt ist, ist der Import erfolgreich und die Bildbausteingröße wird auf 100 x 100 Pixel gesetzt.
 - Die Größe eines Bildbausteintyps beträgt 100 x 50 Pixel. Wenn eine Bildbausteininstanz mit einer Größe von 100 x 100 Pixel importiert wird und der Wert "KeepRatio" auf das Attribut "Resizing" eingestellt ist, ist der Import erfolgreich und die Bildbausteingröße wird auf 200 x 100 Pixel gesetzt.

Hinweis

Skalierungsverhalten von importierten Bildbausteininstanzen

Die Werte von „Resizing“ und die Werte der Schnittstellenattribute können die Größe der importierten Bildbausteininstanz und sogar die Größe der darin enthaltenen Bildelemente beeinflussen.

Um ungewünschte Änderungen am Aussehen einer Bildbausteininstanz zu vermeiden, importieren Sie einen Bildbaustein in der ursprünglichen Größe oder ohne die Attributwerte "Width" und "Height".

- Abweichende Schnittstellenattributwerte
 - Wenn Sie die Attribute für den Import ändern, wird der letzte verwendete Schnittstellenattributwert importiert.
 - Wenn die Attribute voneinander abhängig sind, können andere Attribute während des Imports geändert werden.
Beispiel: Ein Bildbaustein enthält ein E/A-Feld. Das Attribut „Betriebsart“ wird mit einem Schnittstellenattribut verbunden. Wenn Sie erst die Betriebsart auf „Ausgang“ und dann das Attribut „Versteckter Eingang“ auf wahr setzen, wird der Wert von „Versteckter Eingang“ nach dem Import nicht verwendet. Das Attribut „Versteckter Eingang“ wurde bei der ersten Änderung auf schreibgeschützt gesetzt und der Wert kann nicht verwendet werden.
 - Wenn ein Attributwert die Einschränkungen von WinCC nicht erfüllt, wird der Wert des Bildbausteintyps angezeigt.
Beispiel: Der Anzeigebereich eines Zeigerinstrument wird auf 10 - 80 eingestellt. Die Attribute „Maximalwert“ und „Minimalwert“ werden als Schnittstellenattribute konfiguriert. Wenn Sie einen Minimalwert einstellen, der den Maximalwert übersteigt, z. B. 100, dann wird der Wert des Bildbausteintyps für „Minimalwert“ nach dem Import angezeigt.

- Wenn ein Schnittstellenattribut mit mehreren Attributen von Bildelementen innerhalb des Bildbausteintyps verbunden ist, dann zeigt das Schnittstellenattribut an der Bildbausteininstanz den verwendeten Attributwert des ersten verbundenen Bildelements an.
Beispiel: Ein Bildbaustein enthält zwei Zeigerinstrumente mit abweichenden Maximalwerten. Die Minimalwerte beider Zeigerinstrumente werden mit einem einzelnen Schnittstellenattribut verbunden.
Wenn Sie erst einen Minimalwert einstellen, der für beide Zeigerinstrumente anwendbar ist, werden beide Werte eingestellt.
Wenn Sie dann einen Wert einstellen, der nur für das zweite Zeigerinstrument anwendbar ist, wird der Wert nur für das zweite Zeigerinstrument eingestellt, der Wert des ersten Zeigerinstruments wird hingegen als Schnittstellenattribut angezeigt.

Programmcode: Bilder mit einer Bildbausteininstanz importieren

Um ein Bild mit einer Bildbausteininstanz zu importieren, ändern Sie den folgenden Programmcode:

```
//Imports single screen including a faceplate instance
private static void ImportSingleScreenWithFaceplateInstance(HmiTarget hmitarget)
{
    FileInfo info = new FileInfo(@"D:\Samples\Screens\ScreenFaceplate.xml");
    hmitarget.ScreenFolder.Screens.Import(info, ImportOptions.None);
}
```

8.4 Beobachtungs- und Forcetabelle exportieren/importieren

Voraussetzung:

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Sie haben mit Ihrer Anwendung TIA Portal Openness ein Projekt geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Mit TIA Portal Openness können Sie die Beobachtungstabelle und die Forcetabelle aus dem TIA Portal in SIMATIC ML exportieren und die Beobachtungstabelle und Forcetabelle aus SIMATIC ML importieren.

8.4 Beobachtungs- und Forcetabelle exportieren/importieren

Programmcode: Beobachtungstabelle und Forcetabelle exportieren

Sie können den folgenden Programmcode ändern, um die Beobachtungstabelle zu exportieren:

```
SoftwareContainer softwareContainer =
((IEngineeringServiceProvider)item).GetService<SoftwareContainer>();
PlcSoftware plcSoftware = softwareContainer.Software as PlcSoftware;
PlcWatchTableComposition exportWatchTables =
plcSoftware.PlcWatchAndForceTableGroup.WatchTables;
PlcWatchTable watchTable = exportWatchTables.Find(watchTableName);
if(watchTable != null)
{
    watchTable.Export((FileInfo) fileInfo, ExportOptions.None);
}
```

Hinweis

In der Beobachtungstabelle müssen Exportoptionen eingestellt werden (None, WithDefaults, WithReadOnly, WithDefaultsAndReadOnly). Eine Beobachtungstabelle hat nur eine veröffentlichte Eigenschaft, den Namen. Der ist zum Lesen veröffentlicht.

Sie können den folgenden Programmcode ändern, um die Forcetabelle zu exportieren:

```
SoftwareContainer softwareContainer =
((IEngineeringServiceProvider)item).GetService<SoftwareContainer>();
PlcSoftware plcSoftware = softwareContainer.Software as PlcSoftware;
PlcForceTableComposition exportForceTables =
plcSoftware.PlcWatchAndForceTableGroup.ForceTables;
PlcForceTable forceTable = exportForceTables[0];
forceTable.Export((FileInfo) fileInfo, ExportOptions.None);
```

Hinweis

In jeder Situation gibt es nur eine Forcetabelle, deren Name schreibgeschützt ist.

Programmcode: Beobachtungstabelle und Forcetabelle importieren

Sie können den folgenden Programmcode ändern, um die Beobachtungstabelle zu importieren:

```
SoftwareContainer softwareContainer =
((IEngineeringServiceProvider)item).GetService<SoftwareContainer>();
PlcSoftware plcSoftware = softwareContainer.Software as PlcSoftware;
PlcWatchTableComposition importWatchTables =
plcSoftware.PlcWatchAndForceTableGroup.WatchTables;
IList<PlcWatchTable> WatchTables = importWatchTables.Import((FileInfo) fileInfo,
ImportOptions.None);
```

Hinweis

Die importierte Beobachtungstabelle wird der Liste der Beobachtungstabellen hinzugefügt. In der Beobachtungstabelle müssen die erforderlichen Importoptionen eingestellt werden (None, Override).

Forcetabellen können auf ähnliche Weise importiert werden, es ist jedoch nur genau eine Forcetabelle zulässig. Wenn Sie statt Override die Option None verwenden und bereits eine (nicht leere) Forcetabelle vorhanden ist, schlägt der Import mit der folgenden Ausnahme RecoverableException: fehl: Es darf nur eine Forcetabelle importiert werden. Verwenden Sie die Importoption Override, um die vorhandene Tabelle zu überschreiben.

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

8.5 Daten eines PLC-Geräts importieren/exportieren

8.5.1 Bausteine

8.5.1.1 XML-Struktur des Abschnitts Bausteinschnittstelle

Grundprinzip

Die Daten in der Exportdatei vom Import/Export sind mit Verweis auf eine Grundstruktur gegliedert. Jede Importdatei muss die grundlegenden strukturellen Bedingungen erfüllen.

Die Exportdatei enthält alle bearbeiteten Variablen und Konstanten des Schnittstellenabschnitts eines exportierten Bausteins. Alle Attribute mit "ReadOnly = "TRUE"" und "Informative = "TRUE"" werden ausgeschlossen.

Wenn die Information redundant ist, muss sie in der Import-XML-Datei und der Projektdatei identisch sein. Andernfalls wird beim Import eine wiederherstellbare Ausnahme ausgelöst.

Die Projektdaten enthalten mehr Daten als die Import-XML-Datei, z. B. kann ein externer Typ zusätzliche Mitglieder haben.

Nur schreibbare Werte können über TIA Portal Openness XML importiert werden.

Abhängig von den Exporteinstellungen in TIA Portal Openness enthält die Exportdatei einen festgelegten Satz von Attributen und Elementen. Die aus höheren Versionen des Produkts exportierte XML ist beim Importvorgang in niedrigeren Versionen des TIA Portals nicht kompatibel.

Grundstruktur

Der Schnittstellenabschnitt eines exportierten Bausteins ist im Element <Interface> in der SimaticML eines Bausteins enthalten. Das Stammobjekt ist das Element <Sections>, das den Schnittstellenabschnitt eines exportierten Bausteins darstellt. Die Reihenfolge der folgenden Beschreibung von Elementen stellt die erforderliche Reihenfolge in der Eingabedatei dar.

```
<Interface>
  <Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v1">
    <Section Name="Input">
      <Member Name="input1" Datatype="Bool" Remanence="Volatile" Accessibility="Public">
        <AttributeList>
          ...
        </AttributeList>
      </Member>
      <Member Name="input2" Datatype="Bool" Remanence="Volatile" Accessibility="Public">
        <AttributeList>
          ...
        </AttributeList>
      </Member>
    </Section>
    <Section Name="Output">
      <Member Name="output1" Datatype="Bool" Remanence="Volatile" Accessibility="Public">
        <AttributeList>
          ...
        </AttributeList>
      </Member>
    </Section>
    <Section Name="InOut" />
    <Section Name="Static" />
    <Section Name="Temp" />
    <Section Name="Constant" />
  </Sections>
</Interface>
```

- Abschnitt
Abschnitt stellt einen einzelnen Parameter oder Lokaldaten eines Programmbausteins dar.
- Mitglied
Mitglied stellt die im Programmstein verwendeten Variablen oder Konstanten dar.
Abhängig vom Datentyp einer Variablen können Mitglieder verschachtelt sein oder weitere strukturelle Unterelemente haben.
Beim Datentyp "ARRAY" stellt das strukturelle Element "Subelement Path" z. B. den Index von Komponenten eines Array-Elements dar.
Es werden nur die Mitglieder exportiert, die vom Anwender bearbeitet wurden.
- AttributeList
Die <AttributeList> umfasst alle definierten Attribute eines Mitglieds. Attribute, die vom System festgelegt oder von einem Standardwert zugewiesen sind, werden in der XML-Struktur nicht aufgeführt.
Die Mitgliedsattribute <ReadOnly> und <Informative> werden nur in die XML-Exportdatei geschrieben, wenn ihr Wert TRUE ist.

- StartValue

Das Element <StartValue> wird nur geschrieben, wenn der Standardwert der Variablen oder Konstanten vom Anwender festgelegt ist.

```
...
<Member Name="Static_1" Datatype="Int">
...
<StartValue>1</StartValue>
</Member>
...
```

- Kommentar

Das Element <Comment> wird geschrieben, wenn es vom Anwender festgelegt ist.

Kommentare einer Variablen oder Konstanten werden als mehrsprachiger Text exportiert:

```
...
<Member Name="Static_3" Datatype="Struct">
<AttributeList>
<BooleanAttribute Name="ExternalAccessible" SystemDefined="true">false</BooleanAttribute>
<BooleanAttribute Name="ExternalVisible" SystemDefined="true">false</BooleanAttribute>
<BooleanAttribute Name="ExternalWritable" SystemDefined="true">false</BooleanAttribute>
</AttributeList>
<Comment>
<MultiLanguageText Lang="de-DE">An individual comment</MultiLanguageText>
</Comment>
</Member>
...
```

Attribute

- Hauptattribute

Die Hauptattribute werden in das Element <Member> in der XML-Struktur geschrieben.

```
...
<Member Name="Static_1" Datatype="User_data_type_1" Remanence="Retain">
...
</Member>
...
```

Die folgende Tabelle zeigt die Hauptattribute einer Variablen oder Konstanten am Abschnitt Bausteinschnittstelle.

Name	Datentyp	Standardeinstellung	Importbedingung	Kommentar
Name	STRING	-	Erforderlich	
Datentyp	ENUM	-	Erforderlich	
Version	STRING	-	Optional	
Remanenz	ENUM	NonRetain	-	Wird nur geschrieben, wenn es sich nicht um die Standardeinstellung handelt
Zugänglichkeit	ENUM	Öffentlich	-	Vom System vordefiniert Kann vom Anwender nicht geändert werden
Informativ	BOOL	FALSE	-	

Mitglieder mit dem Merker "Informativ" werden beim Import ignoriert. Wenn das Attribut gelöscht oder auf FALSE gesetzt ist, wird eine Ausnahme ausgelöst.

Hinweis

Remanenzeinstellungen "Im IDB setzen"

Wenn der Remanenzwert einer Variablen oder Konstanten "Im IDB setzen" ist, muss die im IDB festgelegte Remanenz für alle anderen Variablen und Konstanten mit dem Remanenzwert "SetInIDB" die gleiche sein.

Das erste importierte Mitglied mit dem Attribut "Im IDB setzen" legt die erwartete Remanenz im IDB für die folgenden Variablen und Konstanten mit dem Remanenzwert "SetInIDB" fest.

- Systemdefinierte Mitgliedsattribute

Systemdefinierte Mitgliedsattribute werden im Element <AttributeList> aufgelistet.

Systemdefinierte Mitgliedsattribute werden mit dem Merker <Informativ> gekennzeichnet und beim Import ignoriert.

```

...
<Member Name="Static_3" Datatype="Struct">
  <AttributeList>
    <BooleanAttribute Name="ExternalAccessible" SystemDefined="true">false</BooleanAttribute>
    <BooleanAttribute Name="ExternalVisible" SystemDefined="true">false</BooleanAttribute>
    <BooleanAttribute Name="ExternalWritable" SystemDefined="true">false</BooleanAttribute>
  </AttributeList>
  <Comment>
    <MultiLanguageText Lang="de-DE">An individual comment</MultiLanguageText>
  </Comment>
</Member>
...

```

Name	Typ	Standardeinstellung	SimaticML schreibgeschützt (informativ)	Kommentar
At	String	""	FALSE	Mitgliedsanteile sind mit einem anderen Mitglied in dieser Struktur versetzt
SetPoint	Bool	FALSE	FALSE	Mitglied kann mit Arbeitsspeicher synchronisiert werden
UserReadOnly	Bool	FALSE	TRUE	Anwender kann keine Mitgliedsattribute ändern (einschl. Name)
UserDeletable	Bool	TRUE	TRUE	Mitglied kann im Editor nicht gelöscht werden
HmiAccessible	Bool	TRUE	FALSE	Kein HMI-Zugriff, kein Strukturelement
HmiVisible	Bool	TRUE	FALSE	Filter zum Verringern der Anzahl Mitglieder, die zuerst angezeigt werden
Offset	Int	-	TRUE	DB, FB, FC (Temp). Bei klassischen PLCs und Plus PLCs, wo die klassische Remanenz eingestellt ist.

8.5 Daten eines PLC-Geräts importieren/exportieren

Name	Typ	Standardeinstellung	SimaticML schreibgeschützt (informativ)	Kommentar
PaddedSize	Int	-	TRUE	DB, FB, FC (Temp). Bei klassischen PLCs und Plus PLCs, wo die klassische Remanenz eingestellt ist. Nur bei Arrays.
HiddenAssignment	Bool	FALSE	FALSE	Zuweisung bei Aufruf ausblenden, wenn sie PredefinedAssignment entspricht.
PredefinedAssignment	String	""	FALSE	Eingabe für den verwendeten Parameter, wenn der Aufruf platziert wird.
ReadOnlyAssignment	Bool	FALSE	FALSE	Der Anwender kann die vordefinierte Zuweisung beim Aufruf nicht ändern.
UserVisible	Bool	TRUE	TRUE	Dieses Mitglied wird in der UI nicht angezeigt.
HmiReadOnly	Bool	TRUE	TRUE	Dieses Mitglied für die HMI schreibgeschützt.
CodeReadOnly	Bool	FALSE	TRUE	-

- Anwenderdefinierte Attribute

Anwenderdefinierte Attribute werden mit dem Merker <ReadOnly> gekennzeichnet.

Mitglieder mit diesem Merker werden beim Import ignoriert. Wenn der Merker gelöscht oder auf FALSE gesetzt ist, wird eine Ausnahme ausgelöst.

Nicht bearbeitete anwenderdefinierte Attribute werden vom Export ausgeschlossen.

Name	Typ	Standardeinstellung	SimaticML schreibgeschützt (informativ)	Kommentar
CFC	IBlockAttribute	---	FALSE	Dies ist eine Ladung

Datentyp "STRUCT"

Die Komponenten des Datentyps "STRUCT" werden in der XML-Struktur einer Import-/Exportdatei als verschachtelte Mitglieder dargestellt:

```
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Static">
    <Member Name="Static_1" Datatype="Struct">
      <!-- Basic struct -->
      <Member Name="Static_1" Datatype="Int">
        <!-- First Member of struct -->
        <StartValue>1</StartValue>
      </Member>
      <Member Name="Static_2" Datatype="Int">
        <!-- Second Member of struct -->
      </Member>
      <Member Name="Static_3" Datatype="Struct">
        <!-- A subsequent struct -->
        <Member Name="Static_1" Datatype="Int">
          <!-- First Member of the subsequent struct -->
          <StartValue>3</StartValue>
        </Member>
        <Member Name="Static_2" Datatype="Int">
          <!-- Second Member of the subsequent struct -->
        </Member>
      </Member>
    </Section>
  </Sections>
```

Datentyp "ARRAY" Basistyp

Die Komponenten des Basisdatentyps "ARRAY" werden in der XML-Struktur einer Import-/Exportdatei als Unterelemente mit dem Attribut "Path" dargestellt:

```
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Static">
    <Member Name="Static_1" Datatype="Array[0..2] of Int" Remanence="Retain">
      <!-- Basic Array -->
      <Subelement Path="0">
        <!-- First Array Component-->
        <StartValue>1</StartValue>
      </Subelement>
      <Subelement Path="1">
        <!-- Second Array Component-->
        <StartValue>2</StartValue>
      </Subelement>
    </Member>
  </Section>
</Sections>
```

Datentyp "ARRAY" von UDT

Die Komponenten des Datentyps "ARRAY" eines UDT werden in der XML-Struktur einer Import-/Exportdatei als neues Element `<sections>` in einem Element `<member>` dargestellt. Die Mitglieder im neuen Abschnitt für UDT in einem ARRAY sind als Unterelemente mit dem Attribut "Path" zugewiesen:

```
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Static">
    <Member Name="Static_1" Datatype=""User_data_type_1"" Remanence="Retain">
      <Sections> <!-- Sections including the UDT "User_data_type_1" -->
        <Section Name="None">
          <Member Name="Element_2" Datatype="Int">
            <StartValue>47</StartValue>
          </Member>
        </Section>
      </Sections>
    </Member>
    <Member Name="Static_2" Datatype="Array[0..1] of "User_data_type_1"">
      <Sections> <!-- Sections including the UDT "User_data_type_1" -->
        <Section Name="None">
          <Member Name="Element_2" Datatype="Int">
            <Subelement Path="0"> <!-- Component of the array -->
              <StartValue>123</StartValue>
            </Subelement>
          </Member>
        </Section>
      </Sections>
    </Member>
  </Section>
</Sections>
```

Datentyp "ARRAY" in "ARRAY"

Die Komponenten des Datentyps "ARRAY" in einem anderen ARRAY werden in der XML-Struktur einer Import-/Exportdatei als Unterelemente mit dem Attribut "Path" dargestellt.

Die Mitglieder in einem anderen ARRAY werden als Unterelemente mit dem Attribut "Path" zugewiesen, wenn die Komponente vom Anwender bearbeitet wird:

```

<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Static">
    <Member Name="Static_1" Datatype="Array[0..2] of Struct">
      <Member Name="Static_1" Datatype="Int" />
      <Member Name="Static_2" Datatype="Array[0..1, 0..3, -9..-2] of Struct">
        <Member Name="Static_1" Datatype="Int">
          <Subelement Path="0,0,3,-5">
            <StartValue>1</StartValue>
          </Subelement>
        </Member>
        <Subelement Path="0,0,2,-6">
          <Comment>
            <MultiLanguageText Lang="de-DE">A individual comment</MultiLanguageText>
          </Comment>
        </Subelement>
      </Member>
    </Member>
  </Section>
</Sections>

```

PLC-Datentypen (UDT)

Die XML-Struktur eines PLC-Datentyps ist von den Exporteinstellungen in TIA Portal Openness abhängig.

- ExportOptions.None
Mitglieder des PLC-Datentyps werden nur geschrieben, wenn der Standardwert von mindestens einer der Komponenten vom Anwender festgelegt ist. Bei diesen Mitgliedern werden nur die zwei zusätzlichen Attribute "Name" und "Datatype" geschrieben, um das Mitglied zu identifizieren, zu dem der <StartValue> gehört. Andere Mitglieder und Attribute werden nicht geschrieben.
- ExportOptions.WithDefaults
Die folgenden Attribute werden immer geschrieben:
 - Name
 - Datatype
 - ExternalAccessible
 - ExternalVisible
 - ExternalWritable
 - SetPoint
 - StartValue
Sie werden nur in XML geschrieben, wenn der Standardwert in diesem Typ vom Anwender festgelegt ist. Wenn er nur im PLC-Datentyp festgelegt ist, wird er nicht geschrieben.
- ExportOptions.ReadOnly
Bei PLC-Datentypen führt diese Einstellung nicht zu einem aussagekräftigen Ergebnis. In Kombination mit anderen Einstellungen hat dies keine Auswirkung auf das Ergebnis.

Überlagerte Variablen

Wenn eine Variable mit einem neuen Datentyp überlagert wird, werden die Mitglieder in der XML-Struktur des neuen Datentyps dargestellt. Die folgende XML-Struktur zeigt einen Datentyp WORD, der mit einem ARRAY von BYTE überlagert ist.

```
<Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
  <Section Name="Input" />
  <Section Name="Output" />
  <Section Name="InOut" />
  <Section Name="Static">
    <Member Name="Static_1" Datatype="Word">
      <!-- Basic Member -->
      <StartValue>16#17</StartValue>
    </Member>
    <Member Name="Static_2" Datatype="Array[0..1] of Byte">
      <AttributeList>
        <StringAttribute Name="At" SystemDefined="true">Static_1</StringAttribute>
      </AttributeList>
      <!-- The AT member -->
      <Subelement Path="0">
        <!-- First overlay byte -->
      </Subelement>
      <Subelement Path="1">
        <!-- Second overlay byte -->
      </Subelement>
    </Member>
  </Section>
  <Section Name="Temp" />
  <Section Name="Constant" />
</Sections>
```

Bausteinschnittstelle

Alle Attribute mit `ReadOnly = "TRUE"` und `Informative = "FALSE"` werden ausgeschlossen. Die XML-Struktur einer Bausteinschnittstelle ist von den Exporteinstellungen in TIA Portal Openness abhängig.

- `ExportOptions.None`
Diese Einstellung exportiert nur die geänderten Daten oder die Daten, die von den Standardwerten abweichen.
Wenn die Attributdefinition keinen Standardwert angibt, wird das Attribut immer geschrieben.
Die Exportdatei enthält auch alle Werte, die für den anschließenden Datenimport obligatorisch sind.
- `ExportOptions.WithDefaults`
Die folgenden Attribute werden immer geschrieben:
 - `Name`
 - `Datatype`
 - `HmiAccessible` exportiert als `ExternalAccessible`
 - `HmiVisible` exportiert als `ExternalVisible`
 - `ExternalWritetable`
 - `SetPoint` (sofern vorhanden)
 - `Offset` (sofern vorhanden)
 - `PaddedSize` (sofern vorhanden)Alle anderen Attribute werden nur geschrieben, wenn sich ihre Werte vom Standardwert unterscheiden.
Das Element `<StartValue>` wird nur in XML geschrieben, wenn es explizit festgelegt wurde.
- `ExportOptions.ReadOnly`
Bei Bausteinschnittstellen führt diese Einstellung nicht zu einem aussagekräftigen Ergebnis. In Kombination mit anderen Einstellungen hat dies keine Auswirkung auf das Ergebnis.

8.5.1.2 Änderungen des Objektmodells und Dateiformat XML

Einleitung

Um eine kundenspezifisch erzeugte oder editierte XML-Datei über TIA Portal Openness erfolgreich in das TIA Portal zu importieren, muss die Datei definierten Schemas entsprechen.

Die XML-Dateien bestehen immer aus zwei Hauptteilen:

- Schnittstelle
- Übersetzungseinheit

Die Schemas, denen die Dateien entsprechen müssen, werden nachstehend erläutert.

Schnittstelle

Eine Schnittstelle kann mehrere Abschnitte enthalten (z. B. Input, InOut, Static): Sie finden alle diese Abschnitte im folgenden Verzeichnis:

- C:\Programme\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.InterfaceSections_v3.xsd
- C:\Programme\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.Interface.Snapshot.xsd

Übersetzungseinheit

Es gibt separate Schemas für die Übersetzungseinheiten der GRAPH-, KOP-/FUP-, AWL- und SCL-Bausteine. Sie finden diese Schemas in den folgenden Verzeichnissen:

- GRAPH: C:\Programme\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.PlcBlocks.Graph_v4.xsd
- KOP/FUP: C:\Programme\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.PlcBlocks.LADFBDB_v3.xsd
- AWL: C:\Programme\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.PlcBlocks.STL_v3.xsd
- SCL: C:\Programme\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.PlcBlocks.SCL_v2.xsd

Subschemas

Es gibt folgende zusätzliche Schemadefinitionen, die von allen Übersetzungseinheiten verwendet werden:

- Zugriff
- Allgemein

Zugriff

Der Zugriffsknoten beschreibt zum Beispiel:

- lokale/globale Mitglieder und konstante Nutzungen
- FB-, FC-, Anweisungsaufälle
- DBs für Aufrufe

Sie finden das Zugriffsschema im folgenden Verzeichnis:

C:\Programme\Siemens\Automation\Portal V*\PublicAPI\V*\Schemas\SW.PlcBlocks.Access_v3.xsd

Allgemein

Hierunter fallen die allgemein verwendeten Attribute und Elemente, zum Beispiel Kommentare verschiedener Art, Texte und Token.

Sie finden das allgemeine Schema im folgenden Verzeichnis:

C:\Programme\Siemens\Automation\Portal V*\PublicAPI\V*.schemas\SW.Common_v2.xsd

Hinweis

V* bezieht sich auf die installierte Version von TIA Portal.

8.5.1.3 DBs mit Schnappschüssen exportieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Mit TIA Portal Openness können Sie die DBs mit Schnappschusswerten als XML exportieren und die Werte mit verschiedenen Schnappschusszeiten vergleichen. Mit dem Vergleichsergebnis können sie manuell einzelne Startwerte in der UI anpassen und für die spätere Wiederherstellung speichern. Das Schema "SW.Interface.Snapshot.xsd" unterstützt Sie beim Auswerten oder Verarbeiten der exportierten XML-Datei.

Programmcode

Ändern Sie folgenden Programmcode, um Schnappschusswerte mit dem Snapshot Service zu exportieren:

```
InterfaceSnapshot interfaceSnapshot = dataBlock.GetService<InterfaceSnapshot>();  
interfaceSnapshot.Export(new FileInfo("C:\temp\MyInterfaceSnapshot.xml"),  
ExportOptions.None);
```

Der Snapshot Service "InterfaceSnapshot" wird im Namespace "Siemens.Engineering.SW.Blocks" bereitgestellt. Das Handling der Dateien (z. B. wenn das Exportverzeichnis nicht existiert; Erstellen des Exportverzeichnisses; wenn das Exportverzeichnis schreibgeschützt ist; wenn die Exportdatei bereits existiert) ist das gleiche wie beim normalen Export über die Standardschnittstelle von Openness. Der Snapshot Service wird unterstützt für globale DBs, Instanz-DBs und Array-DBs.

Hinweis

Der Export von Schnappschusswerten mit dem Snapshot Service ist unabhängig vom Export über die Standardschnittstelle von Openness und beeinflusst daher nicht den bereits vorhandenen Export der Schnittstellenmitglieder. Die exportierte XML kann nicht mehr importiert werden.

8.5 Daten eines PLC-Geräts importieren/exportieren

Die Schnappschusswerte werden wie folgt exportiert:

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
  <Engineering version="V15 SP1" />
  <DocumentInfo>
    ...
  </DocumentInfo>
  <SW.Blocks.InterfaceSnapshot ID="0">
    <AttributeList>
      <Name>GlobalDB</Name>
      <Snapshot ReadOnly="true"><SnapshotValues>
        <SnapshotValues>
          <Value Path="Static_1" Type="Bool">TRUE</Value>
          <Value Path="Static_2[0]" Type="Int">1</Value>
          <Value Path="Static_2[1]" Type="Int">2</Value>
          <Value Path="Static_2[2]" Type="Int">3</Value>
          <Value Path="Static_3" Type="DTL">DTL#1973-01-00:00:00</Value>
          <Value Path="Static_4.Element_1" Type="Int">7</Value>
          <Value Path="Static_4.Element_2[0]" Type="Bool">FALSE</Value>
          <Value Path="Static_4.Element_2[1]" Type="Bool">TRUE</Value>
          <Value Path="Static_4.Element_2[2]" Type="Bool">TRUE</Value>
          <Value Path="Static_4.Element_3.Element_1" Type="Int">5</Value>
          <Value Path="Static_4.Element_3.Element_2.Element_1" Type="Bool">TRUE</Value>
          <Value Path="Static_4.Element_3.Element_2.Element_2[0]" Type="Int">100</Value>
          <Value Path="Static_4.Element_3.Element_2.Element_2[1]" Type="Int">200</Value>
        </SnapshotValues></Snapshot>
        <SnapshotDate ReadOnly="true">2017-12-06T08:04:11.4590585Z</SnapshotDate>
        <StructureModified ReadOnly="true">2017-12-06T08:22:13.3292585Z</StructureModified>
      </AttributeList>
    </SW.Blocks.InterfaceSnapshot>
  </Document>
```

Wenn ein DB keine Schnappschusswerte enthält, würde der Inhalt der exportierten Datei folgendermaßen aussehen:

```
<SnapshotValues xmlns="http://www.siemens.com/automation/Openness/SW/Interface/Snapshot/v1"></SnapshotValues>
```

Siehe auch

[Projekt öffnen \(Seite 111\)](#)

8.5.1.4 Bausteine mit Know-how-Schutz exportieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Die PLC ist nicht online.

Verwendung

Die resultierende XML-Datei ähnelt der Exportdatei eines Bausteins ohne Knowhow-Schutz. Mit PlcBlockProtectionProvider können Sie einen Knowhow-geschützten Baustein entsperren, indem Sie in der Openness API das Passwort eingeben und dann einen KHP-Baustein komplett mit seinem Code exportieren. Ein Baustein kann importiert und dann wieder über die API mit einem Passwort geschützt werden.

Wenn der Baustein ohne Entsperren exportiert wird, wird nur die öffentliche Bausteinschnittstelle exportiert.

- TIA Portal-Baustein -> Entsperren -> Datei ohne Schutz exportieren
- Datei ohne Schutz importieren -> Sperren -> TIA Portal-Baustein

Die Attributliste des Bausteins zeigt an, dass der entsprechende Baustein Knowhow-Schutz besitzt.

Programmcode

Um die sichtbaren Daten eines Bausteins mit Knowhow-Schutz in eine XML-Datei zu exportieren, ändern Sie folgenden Programmcode:

```
private static void ExportBlock(PlcSoftware plcSoftware)
{
    PlcBlock plcBlock = plcSoftware.BlockGroup.Blocks.Find("MyBlock");
    plcBlock.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml", plcBlock.Name)), 
    ExportOptions.WithDefaults);
}
```

8.5.1.5 Export/Import von SCL-Bausteinen

SCL-Anweisungen mit XML-Tags für Export

Die Exportoperation für SCL-Bausteine exportiert ihre entsprechenden XML-Tags auf der Grundlage des SCL-Anweisungstyps. Diese Operation unterstützt die SCL-Netzwerke von SCL-Anweisungen in KOP/FUP-Bausteinen von SCL-Anweisungen. Die SCL-Anweisungen sind klassifiziert als Textelemente, Operanden, Ausdrücke, Steueranweisungen usw. Das Schema SW.PlcBlocks.SCL_v2.xsd ist verfügbar, um Sie beim Verarbeiten des SCL-Bausteins mit Export-XML-Tags zu unterstützen. Die SCL-Bausteinanweisungen mit ihren entsprechenden exportierten XML-Tags und Attributen sind unten angegeben.

Neue Zeile

Neue Zeilen in SCL-Bausteinen werden durch NewLine XML-Tags dargestellt.

- Enthält das vorzeichenlose Attribut Num mit dem Standardwert 1.
- Attribut Num hat nicht den Wert 0.
- Nur bei SCL unterstützt.

SCL-Baustein	XML-Tag
	<NewLine Num="2" />

Leer

Leerzeichen in SCL-Bausteinen werden durch Blank XML-Tags dargestellt.

- Enthält das vorzeichenlose Attribut Num mit dem Standardwert 1.
- Attribut Num hat nicht den Wert 0.
- Nur bei SCL unterstützt.
- Unterstützt nicht das Attribut Integer, das in anderen Sprachen von STEP 7 verfügbar ist.

SCL-Baustein	XML-Tag
	<Blank Num="2" />

Einrückung von SCL-Bausteinanweisungen

In den Einstellungen im TIA Portal können Sie die Einrückung von SCL-Code über Options/Settings/General/Script/text editors ändern. In der folgenden Tabelle wird die Art der Einrückung auf der Grundlage des Ident-Modus definiert.

Ident-Modus	Ergebnis
Ohne	Importoperation fügt die Leerzeichen ein, wie in den Quelldateien verfügbar.
Paragraph oder Smart	Importoperation fügt die angegebenen Ident-Leerzeichen in die importierte Datei ein.

Auf der Grundlage der gewählten Einrückung wird diese in der importierten XML file des SCL-Bausteins vorgenommen.

Kommentar

Ein- und mehrzeilige Kommentare in SCL-Bausteinen werden durch LineComment XML-Tags dargestellt.

- Nur das Tag LineComment (für einsprachigen Kommentar) wird von SCL unterstützt.
- Das Tag Comment (für mehrsprachigen Kommentar) wird von SCL nicht unterstützt.
- Enthält das Attribut Inserted mit Standardwert false.
- Inserted="false" zeigt "://" einzeiligen Kommentar in SCL-Bausteinen an.
- Inserted="true" zeigt "(*)" mehrzeiligen Kommentar in SCL-Bausteinen an.
- NoClosingBracket="true" zeigt Kommentare ohne schließende Klammer in SCL-Bausteinen an. Dieses Attribut ist optional und hat den Standardwert false.
- XML zeigt keine Kommentarhierarchie in SCL-Bausteinen an.

SCL-Baustein	XML-Tag
// one line comment	<LineComment> <Text>one line comment</Text> </LineComment>
(* one line comment second line *)	<LineComment Inserted="true"> <Text>one linecomment secondline</Text> </LineComment>
(* first comment (* second comment *) end first comment *)	<LineComment Inserted="true"> <Text> first comment (* second comment *) end first comment</Text> </LineComment > Der verschachtelte Kommentar ist Bestandteil des äußersten Kommentartexts.
(* comment without closing bracket	<LineComment Inserted="true" NoClosingBracket="true"> <Text> comment without closing bracket</Text> </LineComment >

Region

Regionen in SCL-Bausteinen werden durch Token XML-Tags dargestellt.

- Das Text XML-Tag repräsentiert den Regionsnamen (region_name).
- Beim Attribut Text des Token XML-Tags spielt die Groß- und Kleinschreibung keine Rolle.

8.5 Daten eines PLC-Geräts importieren/exportieren

- Beim Importvorgang wird die Groß- und Kleinschreibung beachtet und der Editor zeigt die Schlüsselwörter wie in den TIA Portal-Einstellungen konfiguriert an.
- Wenn das Schlüsselwort end_region mit ";" (Semikolon) im SCL-Baustein endet, steht das Zeichen ";" im Text XML-Tag.

SCL-Baustein	XML-Tag
<pre>region myregion ... end_region here is the end of myregion</pre>	<pre><Token Text="REGION" /> <Blank /> <Text>myregion</Text> <NewLine /> ... <Token Text="END_REGION" /> <Blank /> <Text>here is the end of myregion</Text> <NewLine /></pre>
<pre>region // here are no blanks ... end_region</pre>	<pre><Token Text="REGION" /> <NewLine /> <LineComment .../> <Token Text="END_REGION" /> <NewLine /></pre>
<pre>region ... end_region;</pre>	<pre><Token Text="REGION" /> <NewLine /> ... <Token Text="END_REGION" /> <Text>;</Text> <NewLine /></pre>

Pragma

Pragma in SCL-Bausteinen werden durch Token XML-Tags dargestellt. Die Parameter werden im XML-Tag Access mit dem Attribut Scope als LiteralConstant dargestellt.

SCL-Baustein	XML-Tag
<pre>{PRAGMA_BEGIN 'Param1', 'Param2' (*parm 2*) // something else {PRAGMA_END}</pre>	<Token Text="{" /> <Token Text="PRAGMA_BEGIN" /> <Blank /> <Access Scope="LiteralConstant"> <Constant> <ConstantValue>'Param1'</ ConstantValue> </Constant> </Access> <Token Text="," /> <Blank /> <Access Scope="LiteralConstant"> <Constant> <ConstantValue>'Param2'</ ConstantValue> </Constant> </Access> <Blank /> <LineComment Inserted="True"> <Text>param 2</Text> </LineComment> <Token Text="," /> <Blank /> <Token Text="}" /> <NewLine /> <LineComment> <Text> something else</Text> </LineComment> <NewLine /> <Token Text="{" /> <Token Text="PRAGMA_END" /> <Token Text="}" />

Konstanten: Literale Konstanten

Die Konstanten in SCL-Bausteinen werden durch Access XML-Tags dargestellt.

- Das Attribut Scope kann Werte wie LiteralConstant, TypedConstant, LocalConstant, und GlobalConstant. haben.
- Die Namen von Konstanten, denen das Zeichen "#" vorangestellt ist, werden in XML ignoriert.
- Das Zeichen "#" wird während des Importvorgangs von XML hinzugefügt.
- Der Wert globaler Konstanten, repräsentiert durch Anführungszeichen, wird in XML ignoriert.
- Die Anführungszeichen werden während des Importvorgangs von XML hinzugefügt.

8.5 Daten eines PLC-Geräts importieren/exportieren

Art der Konstante	SCL-Baustein	XML-Tag
Literele Konstante: Integer	#Out := 10;	<pre><Access Scope="LiteralConstant"> <Constant> <ConstantValue>10</ConstantValue> <ConstantTypeInformative="true">LINT</ ConstantType> </Constant> </Access></pre>
Literele Konstante: String	#myString := 'Hello world';	<pre><Access Scope="LiteralConstant"> <Constant> <ConstantValue>Hello world</ ConstantValue> <ConstantTypeInformative="true">STRING</ ConstantType> </Constant> </Access></pre>
Literale Konstante: Typed	#Out := int#10;	<pre><Access Scope="TypedConstant"> <Constant> <ConstantValue>int#10</ConstantValue> </Constant> </Access></pre>
		<p>Format von XML nach Export mit Einstellung ExportOptions.ReadOnly.</p> <pre><Access Scope="TypedConstant"> <Constant> <ConstantValue>int#10</ConstantValue> <StringAttribute Name="Format" Informative="true">Dec_signed</ StringAttribute> <StringAttribute Name="FormatFlags" Informative="true">TypeQualifier</ StringAttribute> </Constant> </Access></pre>
Lokale Konstante	#Out := #mylocal;	<pre><Access Scope="LocalConstant"> <Constant Name="mylocal" /> </Access></pre>
		<p>Format von XML nach Export mit Einstellung ExportOptions.ReadOnly</p> <pre><Access Scope="LocalConstant"> <Constant Name="mylocal"> <ConstantType Informative="true">Int</ ConstantType> <ConstantValue Informative="true">10</ ConstantValue> <StringAttribute Name="Format" Informative="true">Dec_signed</ StringAttribute> </Constant> </Access></pre>

Art der Konstante	SCL-Baustein	XML-Tag
Globale Konstante	#Out := "myglobal";	<pre><Access Scope="GlobalConstant"> <Constant Name="myglobal" /> </Access></pre> <p>Format von XML nach Export mit Einstellung ExportOptions.ReadOnly.</p> <pre><Access Scope="GlobalConstant"> <Constant Name="myglobal"> <ConstantType Informative="true">Int</ConstantType> <ConstantValue Informative="true">10</ConstantValue> <StringAttribute Name="Format" Informative="true">Dec_signed</StringAttribute> </Constant> </Access></pre>

Die Adresskonstanten werden in SCL-Bausteinen nicht unterstützt und in dieser Tabelle ignoriert.

Variablen

Die lokalen und globalen Variablen in SCL-Bausteinen werden durch Access XML-Tags dargestellt.

- Das Attribut Scope hat Werte von LocalVariable und GlobalVariable.
- Das XML-Tag für das Zuweisen des Werts 10 wird hier ignoriert.

Art der Variable	SCL-Baustein	XML-Tag
Lokale Variable	#Out := 10;	<pre><Access Scope="LocalVariable"> <Symbol> <Component Name="Out" /> </Symbol> </Access></pre>
Globale Variable	"Tag_3":= 10;	<pre><Access Scope="GlobalVariable"> <Symbol> <Component Name="Tag_3" /> </Symbol> </Access></pre> <p>Format von XML nach Export mit Einstellung ExportOptions.ReadOnly.</p> <pre><Access Scope="GlobalVariable"> <Symbol> <Component Name="Tag_3" /> <Address Area="Memory" Type="Int" BitOffset="96" Informative="true" /> </Symbol> </Access></pre>

Ausdrücke

Die einfachen Ausdrücke in SCL-Bausteinen werden durch Access XML-Tags dargestellt. Das Attribut Scope hat den Wert von LocalVariable für die Ausdrücke.

SCL-Baustein	XML-Tag
#a := #b + #c;	<Access Scope="LocalVariable"> <Symbol> <Component Name="a" /> </Symbol> </Access> <Blank /> <Token text=":=" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="b" /> </Symbol> </Access> <Blank /> <Token text="+" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="c" /> </Symbol> </Access> <Token text=";" />

Steuerstrukturen in SCL-Bausteinen

Die Steueranweisungen wie IF, CASE, FOR, WHILE, REPEAT, GOTO, EXIT, CONTINE, and RETURN werden durch Token XML -Tags dargestellt.

- Die im SCL-Baustein verwendeten bedingten Symbole wie >, <, & werden in XML als Escape-Sequenzen (< ; > ; &) dargestellt.
- Diese Kombination von XML-Tags ist nur bei SCL-Bausteinen gültig. In anderen Sprachen wird eine Ausnahme ausgelöst.

Name des Bausteins	SCL-Baustein	XML-Tag
IF	IF #a<#c THEN ; END_IF;	<Token Text="IF" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="a" /> </Symbol> </Access> <Token Text=";" /> <Access Scope="LocalVariable"> <Symbol> <Component Name="c" /> </Symbol> </Access> <Blank /> <Token Text="THEN" /> <NewLine /> <Blank Num="4" /> <Token Text=";" /> <NewLine /> <Token Text="END_IF" /> <Token Text=";" />

8.5 Daten eines PLC-Geräts importieren/exportieren

Name des Bausteins	SCL-Baustein	XML-Tag
CASE	<pre>CASE #a OF 1 (*test*): // Statement section case 1 ; 2..4: // Statement section case 2 to 4 ; ELSE // Statement section ELSE ; END_CASE;</pre>	<Token Text="CASE" /><Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="a" /> </Symbol> </Access> <Blank /> <Token Text="OF" /> <NewLine /> <Blank Num="2"/> <Access Scope="LiteralConstant"> <Constant> <ConstantValue>1</ConstantValue> <ConstantType Informative="true">LINT</ConstantType> </Constant> </Access> <Blank /> <LineComment Inserted="true"> <Text>test</Text> </LineComment > <Token Text=":" /> <Blank /> <LineComment> <Text> Statement section case 1</Text> </LineComment > <NewLine /> <Blank Num="4"/> <Token Text=";" /> <NewLine /> <Blank Num="2"/> <Access Scope="LiteralConstant"> <Constant> <ConstantValue>2</ConstantValue> <ConstantType Informative="true">LINT</ConstantType> </Constant> </Access> <Token Text=".." /> <Blank Num="2"/> <Access Scope="LiteralConstant"> <Constant> <ConstantValue>4</ConstantValue> <ConstantType Informative="true">LINT</ConstantType> </Constant> </Access> <Blank /> <LineComment> <Text> Statement section case 2 to 4</Text> </LineComment >

Name des Bausteins	SCL-Baustein	XML-Tag
		<NewLine /> <Blank Num="4"/> <Token Text=";" /> <NewLine /> <Blank Num="2"/> <Token Text="ELSE" /> <NewLine /> <Blank Num="4"/> <Token Text=";" /> <NewLine /> <Token Text="END_CASE" /> <Token Text=";" />

8.5 Daten eines PLC-Geräts importieren/exportieren

Name des Bausteins	SCL-Baustein	XML-Tag
FOR	FOR #i := #a TO #b DO // Statement section FOR ; END_FOR;	<Token Text="FOR" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="i" /> </Symbol> </Access> <Blank /> <Token Text=":=" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="a" /> </Symbol> </Access> <Blank /> <Token Text="TO" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="b" /> </Symbol> </Access> <Blank /> <Token Text="DO" /> <NewLine /> <Blank Num="2" /> <LineComment> <Text> Statement section FOR</Text> </LineComment > <NewLine /> <Blank Num="2" /> <Token Text=";" /> <NewLine /> <Token Text="END_FOR" /> <Token Text=";" />

8.5 Daten eines PLC-Geräts importieren/exportieren

Name des Baustein-eins	SCL-Baustein	XML-Tag
WHILE	<pre>WHILE #a<#b DO // Statement section WHILE ; END WHILE;</pre>	<pre><Token Text="WHILE" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="a" /> </Symbol> </Access> <Token Text="<" /> <Access Scope="LocalVariable"> <Symbol> <Component Name="b" /> </Symbol> </Access> <Blank /> <Token Text="DO" /> <NewLine /> <Blank Num="2" /> <LineComment> <Text> Statement section WHILE</Text> </LineComment > <NewLine /> <Blank Num="2" /> <Token Text=";" /> <NewLine /> <Token Text="END WHILE" /> <Token Text=";" /></pre>

8.5 Daten eines PLC-Geräts importieren/exportieren

Name des Bausteins	SCL-Baustein	XML-Tag
REPEAT	<pre>REPEAT // Statement section REPEAT ; UNTIL #a<#b END_REPEAT;</pre>	<pre><Token Text="REPEAT" /> <NewLine /> <Blank Num="2" /> <LineComment> <Text> Statement section REPEAT</Text> </LineComment > <NewLine /> <Blank Num="2" /> <Token Text=";" /> <NewLine /> <Token Text="UNTIL" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="a" /> </Symbol> </Access> <Token Text="&lt;" /> <Access Scope="LocalVariable"> <Symbol> <Component Name="b" /> </Symbol> </Access> <Blank /> <Token Text="END_REPEAT" /> <Token Text=";" /></pre>

Name des Bausteins	SCL-Baustein	XML-Tag
GOTO	here // well : // this is goto statement	<p>XML-Beispiel für Definition von GOTO-Labels</p> <pre><Blank Num="3"/> <Access Scope="Label"> <Label Name="here"> <NewLine /> <Blank Num="3"/> <LineComment> <Text> well</Text> </LineComment> <NewLine /> <Token Text=":" /> <Blank /> </Label> </Access> <LineComment> <Text> this is goto statement</Text> </LineComment></pre>
	GOTO (*comment*) here;	<p>XML-Beispiel für Verwendung von GOTO-Labels</p> <pre><Token Text="GOTO" /> <Blank /> <LineComment inserted="true"> <Text>comment</Text> </LineComment> <Blank /> <Access Scope="Label"> <Label Name="here" /> </Access> <Token Text=";" /></pre>

Referenzierende Attribute

Die referenzierenden Attribute von SCL-Bausteinen werden durch das Attribut AccessModifier des Tags Component dargestellt.

- Zum einfachen Referenzieren hat AccessModifier den Wert als Reference.
- Zur Array-Referenzierung hat AccessModifier den Wert als ReferenceToArray..

SCL-Baustein	XML-Tag
RefToUDT^ (*RefToUDT*) .element	<pre> <Symbol> <Component Name="RefToUDT" AccessModifier="Reference" /> <Token Text="^" /> <LineComment Inserted="True"> <Text>RefToUDT</Text> </LineComment> <Token Text="." /> <Component Name="element" /> </Symbol></pre>
RefToArrayOfUDT^(*RefToArrayOfUDT*)[#i].element	<pre> <Symbol> <Component Name="RefToArrayOfUDT" AccessModifier="ReferenceToArray" /> <Token Text="^" /> <LineComment Inserted="True"> <Text>RefToArrayOfUDT</Text> </LineComment> <Token Text="[" /> <Access Scope=LocalVariable> <Symbol> <Component Name="i" /> </Symbol> </Access> <Token Text="]" /> </Component> <Token Text="." /> <Component Name="element" /> </Symbol></pre>

8.5.1.6 Export/Import strukturierter Typen von SCL-Bausteinen

Strukturierte SCL-Typen mit XML-Tags für den Export

In die strukturierten SCL-Typen können Sie Leerzeichen, neue Zeilen und Kommentare zu den SCL-Anweisungen einfügen. Die strukturierten SCL-Anweisungen mit ihren entsprechenden exportierten XML-Tags und Attributen sind unten angegeben.

Globaler Zugriff

In SCL-Anweisungen stehen Variablen und Konstanten für globalen Zugriff in Anführungszeichen. Die Kommentare zwischen den Variablen und Adresssteilen werden durch das LineComment XML-Tag dargestellt.

SCL-Baustein	XML-Tag
"Datenbaustein_1".(*Kommentare 1*) Statisch_1(*Kommentare 2*).Statisch_2	<Access Scope="GlobalVariable"> <Symbol> <Component Name="Data_block_1" /> <Token Text="." /> <LineComment Inserted="True"> <Text>comment 1</Text> </LineComment> <Component Name="Static_1" /> </Symbol> > <LineComment Inserted="True"> <Text>comment 2</Text> </LineComment> <Token Text="." /> <Component Name="Static_2" /> > </Symbol> </Access>
"Data_block_1".Static_1 := 10	Format von XML nach Export mit Einstellung ExportOptions.None <Access Scope="GlobalVariable"> <Symbol> <Component Name="Data_block_1" /> <Token Text="." /> <Component Name="Static_1" /> </Symbol> </Access>
	Format von XML nach Export mit Einstellung ExportOptions.ReadOnly <Access Scope="GlobalVariable"> <Symbol> <Component Name="Data_block_1" /> <Token Text="." /> <Component Name="Static_1" /> </Symbol> <Address Area="DB" Type="Word" BlockNumber="1" BitOffset="0" Informative="true" /> </Symbol> </Access>

Verwendung von Anführungszeichen und

Die in der ersten Ebene verwendeten Anführungszeichen beschreiben die Art der Variable und dienen als Escape-Sequenzen für Sonderzeichen in SCL-Anweisungen. Wenn Anführungszeichen in der ersten Ebene verwendet werden, definieren sie die Variable als globale Variable. Wenn die Anführungszeichen nach # verwendet werden, stellen sie die Escape-Sequenz von Sonderzeichen wie # und von Leerzeichen dar.

- Zum Darstellen der unterschiedlichen Verwendung dient in der XML-Datei das Tag BooleanAttributes mit dem Attribut Name. Der Name enthält Werte wie HasQuotes und HasHash.
- Um im Scope-Attribut die Struktur zu definieren, wird # festgelegt.
- Diese Werte gelten nur für SCL.
- Die Standardwerte für diese Tags sind anfänglich FALSE, aber die Werte werden auch mit der Einstellung ExportOptions.WithDefaults nie exportiert.

SCL-Baustein	XML-Tag
"a".#b."c".#"d"	<pre><Access Scope="GlobalVariable"> <Symbol> <Component Name="a" /> <Token Text="." /> <Component Name="b"> <BooleanAttribute Name="HasHash">TRUE</ BooleanAttribute> </Component> <Token Text="." /> <Component Name="c"> <BooleanAttribute Name="HasQuotes">TRUE</ BooleanAttribute> </Component> <Token Text="." /> <Component Name="d"> <BooleanAttribute Name="HasQuotes">TRUE</ BooleanAttribute> <BooleanAttribute Name="HasHash">TRUE</ BooleanAttribute> </Component> </Symbol> <Access /></pre>

Array

SCL erlaubt das Einfügen von Kommentaren in die Arrayindizes um "[" und "]". Zum Kennzeichnen des Vorhandenseins eines Arrays dient in der XML-Datei das Attribut AccessModifier im Tag Component.

- Wenn Accessmodifier den Wert Array enthält, ist ein untergeordnetes Tag Access obligatorisch, um die Indexvariable des Arrays anzuzeigen.
- Der Standardwert für AccessModifier ist None.

SCL-Baustein	XML-Tag
#a.b[#i+#j,#k+#l].c	<Access Scope="LocalVariable"> <Symbol> <Component Name="a" /> <Token Text="." /> <Component Name="b" AccessModifier="Array" /> <Token Text="[" /> <Access Scope=LocalVariable> <Symbol> <Component Name="i" /> </Symbol> </Access> <Token Text="+" /> <Access Scope=LocalVariable> <Symbol> <Component Name="j" /> </Symbol> </Access> <Token Text="," /> <Access Scope=LocalVariable> <Symbol> <Component Name="k" /> </Symbol> </Access> <Token Text="+" /> <Access Scope=LocalVariable> <Symbol> <Component Name="l" /> </Symbol> </Access> <Token Text="]" /> </Component> <Token Text="." /> <Component Name="c" /> </Symbol> </Access>

Absoluter Zugriff

SCL ermöglicht verschiedene Zugriffsarten wie Absolut, Absolut-Offset, Gemischt (Datenbank und Membervariable), Slice, Peripher und Direkt. Der absolute Zugriff wird in XML durch das Tag Address dargestellt.

- Das Zeichen % in der DB wird in XML nicht geschrieben. Es wird beim Import automatisch erzeugt.
- Zwischen den Teilen der Adresse sind Leerzeichen zulässig.

SCL-Baustein	XML-Tag
%DB20 . DBW10	<pre><Access Scope="Address"> <Symbol> <Address Area="DB" BlockNumber="20" /> <Blank /> <Token Text="." /> <Blank /> <Address Area="DB" BitOffset="80" Type="Word"/> </Symbol> </Access></pre>
%DB20.DBX10.3 := true;	<p>Folgende XML gilt für alle Sprachen außer SCL.</p> <pre><Access Scope="Address"> <Address Area="DB" BlockNumber="20" BitOffset="83" Type="Bool" /> </Access></pre> <p>Folgende XML gilt für SCL.</p> <pre><Access Scope="Address"> <Symbol> <Address Area="DB" BlockNumber="20" /> <Token Text="." /> <Address Area="DB" BitOffset="83" Type="Bool"/> </Symbol> </Access></pre>

Zugriffsart "Absolut-Offset"

In AWL stellt das Tag AbsoluteOffset die Zugriffsart "Absolut-Offset" dar. In SCL wird das Tag Address für den absoluten Zugriff verwendet.

SCL-Baustein	XML-Tag
#Input_DB_ANY.%DBX2.3 := TRUE;	<Access Scope="LocalVariable"> <Symbol> <Component Name="Input_DB_ANY" /> </Symbol> <Token Name="." /> <Address BitOffset="19" Type="Bool" /> </Access>

Zugriffsart "Slice"

In SCL wird das Attribut SliceAccessModifier nicht unterstützt, und die Zugriffsart "Slice" wird durch das Tag Token dargestellt.

SCL-Baustein	XML-Tag
"tag_1" (*1*) . (*2*) member (*3*) . (*4*) %x1	<Access Scope="GlobalVariable"> <Symbol> <Component Name="tag_1" /> <LineComment Inserted="True"> <Text>1</Text> </LineComment> <Token Text="." /> <LineComment Inserted="True"> <Text>2</Text> </LineComment> <Component Name="member"/> <LineComment Inserted="True"> <Text>3</Text> </LineComment> <Token Text="." /> <LineComment Inserted="True"> <Text>4</Text> </LineComment> <Token Text="%x1" /> </Symbol> </Access>

Peripherer Zugriff

Der periphere Zugriff wird durch das Tag Token dargestellt.

SCL-Baustein	XML-Tag
"tag_1" (*1*) . (*2*) member:P	<Access Scope="GlobalVariable"> <Symbol> <Component Name="tag_1" /> <LineComment Inserted="True"> <Text>1</Text> </LineComment> <Token Text="." /> <LineComment Inserted="True"> <Text>2</Text> </LineComment> <Component Name="member"/> <Token Text=":P" /> </Symbol> </Access>

Zugriffsart "Direkt"

Die Anweisungen TypeOf und TypeOfDB werden entweder mit systemdefiniertem oder benutzerdefiniertem Typ ausgeführt. Angegeben sind die Typen im Tag Access mit dem Attribut Scope und den Werten SystemType und UserType.

SCL-Baustein	XML-Tag
Beispiel für systemdefinierten Typ if TypeOf(#inVariant) = TO_SpeedAxis then ... end_if	<Token text="=" /> </Blank> <Access Scope="SystemType"> <DataType>TO_SpeedAxis</DataType> </Access>
Beispiel für benutzerdefinierten Typ if TypeOf(#inVariant) = "aUserDefinedType" then ... end_if	<Token text="=" /> </Blank> <Access Scope="UserType"> <DataType>aUserDefinedType</DataType> </Access>

8.5.1.7 Export/Import von SCL-Aufrufbausteinen

SCL-Aufrufbausteine mit XML-Tags für den Export

SCL-Aufrufparameter werden in XML durch das Tag Parameter dargestellt. Das Attribut informative dient zum Darstellen der nicht zugewiesenen Parameter und Rückgabewerte wie Zeitstempel, Merkerinformationen usw. Das XML-Format folgt der gleichen willkürlichen Reihenfolge wie im SCL-Baustein.

Nachstehend ein Beispiel für einen Baustinaufruf:

8.5 Daten eines PLC-Geräts importieren/exportieren

SCL-Baustein	XML-Tag
#Callee_Instance(Input_1 := 5);	<p>Format von XML nach Export mit Einstellung ExportOptions.None</p> <pre><Access Scope="Call"> <CallInfo BlockType="FB"> <Instance Scope="LocalVariable"> <Component Name="Callee_Instance" /> </Instance> <Token text="("> <Parameter Name="Input_1"> <Blank /> <Token text=":=" /> <Blank /> <Access Scope="LiteralConstant"> <Constant> <ConstantType>Int</ ConstantType> <ConstantValue>5</ ConstantValue> </Constant> </Access> </Parameter> <Token text=")"> </CallInfo> </Access> <Token text=";" /></pre>
	<p>Format von XML nach Export mit Einstellung ExportOptions.ReadOnly</p> <pre><Access Scope="Call"> <CallInfo BlockType="FB"> <IntegerAttribute Name="BlockNumber" Informative="true">1</ IntegerAttribute> <DateAttribute Name="ParameterModifiedTS" Informative="true">2016-10-24T08:27: 34</DateAttribute> <Instance Scope="LocalVariable"> <Component Name="Callee_Instance" /> </Instance> <Token text="("> <Parameter Name="Input_1"> <StringAttribute Name="InterfaceFlags" Informative="true">S7_Visible</ StringAttribute> <Blank /> <Token text=":=" /></pre>

SCL-Baustein	XML-Tag
	<Blank /> <Access Scope="LiteralConstant"> <Constant> <ConstantType>Int</ConstantType> <ConstantValue>5</ConstantValue> </Constant> </Access> </Parameter> <Token text=")" /> </CallInfo> </Access> <Token text=";" />

Beispiel für unverbundene Parameter

Der FB hat vier Parameter, wobei a, b, c sowie d. b und d nicht verbunden sind.

SCL-Baustein	XML-Tag
"Block_4_DB" (a:=TRUE,c:=TRUE);	<pre> <Access Scope="Call"> <CallInfo Name="Block_4" BlockType="FB"> <Instance Scope="GlobalVariable"> <Component Name="Block_4_DB" /> </Instance> <Token text="(" /> <Parameter Name="a"> <Token text=":=" /> <Access Scope="LiteralConstant"> <Constant> <ConstantType>Bool</ ConstantType> <ConstantValue>TRUE</ ConstantValue> </Constant> </Access> </Parameter> <Token text="," /> <Parameter Name="b" Informative="true"/> <Parameter Name="c" > <Token text=":=" /> <Access Scope="LiteralConstant"> <Constant> <ConstantType>Bool</ ConstantType> <ConstantValue>True</ ConstantValue> </Constant> </Access> </Parameter> <Parameter Name="d" Informative="true"/> <Token text=")" /> </CallInfo> </Access></pre>

Beispiel für "ein Parameter"

Der SCL-Baustein ermöglicht Ihnen das Weglassen des Parameternamens. Dieser Parameter wird durch das Tag NamelessParameter dargestellt. Das Tag NamelessParameter hat keine Attribute und gilt nur für SCL.

SCL-Baustein	XML-Tag
"Block_4_DB" (TRUE);	<Access Scope="Call"> <CallInfo Name="Block_4" BlockType="FB"> <Instance Scope="GlobalVariable"> <Component Name="Block_4_DB" /> </Instance> <Token text="("> <NamelessParameter> <Access Scope="LiteralConstant"> <Constant <ConstantType>Bool</ConstantType> <ConstantValue>TRUE</ConstantValue> </Constant> </Access> </NamelessParameter> <Token text=")" /> </CallInfo> </Access>

Ausdruck als Aktualparameter

SCL-Baustein	XML-Tag
#Callee_Instance(Input_1 := #a+3);	<Access Scope="Call"> <CallInfo BlockType="FB"> <Instance Scope="LocalVariable"> <Component Name="Callee_Instance" /> </Instance> <Token text="(" /> <Parameter Name="Input_1"> <Blank /> <Token text=":=" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="a" /> </Symbol > </Access> <Token text="+" /> <Access Scope="LiteralConstant"> <Constant> <ConstantType>Int</ ConstantType> <ConstantValue>3</ ConstantValue> </Constant> </Access> </Parameter> <Token text=")" /> </CallInfo> </Access> <Token text=";" />

Ausdruck als Aktualparameter ohne Formalparameter

SCL-Baustein	XML-Tag
#Callee_Instance (#a+3);	<Access Scope="Call"> <CallInfo BlockType="FB"> <Instance Scope="LocalVariable"> <Component Name="Callee_Instance" /> </Instance> <Token text="(" /> <NamelessParameter> <Access Scope="LocalVariable"> <Symbol> <Component Name="a" /> </Symbol> </Access> <Token text="+" /> <Access Scope="LiteralConstant"> <Constant> <ConstantType>Int</ ConstantType> <ConstantValue>3</ ConstantValue> </Constant> </Access> </NamelessParameter> <Token text=")" /> </CallInfo> </Access> <Token text=";" />

Funktionsaufruf

SCL-Baustein	XML-Tag
#myInt := "MyFunction"(Param_1 := 1, Param_2 := 15, Param_3 := TRUE);	<Access Scope="LocalVariable"> <Symbol> <Component Name="myInt" /> </Symbol> </Access> <Blank /> <Token text=":=" /> <Blank /> <Access Scope="Call"> <CallInfo Name="MyFunction" BlockType="FC"> <Token text="(" /> <Parameter Name="Param_1"> ... </Parameter> </CallInfo> </Access> <Token text=";" />

Absoluter Aufruf

In SCL kann der Aufruf über die absolute Adresse des DB initiiert werden. Wegen der absoluten Adresse ist das Attribut Name des Knotens CallInfo leer.

Eine wiederherstellbare Ausnahme wird durch den Import ausgelöst, wenn

- Ein Knoten "Address" mit gültigem Wert des Attributs Name verfügbar ist.
- Der Knoten "Address" nicht vorhanden ist und kein gültiger Wert des Attributs Name vorhanden ist.

SCL-Baustein	XML-Tag
%DB20 (...);	<Access Scope="Call"> <CallInfo Name="" BlockType="FB"> <Instance Scope="GlobalVariable"> <Address Area="DB" BlockNumber="20" /> </Instance> <Token text="(" /> <Parameter> ... </Parameter> <Token text=")" /> </CallInfo> </Access>

Anweisung

Die Anweisung im SCL-Baustein wird während des Importvorgangs in der Systembibliothek geprüft, und die Anweisungsversionen werden beim Exportvorgang nicht exportiert.

Der allgemeine Anweisungstyp ist unten angegeben.

8.5 Daten eines PLC-Geräts importieren/exportieren

SCL-Baustein	XML-Tag
#myInt := ATTACH(OB_NR := 1, EVENT := 15, ADD := TRUE);	<p>Format von XML nach Export mit Einstellung ExportOptions.ReadOnly</p> <pre> <Access Scope="LocalVariable"> <Symbol> <Component Name="myInt" /> </Symbol> </Access> <Blank /> <Token text=":=" /> <Blank /> <Access Scope="Call"> <Instruction Name="ATTACH"> <Token text="("> <Parameter Name="OB_NR"> <Blank /> <Token text=":=" /> <Blank /> <Access Scope="LiteralConstant"> <Constant> <ConstantType>OB_ATT</ ConstantType> <ConstantValue>1</ ConstantValue> </Constant> </Access> </Parameter> <Token text="," /> <Blank /> <Parameter Name="EVENT"> <Blank /> <Token text=":=" /> <Blank /> <Access Scope="LiteralConstant"> <Constant> <ConstantType>EVENT_ATT</ ConstantType> <ConstantValue>15</ ConstantValue> </Constant> </Access> </Parameter> <Token text="," /> <Blank /> <Parameter Name="ADD"> <Blank /> <Token text=":=" /> <Blank /> <Access Scope="LiteralConstant"> <Constant> <ConstantType>Bool</ ConstantType> <ConstantValue>TRUE</ ConstantValue> </Access> </Parameter> </Instruction> </Call> </pre>

SCL-Baustein	XML-Tag
	</Constant> </Access> </Parameter> <Parameter Name="RET_VAL" Informative="true" /> <Token text=")" /> </Instruction> </Access> <Token text=";" />

Anweisung mit Vorlage

Wenn der Vorlagenparameter den Anweisungsnamen ergänzt, ist der Export des Vorlagenparameters notwendig. Wenn ein "TemplateValue"-Tag mit Attribut Type="Type" auf das Instruction-Tag folgt, verkettet der Importvorgang den Vorlagenwert mit dem Anweisungsnamen.

SCL-Baustein	XML-Tag
"tag_4" := MIN_DINT(IN1:="Tag_1", IN2:="Tag_2", IN3:="Tag_3");	<Access Scope="GlobalVariable"> <Symbol> <Component Name="Tag_4" /> </Symbol> </Access> ... <Access Scope="Call"> <Instruction Name="MIN"> <TemplateValue Name="value_type" Type="Type">DInt</ TemplateValue> ... <Parameter Name="IN1"> ... <Access Scope="GlobalVariable"> <Symbol> <Component Name="Tag_1" /> </Symbol> </Access> </Parameter> ... <Parameter Name="IN2">... <Access Scope="GlobalVariable"> <Symbol> <Component Name="Tag_2" /> </Symbol> </Access> </Parameter> ... <Parameter Name="IN3"> ... <Access Scope="GlobalVariable"> <Symbol> <Component Name="Tag_3" /> </Symbol> </Access> </Parameter> ... </Instruction> </Access> ...

Umwandlung

Bei Umwandlungsfunktionen werden der reale Anweisungsname und seine Vorlage nicht exportiert. Stattdessen wird der im SCL-Baustein verwendete Name exportiert.

SCL-Baustein	XML-Tag
#output_1 := TIME_TO_S5TIME(#input_1);	<Access Scope="LocalVariable"> <Symbol> <Component Name="output_1" /> </Symbol> </Access> ... <Access Scope="Call"> <Instruction Name="TIME_TO_S5TIME"> <Token text="(" /> <NamelessParameter> <Access Scope="LocalVariable"> <Symbol> <Component Name="input_1" /> </Symbol> </Access> </NamelessParameter> <Token text=")" /> </Instruction> </Access> ...

Anweisung mit Instanz

Instanz und Anweisung werden durch Leerzeichen getrennt. Leerzeichen sind optional, und sie können durch neue Zeilen und Kommentare dargestellt werden. Die Anweisung TON wird durch das Attribut Name des Tags Instruction dargestellt.

8.5 Daten eines PLC-Geräts importieren/exportieren

SCL-Baustein	XML-Tag
IEC_Timer_0_DB . TON (IN:="Tag_1", PT:="Tag_2");	<Access Scope="GlobalAccess"> <Symbol> <Component Name="IEC_Timer_0_DB" /> </Symbol > </Access> <Blank /> <Token text="." /> <Blank /> <Access Scope="Call"> <Instruction Name="TON"> <Blank /> <Token text="(" /> <Parameter Name="IN"> <Access Scope="GlobalVariable"> <Symbol> <Component Name="Tag_1" /> </Symbol> </Access> </Parameter> <...> <Token text=")" /> </Instruction> </Access> <Token text=";" />

Alarmkonstante

Die Alarmkonstanten werden nur in S7 400-PLCs verwendet, und der exportierte XML-Code hat Ähnlichkeit mit anderen Sprachen.

SCL-Baustein	XML-Tag
"Block_1_DB"(16#0000_0001);	<p>Format von XML nach Export mit Einstellung ExportOptions.None</p> <pre> <Access Scope="Call"> <CallInfo Name="Block_1" BlockType="FB"> <Instance Scope="GlobalVariable"> <Component Name="Block_1_DB" /> </Instance> <NamelessParameter> <Access Scope="AlarmConstant"> <Constant> <ConstantType>C_Alarm_8</ ConstantType> <ConstantValue>16#0000_0001</ ConstantValue> </Constant> </Access> </NamelessParameter > </CallInfo> </Access></pre>
	<p>Format von XML nach Export mit Einstellung ExportOptions.ReadOnly</p> <pre> <Access Scope="Call"> <CallInfo Name="Block_1" BlockType="FB"> <Instance Scope="GlobalVariable"> <Component Name="Block_1_DB" /> </Instance> <NamelessParameter> <Access Scope="AlarmConstant" > <Constant> <ConstantValue>16#00000001</ ConstantValue> <ConstantType>C_Alarm</ ConstantType> <StringAttribute Name="Format" Informative="true">Hex</ StringAttribute> </Constant> </Access> </NamelessParameter> </CallInfo> </Access></pre>

ENO (Freigabeausgang)

Um den Freigabeausgang ENO im SCL-Baustein zu unterstützen, wird im Tag "Access" das Attribut "Scope" mit dem Wert "PredefinedVariable" verwendet. Es enthält auch das Tag "PredefinedVariable" als untergeordnetes Element des Tags Access.

- Das Tag "PredefinedVariable" hat ein obligatorisches Attribut "Name".
- Der Umfang "PredefinedVariable" und das Tag "PredefinedVariable" sind nur für SCL zulässig.

SCL-Baustein	XML-Tag
Call(..., ENO => ENO);	<pre> <Access Scope="Call"> <CallInfo BlockType="FC"> <Token text="("> ... <Token text="," /> <Blank /> <Parameter Name="ENO"> <Blank /> <Token text="=>" /> <Blank /> <Access Scope="PredefinedVariable"> <PredefinedVariable Name="ENO" /> </Access> </Parameter> <Token text=")" /> </CallInfo> </Access> <Token text=";" /></pre>
IF ENO = #c THEN ...	<pre> <Token text="IF" /> <Blank /> <Access Scope="PredefinedVariable"> <PredefinedVariable Name="ENO" /> </Access> <Blank /> <Token Text="=" /> <Blank /> <Access Scope="LocalVariable"> <Symbol> <Component Name="c" /> </Symbol> </Access> <Blank /> <Token Text="THEN" /></pre>

8.5.1.8 Mehrsprachige Kommentare in SCL exportieren/importieren

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

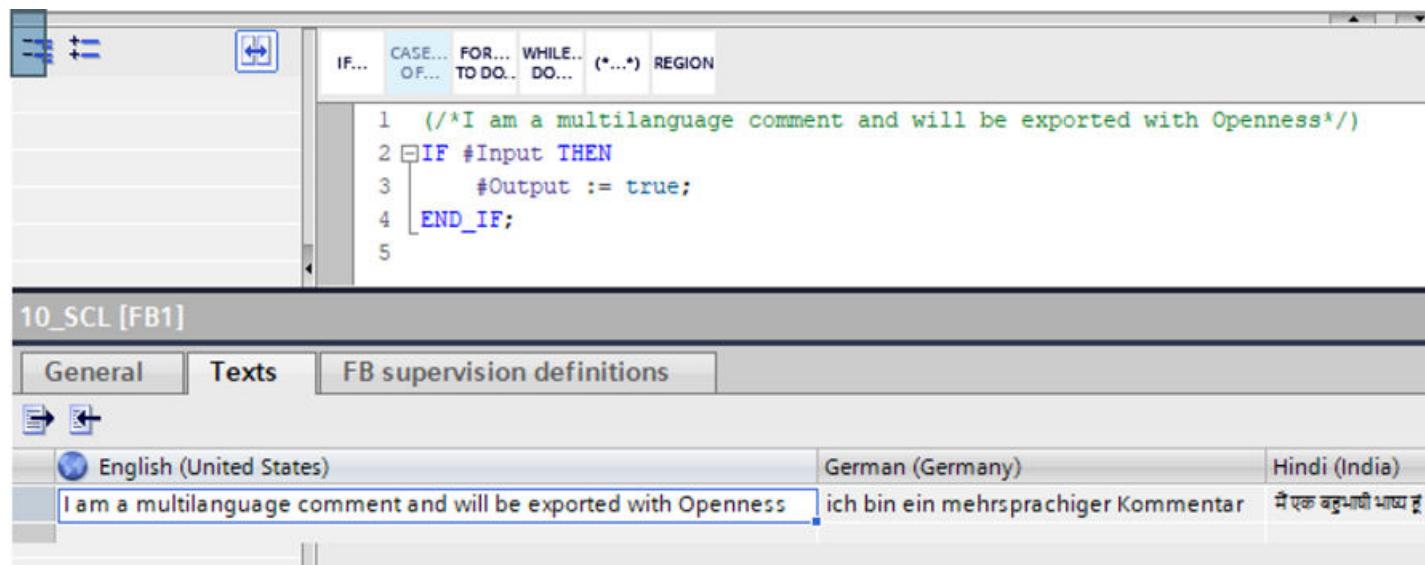
Anwendung

In TIA Portal Openness können mehrsprachige Kommentare im SCL-Editor auf eine der folgenden Arten importiert und exportiert werden:

- Beim Export von SCL in Openness werden alle mehrsprachigen Kommentare mit Übersetzung entsprechend den aktivierten Projektsprachen in die gleiche Openness xml-Datei exportiert.
- Beim Import von Openness in SCL werden alle mehrsprachigen Kommentare mit Übersetzung entsprechend den aktivierten Projektsprachen aus der Openness xml-Datei zurückimportiert. Dies muss im Textbereich für das Projekt angezeigt werden. Die Sprache wird aktiviert, wenn sie während des Imports im Projekt nicht aktiviert war.
- Nach dem Import ist zu prüfen, ob alle mehrsprachigen Kommentare im SCL-Editor mit den entsprechenden Kommentaren im Projekttext verknüpft sind.

Beispiel

Die folgende Abbildung zeigt ein Beispiel eines mehrsprachigen Kommentars im SCL-Editor:



The screenshot shows the TIA Portal SCL Editor interface. The code editor displays the following SCL code:

```

1 /*I am a multilanguage comment and will be exported with Openness*/
2 IF #Input THEN
3   #Output := true;
4 END_IF;
5

```

The code editor has tabs for IF..., CASE..., FOR..., WHILE..., DO..., (*...*), and REGION. The window title is "10_SCL [FB1]". Below the code editor, there is a tab bar with "General", "Texts", and "FB supervision definitions". The "Texts" tab is selected. It shows three language entries: English (United States), German (Germany), and Hindi (India). The English entry contains the text "I am a multilanguage comment and will be exported with Openness". The German entry contains the text "ich bin ein mehrsprachiger Kommentar". The Hindi entry contains the text "मैं एक बहुभाषी भाष्य हूँ".

Die folgende Abbildung zeigt ein Beispiel eines Exports eines mehrsprachigen Kommentars in die Openness XML-Datei:

8.5 Daten eines PLC-Geräts importieren/exportieren

```
<Comment Inserted="true" UIId="21">
  <MultiLanguageText Lang="en-US">I am a multilanguage comment and will be exported with Openness</MultiLanguageText>
  <MultiLanguageText Lang="de-DE">ich bin ein mehrsprachiger Kommentar</MultiLanguageText>
  <MultiLanguageText Lang="hi-IN">मैं एक बहुभाषी भाष्य हूँ</MultiLanguageText>
</Comment>
```

Siehe auch

- [Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- [Projekt öffnen \(Seite 111\)](#)

8.5.1.9 F-Bausteine exportieren

Fehlersichere Bausteine exportieren

Es ist jetzt möglich, die fehlersicheren Bausteine zu importieren und zu exportieren, doch F-Systembausteine können nicht exportiert werden.

Fehlersichere Bausteine importieren

Konsistente F-Bausteine können exportiert und wieder importiert werden. Sie werden als F-Bausteine erstellt.

Sie werden als Standardbausteine importiert, wenn Sie das Präfix "F_" aus dem Wert sämtlicher Attribute "ProgrammingLanguage" entfernen.

Siehe auch

- [Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- [Projekt öffnen \(Seite 111\)](#)
- [Bausteine exportieren \(Seite 898\)](#)
- [Bausteine mit Know-how-Schutz exportieren \(Seite 855\)](#)

8.5.1.10 System-Bausteine exportieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Das Projekt enthält einen Systembaustein.

- Der Systembaustein ist kein F-Baustein.
- Die PLC ist nicht online.

Verwendung

Nur sichtbare Systembausteine sind in der Baustein zusammensetzung verfügbar, also keine SFBs und SFCs. Die resultierende XML-Datei ähnelt der Exportdatei eines Bausteins.

Programmcode

Um die sichtbaren Daten eines Bausteins in eine XML-Datei zu exportieren, ändern Sie folgenden Programmcode:

```
//Exports system blocks
private static void ExportSystemBlocks(PlcSoftware plcsoftware)
{
    PlcSystemBlockGroup sbSystemGroup = plcsoftware.BlockGroup.SystemBlockGroups[0];
    foreach (PlcSystemBlockGroup group in sbSystemGroup.Groups)
    {
        foreach (PlcBlock block in group.Blocks)
        {
            block.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml", block.Name)), 
ExportOptions.WithDefaults);
        }
    }
}
```

8.5.1.11 GRAPH-Bausteine mit mehrsprachigem Text exportieren

XML-Struktur von GRAPH-Bausteinen mit mehrsprachigem Text

Die Export-XML von GRAPH-Bausteinen enthält die übersetzten Schrittnamen und Transitionnamen des Graphen. Diese übersetzten mehrsprachigen Texte werden jeweils unter dem übergeordneten Element Step und Transition als Elemente StepName und TransitionName dargestellt. Diese Elemente enthalten für jede unterstützte Sprache ein Element MultiLanguageText. Die Texte für die Sprachen, die nicht explizit eingestellt sind, werden nicht exportiert. Wenn keine Übersetzung vorhanden ist, werden die Elemente StepName und TransitionName nicht exportiert. Die Elemente StepName und TransitionName sind optional. Die TIA Portal Openness XML-Importoperation löst bei den Graph-Versionen kleiner als V5.0 eine wiederherstellbare Ausnahme aus.

Beispiel für das Element StepName

```
<Steps>
  <Step Number="1" Init="true" Name="Step1" MaximumStepTime="T#10S" WarningTime="T#7S">
    <StepName>
      <MultiLanguageText Lang="de-DE">stepDE</MultiLanguageText>
      <MultiLanguageText Lang="en-US">stepEN</MultiLanguageText>
      <MultiLanguageText Lang="it-CH">stepIT</MultiLanguageText>
    </StepName>
    ..
  </Step>
..
</Steps>
```

Beispiel für das Element TransitionName

```
<Transitions>
  <Transition IsMissing="false" Name="Trans1" Number="1" ProgrammingLanguage="LAD">
    <TransitionName>
      <MultiLanguageText Lang="de-DE">transDE</MultiLanguageText>
      <MultiLanguageText Lang="en-US">transEN</MultiLanguageText>
      <MultiLanguageText Lang="it-CH">transIT</MultiLanguageText>
    </TransitionName>
    ..
  </Transition>
..
</Transitions>
```

8.5.1.12 Baustein importieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Die PLC ist nicht online.

Verwendung

Die TIA Portal Openness API unterstützt den Import von Bausteinen mit den Programmiersprachen "AWL", "FUP", "GRAPH", "SCL" oder "KOP" aus einer XML-Datei. Die folgenden Bausteintypen werden unterstützt:

- Funktionsbausteine (FB)
- Funktionen (FC)
- Organisationsbausteine (OB)
- Globale Datenbausteine (DB)

Hinweis

Importieren optimierter Datenbausteine

Optimierte Datenbausteine werden nur von CPUs ab S7-1200 unterstützt. Wenn Sie optimierte Datenbausteine in S7-300 oder S7-400 importieren, wird eine Ausnahme ausgelöst und der Import schlägt fehl.

Reaktion auf den Import

Beim Importieren eines Bausteins gelten die folgenden Regeln:

- Die XML-Datei kann weniger Daten enthalten als der Baustein im Projekt, z. B. weniger Parameter.
- Redundante Informationen wie Aufrufinformationen müssen im Projekt und in der XML-Datei identisch sein. Andernfalls wird eine Ausnahme ausgelöst.
- Die Daten in der XML-Datei dürfen im Hinblick auf ihre Übersetzungsfähigkeit im TIA Portal „inkonsistent“ sein.
- Attribute mit den Attributen "ReadOnly=True" und "Informative=True" werden nicht importiert.
- Fehlende Instanz-DBs werden nicht automatisch erstellt.
- Wenn in der XML-Datei keine Bausteinnummer angegeben ist, wird die Bausteinnummer automatisch zugeordnet.
- Wenn der Baustein im Projekt nicht vorhanden ist und in der XML-Datei keine Versionsinformationen angegeben sind, wird die Version "0.1" zugeordnet.

8.5 Daten eines PLC-Geräts importieren/exportieren

Programmcode

Ändern Sie folgenden Programmcode:

```
//Import blocks
private static void ImportBlocks(PlcSoftware plcSoftware)
{
    PlcBlockGroup blockGroup = plcSoftware.BlockGroup;
    IList<PlcBlock> blocks = blockGroup.Blocks.Import(new FileInfo(@"D:\Blocks
\myBlock.xml"), ImportOptions.Override);
}
```

Ändern Sie folgenden Programmcode:

```
//Import system blocks
private static void ImportSystemBlocks(PlcSoftware plcSoftware)
{
    PlcBlockSystemGroup systemblockGroup = plcSoftware.BlockGroup;
    IList<PlcBlock> blocks = systemblockGroup.Blocks.Import(new FileInfo(@"D:\Blocks
\myBlock.xml"), ImportOptions.Override);
}
```

8.5.1.13 Bausteine exportieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Die PLC ist nicht online.

Verwendung

Die API-Schnittstelle unterstützt das Exportieren konsistenter Bausteine und Anwenderdatentypen in eine XML-Datei.

Die XML-Datei erhält den Namen des Bausteins. Die folgenden Bausteintypen werden unterstützt:

- Funktionsbausteine (FB)
- Funktionen (FC)
- Organisationsbausteine (OB)
- Globale Datenbausteine (DB)

Die folgenden Programmiersprachen werden unterstützt:

- AWL
- FUP

- KOP
- GRAPH
- SCL

Für alle Bausteine geltende Attribute

Die folgenden Attribute werden mit den ausgewählten ExportOptions in alle Bausteine exportiert (siehe Export von Projektierungsdaten (Seite 778)). In Fettdruck dargestellten Attribute werden immer exportiert.

Weitere Informationen finden Sie im Informationssystem des TIA Portals unter „Übersicht über die Bausteinattribute“.

Attribut	Typ	Standardwert	Schreibgeschützt
AutoNumber	Bool	true	false
CodeModifiedDate	DateTime	-	true
CompileDate	DateTime	-	true
CreationDate	DateTime	-	true
HeaderAuthor	String	""	false
HeaderFamily	String	""	false
HeaderName	String	""	false
HeaderVersion	String	„0,1“	false
Interface	String	leere Schnittstelle	false
InterfaceModifiedDate	DateTime	-	true
IsConsistent	Bool	-	true
IsKnowHowProtected ¹	Bool	false	true
IsWriteProtected	Bool	false	true
MemoryLayout	enum MemoryLayout	-	false
ModifiedDate	DateTime	-	true
Name	String	-	false
Number	Int32	nächste verfügbare Nummer	false
ParameterModified	DateTime	-	true
PLCSimAdvancedSupport	Bool	false	true
ProgrammingLanguage	enum ProgrammingLanguage	-	false
StructureModified	DateTime	-	true

¹ Das Attribut IsKnowHowProtected gilt auch für UDT.

Hinweis

Es gibt bestimmte Bedingungen, unter denen das Attribut MemoryLayout schreibgeschützt ist.

8.5 Daten eines PLC-Geräts importieren/exportieren

Dynamisch zugängliche allgemeine Attribute

Die folgenden Attribute sind nur unter bestimmten Bedingungen schreibgeschützt:

Attribut	Zustand ReadOnly
AutoNumber	All KnowHowProtected blocks, All Classic OBs; Plus OBs: DiagnosticErrorInterrupt, IOAccessError, ProgrammingError, PullOrPlugOfModules, RackOrStationFailure, Status, TimeErrorInterrupt, Update
HeaderVersion	System- and KnowHowProtected DBs; ArrayDBs that originated from system library
HeaderName	
HeaderFamily	
HeaderAuthor	
Memory-Layout	Classic blocks, System Know how protected blocks, ArrayDBs, IDBofFBs, Graph blocks

Für ArrayDB-Bausteine geltende Attribute

Die folgenden Attribute werden mit den ausgewählten `ExportOptions`. für ArrayDB-Bausteine exportiert:

Attribut	Typ	Standardwert	Schreibgeschützt
ArrayDataType	String	-	true
ArrayLimitUpperBound	Int32	-	true

Für DB-Bausteine geltende Attribute

Die folgenden Attribute werden mit den ausgewählten `ExportOptions`. für DB-Bausteine exportiert:

Attribut	Typ	Standardwert	Schreibgeschützt
IsOnlyStoredInLoadMemory	Bool	false	false
IsPLCDB	Bool	false	false
IsWriteProtectedInAS	Bool	false	false

Für FB-Bausteine geltende Attribute

Die folgenden Attribute werden mit den ausgewählten `ExportOptions`. für FB-Bausteine exportiert:

Attribut	Typ	Standardwert	Schreibgeschützt
AssignedProDiagFB	String	-	-
ISMultInstanceCapable	Bool	-	true
Supervisions	String	no supervisions	true bei IDB of FB und false bei FB

Für DB- und FB-Bausteine geltende Attribute

Die folgenden Attribute werden mit den ausgewählten ExportOptions . für DB- und FB-Bausteine exportiert:

Attribut	Typ	Standardwert	Schreibgeschützt
IsIECCheckEnabled	Bool	false	false
IsRetainMemResEnabled ¹	Bool	false	false
MemoryReserve	Unsigned	0	false
RetainMemoryReserve ²	Unsigned	0	false

² Wenn der Wert des Attributs "IsRetainMemResEnabled" "false" ist und das Attribut "RetainMemoryReserve" ungleich "0" ist, wird eine Ausnahme ausgelöst.

Für FB-, DB- und IDB-Bausteine geltende Attribute

Die folgenden Attribute werden mit den ausgewählten ExportOptions . für FB-, DB- und IDB-Bausteine exportiert:

Attribut	Typ	Standardwert	Schreibgeschützt
DownloadWithoutReinit	Bool	false	true

Für FB- und FC-Bausteine geltende Attribute

Die folgenden Attribute werden mit den ausgewählten ExportOptions . für FB- und FC-Bausteine exportiert:

Attribut	Typ	Standardwert	Schreibgeschützt
LibraryType	String	-	true
LibraryTypeVersionGuid	String	-	true

Für FB- und FC-Bausteine (AWL) geltende Attribute

Die folgenden Attribute werden mit den ausgewählten ExportOptions . für FB- und FC-Bausteine (AWL) exportiert:

Attribut	Typ	Standardwert	Schreibgeschützt
ParameterPassing	Bool	false	false

Für FB, FC und Instanz-DB von FB-Bausteinen geltende Attribute

Die folgenden Attribute werden mit den ausgewählten ExportOptions . für FB, FC und Instanz-DB von FB-Bausteinen exportiert:

8.5 Daten eines PLC-Geräts importieren/exportieren

Attribut	Typ	Standardwert	Schreibgeschützt
UDABlockProperties	String	""	false
UDAEnableTagReadback	Bool	false	false

Für Instanz-DBs von FBs und UDTs geltende Attribute

Die folgenden Attribute werden mit den ausgewählten `ExportOptions`. für Instanz-DBs von FB- und UDT-Bausteinen exportiert:

Attribut	Typ	Standardwert	Schreibgeschützt
InstanceOfName	String	""	false
InstanceOfNumber	Unsigned Short	-	true
InstanceOfType	enum BlockType	-	true
OfSystemLibElement	String	""	false
OfSystemLibVersion	String	""	false

Für OB-Bausteine geltende Attribute

Die folgenden Attribute werden mit den ausgewählten `ExportOptions`. für OB-Bausteine für spezifische Plus PLCs exportiert:

Attribut	Typ	Standardwert	Schreibgeschützt
ApplicationCycle	Single	-	true
AutomaticMinimum	Bool	-	true
ConstantName	String	-	true
CycleTimeDistributedIO	Single	-	true
CyclicApplicationCycleTime	Single	-	true
CyclicTime	Int32	100000	true
DataExchangeMode	OBDataExchangeMode	Cyclic	true
DelayTime	Double	-	true
DistributedIOPName	String	-	true
DistributedIOPNumber	Int32	-	true
EnableTimeError	Bool	-	true
EventClass	String	-	true
EventsToBeQueued	Int32	-	true
EventThresholdForTimeError	Int32	-	true
Execution	OBExecution	Never	true
Factor	Single	-	true
PhaseOffset	Int32	0	true
PriorityNumber	Int32	-	true
ProcessImagePartNumber	UInt32	-	true

Attribut	Typ	Standardwert	Schreibgeschützt
ReportEvents	Bool	-	true
SecondaryType ³	String	-	false
StartDate	DateTime	1/1/2012	true
SynchronousApplicationCycleTime	Single	-	true
TimeMode	OBTIMemode	System	true
TimeOfDay	DateTime	12:00 AM	true
TransformationDBNumber	UInt16	0xffff	true

³ Beim Exportieren eines OB wird anhand der OB-Nummer zusätzlich der SecondaryType festgelegt. Die Zuordnung wird während des Importvorgangs geprüft. Ist die Zuordnung falsch, wird eine Ausnahme vom Typ „Recoverable“ ausgelöst.

Für FB-, FC- und OB-Bausteine geltende Attribute

Die folgenden Attribute werden mit den ausgewählten ExportOptions . für FB-, FC- und OB-Bausteine exportiert:

Attribut	Typ	Standardwert	Schreibgeschützt
HandleErrorsWithinBlock	Bool	false	true

Für FB-, FC- und UDT-Bausteine geltende Attribute

Die folgenden Attribute werden mit den ausgewählten ExportOptions . für FB-, FC- und UDT-Bausteine exportiert:

Attribut	Typ	Standardwert	Schreibgeschützt
LibraryConformanceStatus	String	-	false

Für GRAPH-Bausteine geltende Attribute

Die folgenden Attribute werden mit den ausgewählten ExportOptions . für GRAPH-Bausteine exportiert:

Attribut	Typ	Standardwert	Schreibgeschützt
AcknowledgeErrorsRequired	Bool	true	false
CreateMinimizedDB	Bool	false	false
ExtensionBlockName	String	-	-
GraphVersion	String	-	false
InitialValuesAcquisition	String	-	-
LanguageInNetworks	String	-	false
LockOperatingMode	Bool	false	false

8.5 Daten eines PLC-Geräts importieren/exportieren

Attribut	Typ	Standardwert	Schreibgeschützt
PermanentILProcessingIn-MANMode	Bool	false	false
SkipSteps	Bool	false	false

Für GRAPH FB-Bausteine geltende Attribute

Die folgenden Attribute werden mit den ausgewählten ExportOptions. für GRAPH FB-Bausteine exportiert:

Attribut	Typ	Standardwert	Schreibgeschützt
WithAlarmHandling	Bool	true	false

Für SCL-Bausteine geltende Attribute

Die folgenden Attribute werden mit den ausgewählten ExportOptions. für SCL-Bausteine exportiert. Diese Attribute werden basierend auf dem Typ der PLCs exportiert.

Attribut	Typ	Standardwert	Schreibgeschützt
CheckArrayLimits	Bool	false	false
ExtendedStatus	Bool	false	false
DBAccessibleFromOPCUA	Bool	true	false

Für GRAPH-, SCL- und KOP/FUP-Bausteine geltende Attribute

Die folgenden Attribute werden mit den ausgewählten ExportOptions. für GRAPH-, SCL- und KOP/FUP-Bausteine exportiert:

Attribut	Typ	Standardwert	Schreibgeschützt
SetENOAutomatically	Bool	-	false

Für Programmabausteine (außer OB), DB und UDT-Bausteine geltende Attribute

Die folgenden Attribute werden für Programmabausteine (außer OB), DB und UDT-Bausteine unter einer Unit mit den ausgewählten ExportOptions exportiert:

Attribut	Typ	Standardwert	Schreibgeschützt
Access	UnitAccessType	Unpublished	false

Importverhalten für das Attribut 'Access'

	Import unter Unit	Import nicht unter Unit (ohne SWImportOptions.IgnoreUnitAttributes)	Import nicht unter Unit (mit SWImportOptions.IgnoreUnitAttributes)
XML-Export unter Unit	'Access' verwendet und festgelegt	RecoverableException ausgelöst	'Access' ignoriert
XML-Export nicht unter Unit	'Access' erhält Standardwert	'Access' ist nicht vorhanden	'Access' ist nicht vorhanden

Programmcode

Um einen Baustein ohne Knowhow-Schutz in eine XML-Datei zu exportieren, ändern Sie folgenden Programmcode:

```
//Exports a regular block
private static void ExportRegularBlock(PlcSoftware plcSoftware)
{
    PlcBlock plcBlock = plcSoftware.BlockGroup.Blocks.Find("MyBlock");
    plcBlock.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml", plcBlock.Name)), ExportOptions.WithDefaults);
}
```

8.5.1.14 Bausteine/UDT mit offenem Verweis importieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Der PLC ist nicht online.

Anwendung

Mit der Openness API können Sie bei STEP 7-Objekten einen neuen Importmodus verwenden, bei dem Sie Bausteine und UDTs auch dann importieren können, wenn ein zugehöriges Objekt fehlt.

Die Openness-Schnittstelle unterstützt den neuen Importmodus unter den folgenden Bedingungen:

Import von	Objektverweis
UDT	UDT
DB (global)	UDT
IDBofUDT	UDT
IDBofFB	FB
ArrayDB	Array von UDT

8.5 Daten eines PLC-Geräts importieren/exportieren

Import von	Objektverweis
FB	UDT (Schnittstelle), Multiinstanz
FC	UDT (Schnittstelle)

Programmcode

Den neuen Modus können Sie dank einer neuen Überladung der entsprechenden Import-Methode verwenden. Die neue Überladung weist einen zusätzlichen Parameter auf, der einen Wert der neuen markierten Enumeration SWImportOptions akzeptiert. Um den Import zu ermöglichen, können Sie SWImportOptions.IgnoreMissingReferencedObject auch dann verwenden, wenn das Objekt des Verweises fehlt.

```
Flagged Enum SWImportOptions
{
    None = 0,
    IgnoreStructuralChanges = 1,
    IgnoreMissingReferencedObjects = 2
}
... // All kinds of blocks
PlcBlockComposition.Import(file, ImportOptions.None,
SWImportOptions.IgnoreMissingReferencedObject);
...
... // UDTs
PlcTypeComposition.Import(file, ImportOptions.None,
SWImportOptions.IgnoreMissingReferencedObject);
...
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

8.5.1.15 Bausteine/UDT für Strukturänderungsobjekte importieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe [Verbindung zum TIA Portal herstellen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
Siehe [Öffnen eines Projekts \(Seite 111\)](#)
- Der PLC ist nicht online.

Anwendung

Mit der Openness API können Sie Bausteine und UDTs auch dann importieren, wenn aufgrund einer Strukturänderung an zugehörigen Objekten Instanzdaten verloren gegangen sind.

Die Openness-Schnittstelle unterstützt den neuen Importmodus unter den folgenden Bedingungen:

Import von	Objektverweisen
Variable	UDT
UDT	UDT
DB (global)	UDT
IDBofUDT	UDT
IDBofFB	FB
ArrayDB	Array von UDT
FB	UDT (Schnittstelle), Multiinstanz
FC	UDT (Schnittstelle)

Programmcode

Den neuen Modus können Sie dank einer neuen Überladung der entsprechenden Import - Methode verwenden. Die neue Überladung weist einen zusätzlichen Parameter auf, der einen Wert der neuen markierten Enumeration SWImportOptions akzeptiert. Um den Import zu ermöglichen, auch wenn es strukturelle Änderungen gibt und es möglicherweise zu Datenverlust kommt, verwenden Sie "SWImportOptions.IgnoreStructuralChanges".

```
Flagged Enum SWImportOptions
{
    None = 0,
    IgnoreStructuralChanges = 1,
    IgnoreMissingReferencedObjects = 2
}
...
// All kinds of blocks
PlcBlockComposition.Import(file, ImportOptions.None,
    SWImportOptions.IgnoreStructuralChanges);
...
...
// UDTs
PlcTypeComposition.Import(file, ImportOptions.None,
    SWImportOptions.IgnoreStructuralChanges);
...
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

8.5.1.16 Export/Import des Unit-spezifischen Veröffentlichungsattributs von Bausteinen und Typen

Voraussetzung

- Die Anwendung TIA Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Das Attribut 'Access' ist nur für Bausteine, PLC-Typen und Variablenlisten vorhanden, die sich unter einer Unit befinden. Es ist daher in der exportierten XML-Datei enthalten. Die exportierte XML-Datei des gleichen Bausteins, des gleichen Typs oder der gleichen Variablenliste in einer Umgebung außerhalb einer Unit enthält das Attribut nicht.

Die allgemeinen XML-Importregeln von Openness lassen XML-Dateien mit unbekannten/undefined Attributen nicht zu. Aufgrund dieser Regel können exportierte XML-Dateien, die aus einem Objekt in einer Umgebung unterhalb einer Unit stammen, nicht in eine Umgebung außerhalb einer Unit importiert werden. Hieraus würde sich eine erhebliche Beschränkung der Einsetzbarkeit ergeben. Daher wird die folgende Erweiterung definiert.

Entsprechend der Überladung der Methode import () werden drei Parameter übergeben. Diese drei Parameter sind wie folgt:

Parameter	Rückgabetyp	Beschreibung
path	String	Gibt den Pfad zur SIMATIC ML-Datei an, die importiert werden soll
importOptions	enum: Siemens.Engineering.ImportOptions	Gibt die beim Import zu verwendende(n) allgemeine(n) Importoption(en) an.
swImportOptions	enum: Siemens.Engineering.SW.SWImportOptions	Gibt die für STEP 7 spezifische(n) Importoption(en) an.

Die Siemens.Engineering.SW.SWImportOptions vom Typ enum werden durch die folgende neue Importoption erweitert:

- IgnoreUnitAttributes: gibt an, dass der Importvorgang nicht abgebrochen werden soll, falls in der XML-Datei Unit-bezogene Attribute vorhanden sind und der Import in einer Umgebung außerhalb einer Unit stattfindet.

Das 'IgnoreUnitAttributes' wird nur berücksichtigt, wenn die XML-Datei

- aus einer Unit exportiert wird
- das Attribut 'Access' enthält
- in eine Umgebung außerhalb einer Unit importiert wird

Wenn die exportierte XML-Datei das Openness-Attribut 'Access' nicht enthält und/oder wenn sie in eine Unit importiert wird, so wird die neue Importoption von der Importlogik nicht berücksichtigt.

Programmcode:

```
PlcSoftware plcTarget = GetControllerTargetByPLCName(Session.OpnsProject.Devices, PLCName);
PlcUnitProvider plcUnitProvider = plcTarget.GetService<PlcUnitProvider>();
PlcSoftware plcSoftware = plcTarget.GetService<SoftwareContainer>() as PlcSoftware;
PlcUnit plcUnit1 = plcUnitProvider.UnitGroup.Units[0];
//assuming Unit_1 is already existing
PlcUnit plcUnit2 = plcUnitProvider.UnitGroup.Units[1];
//assuming Unit_2 is already existing
PlcBlock block1 = plcUnit1.BlockGroup.Blocks.Find("Block_1");
//assuming Block_1 is already existing under Unit_1
PlcBlock block2 = plcUnit2.BlockGroup.Blocks.Find("Block_2");
//assuming Block_2 is already existing under Unit_2
PlcBlock block3 = plcSoftware.BlockGroup.Blocks.Find("Block_3");
//assuming Block_3 is already existing under the PLC
```

Programmcode: Objekt wird aus einer Unit exportiert und in eine Unit importiert

Die folgenden Beispiele zeigen das Verhalten der neuen Importoption anhand von Bausteinen. Dasselbe Verhalten gilt jedoch auch für PLC-Typen und Variablenlisten.

Ändern Sie folgenden Programmcode, um einen Baustein zu exportieren, bei dem das Attribut 'Access' auf den Wert 'Unpublished' (Standardwert) gesetzt ist sowie die ExportOptions.WithDefaults und die SWImportOptions.None lauten:

```
block1.SetAttribute("Access", UnitAccessType.Unpublished);
block1.Export(new FileInfo("somepath"), ExportOptions.WithDefaults);
block1.Delete();
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,
SWImportOptions.None);
```

Ändern Sie folgenden Programmcode, um einen Baustein zu exportieren, bei dem das Attribut 'Access' auf den Wert 'Unpublished' (Standardwert) gesetzt ist sowie die ExportOptions.WithDefaults und die SWImportOptions.IgnoreUnitAttributes lauten:

```
block1.SetAttribute("Access", UnitAccessType.Unpublished);
block1.Export(new FileInfo("somepath"), ExportOptions.WithDefaults);
block1.Delete();
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,
SWImportOptions.IgnoreUnitAttributes);
```

8.5 Daten eines PLC-Geräts importieren/exportieren

Ändern Sie folgenden Programmcode, um einen Baustein zu exportieren, bei dem das Attribut 'Access' auf den Wert 'Published' gesetzt ist sowie die ExportOptions.None und die SWImportOptions.None lauten:

```
block1.SetAttribute("Access", UnitAccessType.Published);  
block1.Export(new FileInfo("somepath"), ExportOptions.None);  
block1.Delete();  
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,  
SWImportOptions.None);
```

Ändern Sie folgenden Programmcode, um einen Baustein zu exportieren, bei dem das Attribut 'Access' auf den Wert 'Published' gesetzt ist sowie die ExportOptions.None und die SWImportOptions.IgnoreUnitAttributes lauten:

```
block1.SetAttribute("Access", UnitAccessType.Published);  
block1.Export(new FileInfo("somepath"), ExportOptions.None);  
block1.Delete();  
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,  
SWImportOptions.IgnoreUnitAttributes);
```

Hinweis

In allen obigen Programmcode-Abschnitten mit Fehlerszenarios wird keine Ausnahme ausgelöst und der Importvorgang ist erfolgreich.

Programmcode: Objekt wird aus einer Unit exportiert und in eine Umgebung außerhalb einer Unit importiert

Ändern Sie folgenden Programmcode, um einen Baustein zu exportieren, bei dem das Attribut 'Access' auf den Wert 'Unpublished' (Standardwert) gesetzt ist sowie die ExportOptions.WithDefaults und die SWImportOptions.None lauten:

```
block1.SetAttribute("Access", UnitAccessType.Unpublished);  
block1.Export(new FileInfo("somepath"), ExportOptions.WithDefaults);  
block1.Delete();  
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,  
SWImportOptions.None);
```

Hinweis

Im obigen Programmcode ist der Importvorgang nicht erfolgreich und es wird eine wiederherstellbare Ausnahme ausgelöst.

Ändern Sie folgenden Programmcode, um einen Baustein zu exportieren, bei dem das Attribut 'Access' auf den Wert 'Unpublished' (Standardwert) gesetzt ist sowie die ExportOptions.WithDefaults und die SWImportOptions.IgnoreUnitAttributes lauten:

```
block1.SetAttribute("Access", UnitAccessType.Unpublished);
block1.Export(new FileInfo("somepath"), ExportOptions.WithDefaults);
block1.Delete();
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,
SWImportOptions.IgnoreUnitAttributes);
```

Hinweis

Im obigen Programmcode ist der Importvorgang erfolgreich und es wird keine Ausnahme ausgelöst.

Ändern Sie folgenden Programmcode, um einen Baustein zu exportieren, bei dem das Attribut 'Access' auf den Wert 'Published' gesetzt ist sowie die ExportOptions.None und die SWImportOptions.None lauten:

```
block1.SetAttribute("Access", UnitAccessType.Published);
block1.Export(new FileInfo("somepath"), ExportOptions.None);
block1.Delete();
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,
SWImportOptions.None);
```

Hinweis

Im obigen Programmcode ist der Importvorgang erfolgreich und es wird eine wiederherstellbare Ausnahme ausgelöst.

Ändern Sie folgenden Programmcode, um einen Baustein zu exportieren, bei dem das Attribut 'Access' auf den Wert 'Published' gesetzt ist sowie die ExportOptions.None und die Importoptionen SWImportOptions.IgnoreUnitAttributes lauten:

```
block1.SetAttribute("Access", UnitAccessType.Published);
block1.Export(new FileInfo("somepath"), ExportOptions.None);
block1.Delete();
plcUnit2.BlockGroup.Blocks.Import(new FileInfo("somepath"), ImportOptions.None,
SWImportOptions.IgnoreUnitAttributes);
```

Hinweis

Im allen obigen Programmcode-Abschnitten wird keine Ausnahme ausgelöst und der Importvorgang ist erfolgreich.

8.5 Daten eines PLC-Geräts importieren/exportieren

Objekt wird aus Umgebung außerhalb einer Unit exportiert und in eine Unit importiert

Die exportierte XML-Datei enthält das Openness-Attribut 'Access' nicht; beim Import erhält das Attribut den Standardwert 'Unpublished'.

Objekt wird aus einer Umgebung außerhalb einer Unit exportiert und in eine Umgebung außerhalb einer Unit importiert

Die exportierte XML-Datei enthält das Openness-Attribut 'Access' nicht; beim Import der Datei geschieht nichts.

Siehe auch

Verbindung zum TIA Portal aufbauen (Seite 79)

Projekt öffnen (Seite 111)

8.5.2 Technologieobjekte

8.5.2.1 S7-1200 Motion Control

8.5.2.2 S7-1500 Motion Control

8.5.2.3 PID-Regelung

8.5.2.4 Zählen

8.5.2.5 Easy Motion Control

8.5.3 VariablenTabellen

8.5.3.1 PLC-VariablenTabellen exportieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Verwendung

Pro PLC-Variablenliste wird eine XML-Datei exportiert.

Die Schnittstelle TIA Portal Openness API unterstützt den Export aller PLC-Variablenlisten aus der Systemgruppe und deren Untergruppen.

Programmcode

Ändern Sie den folgenden Programmcode, um alle PLC-Variablenlisten aus der Systemgruppe und deren Untergruppen zu exportieren:

```
private static void ExportAllTagTables(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    // Export all tables in the system group
    ExportTagTables(plcTagTableSystemGroup.TagTables);
    // Export the tables in underlying user groups
    foreach(PlcTagTableUserGroup userGroup in plcTagTableSystemGroup.Groups)
    {
        ExportUserGroupDeep(userGroup);
    }
}

private static void ExportTagTables(PlcTagTableComposition tagTables)
{
    foreach(PlcTagTable table in tagTables)
    {
        table.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml", table.Name)),
ExportOptions.WithDefaults);
    }
}

private static void ExportUserGroupDeep(PlcTagTableUserGroup group)
{
    ExportTagTables(group.TagTables);
    foreach(PlcTagTableUserGroup userGroup in group.Groups)
    {
        ExportUserGroupDeep(userGroup);
    }
}
```

Siehe auch

[Export von Projektierungsdaten \(Seite 778\)](#)

8.5 Daten eines PLC-Geräts importieren/exportieren

8.5.3.2 PLC-Variablentabelle importieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Programmcode

Ändern Sie den folgenden Programmcode, um PLC-Variablentabellen oder eine Ordnerstruktur mit PLC-Variablentabellen aus einer XML-Datei in die Systemgruppe oder eine benutzerdefinierte Gruppe zu importieren:

```
//Imports tag tables to the tag system group
private static void ImportTagTable(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    PlcTagTableComposition tagTables = plcTagTableSystemGroup.TagTables;
    tagTables.Import(new FileInfo(@"D:\Samples\myTagTable.xml"), ImportOptions.Override);
    // Or, to import into a subfolder:
    // plcTagTableSystemGroup.Groups.Find("SubGroup").TagTables.Import(new
    FileInfo(@"D:\Samples\myTagTable.xml"), ImportOptions.Override);
}
```

Siehe auch

Anmerkungen zur Leistung von TIA Portal Openness (Seite 45)

8.5.3.3 Einzelne Variable oder Konstante aus einer PLC-Variablentabelle exportieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Verwendung

Die API-Schnittstelle unterstützt den Export einer Variablen oder Konstanten aus einer PLC-Variablentabelle in eine XML-Datei. Achten Sie dabei darauf, dass die verwendeten Variablentabellennamen den Dateibenennungskonventionen Ihres Dateisystems entsprechen.

Der Kommentar einer Variablen oder Konstanten wird nur exportiert, wenn mindestens eine Sprache für den Kommentar festgelegt ist. Wenn der Kommentar nicht für alle Projektsprachen festgelegt ist, wird dieser Kommentar nur für die festgelegten Projektsprachen exportiert.

Hinweis**PLC-Systemkonstanten**

PLC-Systemkonstanten werden vom Export und Import ausgeschlossen.

Programmcode

Um eine bestimmte Variable oder Konstante aus einer PLC-Variabentabelle in eine XML-Datei zu exportieren, ändern Sie folgenden Programmcode:

```
//Exports a single tag or constant of a controller tag table
private static void ExportTag(PlcSoftware plcSoftware, string tagName)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    PlcTag tag = plcTagTableSystemGroup.TagTables[0].Tags.Find(tagName);
    if(tag == null) return;

    tag.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml", tag.Name)),
    ExportOptions.WithDefaults);
}

private static void ExportUserConstant(PlcSoftware plcSoftware, string userConstantName)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    PlcUserConstant plcConstant =
plcTagTableSystemGroup.TagTables[0].UserConstants.Find(userConstantName);
    if(plcConstant== null) return;

    plcConstant.Export(new FileInfo(string.Format(@"D:\Samples\{0}.xml",
plcConstant.Name)), ExportOptions.WithDefaults);
}
```

Siehe auch

[Export von Projektierungsdaten \(Seite 778\)](#)

[Anmerkungen zur Leistung von TIA Portal Openness \(Seite 45\)](#)

8.5 Daten eines PLC-Geräts importieren/exportieren

8.5.3.4 Einzelne Variable oder Konstante in eine PLC-Variablenliste importieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Verwendung

Sie können in einem Importaufruf entweder Variablen oder Konstanten importieren.

Hinweis

Konstanten können nur als Anwenderkonstanten importiert werden.

Programmcode

Ändern Sie den folgenden Programmcode, um Variablengruppen oder einzelne Variablen und Konstanten aus einer XML-Datei zu importieren:

```
//Imports tags into a plc tag table
private static void ImportTag(PlcSoftware plcSoftware, string tagtableName)
{
    PlcTagTableSystemGroup plcTagTableSystemgroup = plcSoftware.TagTableGroup;
    PlcTagTable tagTable = plcTagTableSystemgroup.TagTables.Find(tagtableName);
    if(tagTable == null) return;

    tagTable.Tags.Import(new FileInfo(@"D:\Samples\myTags.xml"), ImportOptions.Override);
}

//Imports constants into a plc tag table
private static void ImportConstant(PlcSoftware plcSoftware, string tagtableName)
{
    PlcTagTableSystemGroup plcTagTableSystemgroup = plcSoftware.TagTableGroup;
    PlcTagTable tagTable = plcTagTableSystemgroup.TagTables.Find(tagtableName);
    if(tagTable == null) return;

    tagTable.UserConstants.Import(new FileInfo(@"D:\Samples\myConstants.xml"),
ImportOptions.Override);
}
```

Siehe auch

[Export von Projektierungsdaten \(Seite 778\)](#)

[Anmerkungen zur Leistung von TIA Portal Openness \(Seite 45\)](#)

8.5.4 Anwenderdatentyp exportieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Die PLC ist nicht online.

Programmcode

Ändern Sie den folgenden Programmcode, um einen Anwenderdatentyp in eine XML-Datei zu exportieren:

```
//Exports a user defined type
private static void ExportUserDefinedType(PlcSoftware plcSoftware)
{
    string udtname = "udt name XYZ";
    PlcTypeComposition types = plcSoftware.TypeGroup.Types;
    PlcType udt = types.Find(udtname);
    udt.Export(new FileInfo(string.Format(@"C:\OpennessSamples\udts\{0}.xml", udt.Name)), 
    ExportOptions.WithDefaults);
}
```

8.5.5 Anwenderdatentyp importieren

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Projekt öffnen (Seite 111)
- Die PLC ist nicht online.

Verwendung

Die API-Schnittstelle unterstützt das Importieren von Anwenderdatentypen aus einer XML-Datei.

Syntax der Importdatei

Der folgende Beispielcode zeigt einen Ausschnitt aus einer Importdatei eines benutzerdefinierten Datentyps:

```
<Section Name="Input">
<Member Name="Input1" Datatype="myudt1">
<Sections>
  <Section Name="None">
    <Member Name="MyUDT1Member1" Datatype="bool"/>
    <Member Name="MyUDT1Member2" Datatype="myudt1">
      <Sections...>
```

Hinweis

Syntax für benutzerdefinierte Datentypen von Elementen

Wenn der benutzerdefinierte Datentyp eines Elements in der Importdatei für Anwenderdatentypen eine falsche Syntax aufweist, wird eine Ausnahme ausgelöst.

Achten Sie darauf, dass benutzerdefinierte Datentypen mit "notiert werden.

Programmcode

Um einen Anwenderdatentyp zu importieren, ändern Sie folgenden Programmcode:

```
//Imports user data type
private static void ImportUserDataType(PlcSoftware plcSoftware)
{
  FileInfo fullFilePath = new FileInfo(@"C:\OpennessSamples\Import\ExportedPlcType.xml");
  PlcTypeComposition types = plcSoftware.TypeGroup.Types;
  IList<PlcType> importedTypes = types.Import(fullFilePath, ImportOptions.Override);
}
```

Siehe auch

[Import von Projektierungsdaten \(Seite 780\)](#)

8.5.6 Export von Daten im Format OPC UA XML

Voraussetzung

- Die Anwendung TIA Portal Openess ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Die PLC ist nicht online.

Verwendung

Sie können mit TIA Portal Openess PLC-Daten als Datei im Format OPC UA XML exportieren. Als Eingabeparameter für die Aktion benötigen Sie einen absoluten Verzeichnispfad, in dem die XML-Datei gespeichert wird.

Programmcode

Ändern Sie folgenden Programmcode, um PLC-Daten als OPC UA XML-Datei zu exportieren:

```
//Export PLC data as OPC UA XML file
private static void OpcUaExport(Project project, DeviceItem plc)
{
    OpcUaExportProvider opcUaExportProvider = project.HwUtilities.Find("OPCUAExportProvider")
        as OpcUaExportProvider;
    if (opcUaExportProvider == null) return;
    opcUaExportProvider.Export(plc, new FileInfo(string.Format(@"D:\OPC UA export files
\{0}.xml", plc.Name)));
}
```

8.6 Hardware-Daten importieren/exportieren

8.6.1 AML-Dateiformat

Einleitung

AutomationML ist ein neutrales, auf XML basierendes Datenformat zum Speichern und Austauschen von Engineering-Informationen einer Anlage und wird als offener Standard angeboten. Ziel von AutomationML ist die Verschaltung der heterogenen Werkzeuglandschaft moderner Engineering-Tools in deren unterschiedlichen Disziplinen wie mechanisches Anlagen-Engineering, Auslegung der Elektrik, HMI, PLC, Robotersteuerung.

8.6 Hardware-Daten importieren/exportieren

Das für den Export und Import von CAx-Daten verwendete Klassifizierungsmodell basiert auf den folgenden AML-Standards:

- Whitepaper AutomationML Teil 1 – AutomationML Architecture, Oktober 2014
- Whitepaper AutomationML Teil 2 – AutomationML Role Class Libraries, Oktober 2014
- Whitepaper AutomationML Teil 4 – AutomationML Logic, Mai 2010
- Whitepaper AutomationML – AutomationML Communication, September 2014
- Whitepaper AutomationML – AutomationML and eCI@ss Integration, November 2015
- Anwendungsempfehlungen: Automation Project Configuration - AR_001E Version 1.0.0, Mai 2017

Schema

Das AutomationML-Datenaustauschmodell wird durch das CAEX-Schema Version 2.15 beschrieben. Sie können diese Datei auf der Homepage AutomationML e.V. (<https://www.automationml.org/o.red.c/dateien.html>) herunterladen.

8.6.2 Pruned AML

Einleitung

Pruning beschreibt den Vorgang der Optimierung des Inhalts durch Entfernen bestimmter Dinge, die nicht unbedingt angegeben werden müssen. Bei externen Werkzeugen wie EPLAN haben die automatisch erstellten Submodulinformationen in einer Hardware-Konfiguration keine Bedeutung hinsichtlich EPLAN. Deshalb generieren diese Werkzeuge eine AML-Datei, indem die automatisch erstellten Submodulinformationen aus der Hardware-Konfiguration entfernt werden. Diese Datei wird Pruned AML genannt.

Pruned AML generieren

Die Generierung von Pruned AML basiert auf den folgenden Regeln in dieser Reihenfolge.

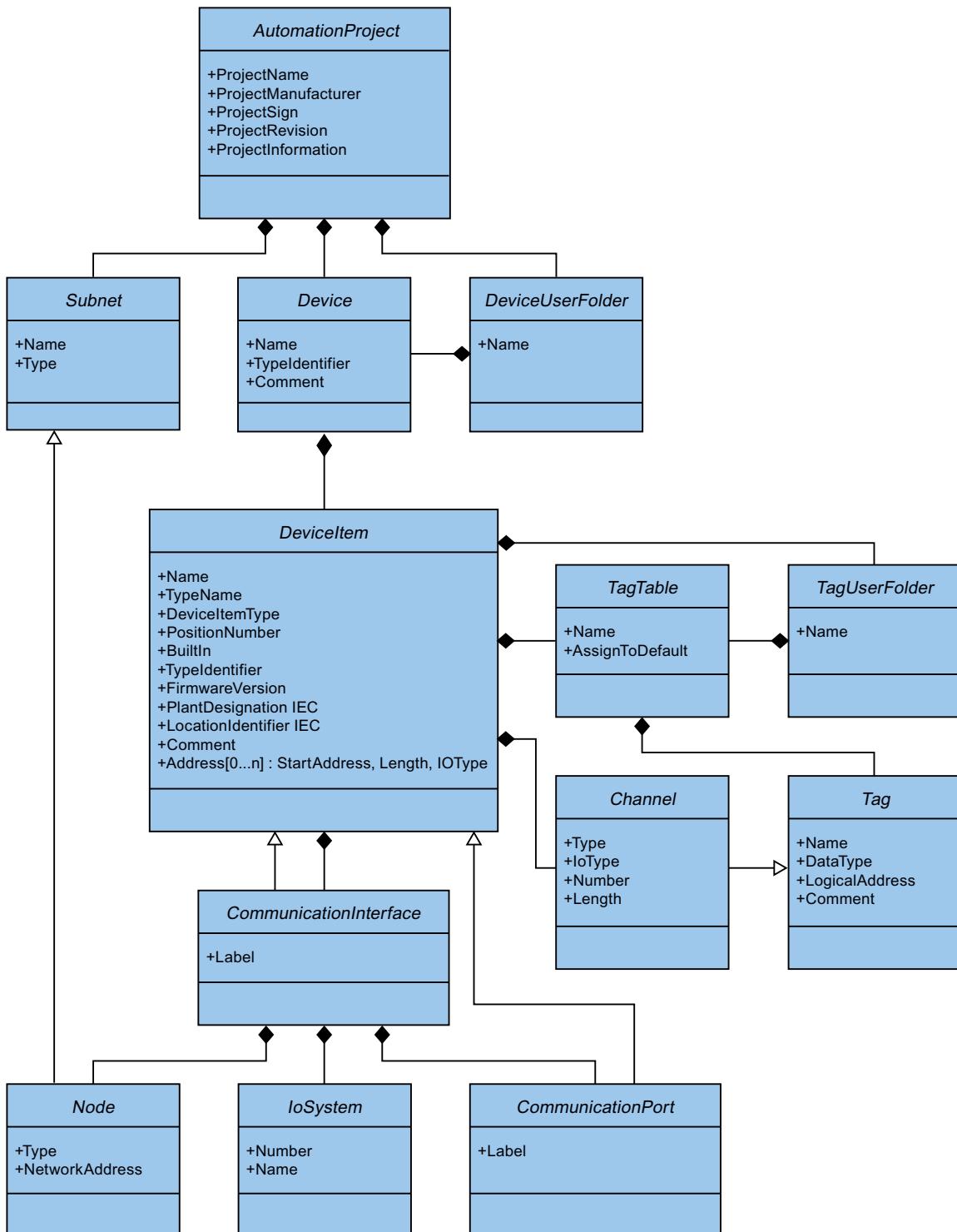
1. Wenn ein Geräteelement steckbar ist, wird kein Pruning durchgeführt.
2. Wenn ein Geräteelement vom Typ "Schnittstelle" oder "Port" ist, wird kein Pruning durchgeführt.
3. Wenn ein Geräteelement integriert ist und sich auf Baugruppenträgerebene befindet, wird kein Pruning durchgeführt.
4. Adressobjekte vom Typ "Diagnose" sind für den Pruning-Algorithmus nicht relevant.
5. Adressobjekte, die mit den automatisch erstellten Submodulen verknüpft sind, werden unter dem direkt übergeordneten Element angegeben (bei dem es sich um ein nicht automatisch erstelltes Submodul handeln muss).
6. Adressobjekte werden in die gleiche Sequenz eingeschlossen, die auch von TIA Portal Openness ausgegeben wird.

8.6.3 Übersicht über Objekte und Parameter von CAx-Import/-Export

Export/Import-Objekte und -Attribute

Die folgende Abbildung zeigt exportierbare Objekte mit ihren Attributen und Abhängigkeiten im CAx-Import/Export.

8.6 Hardware-Daten importieren/exportieren



8.6.4 Struktur der CAx-Daten zum Import/Export

Grundstruktur einer Exportdatei

Die Daten in der Exportdatei vom Import/Export sind mit Verweis auf eine Grundstruktur gegliedert. Die Exportdatei wird in einem AML-Format erzeugt.

Die AML-Datei beginnt mit einer Dokumentinformation. Die Datei beinhaltet die Daten der rechnerspezifischen Installation des exportierten Projekts.

In TIA Portal Openess ab V16 wird es möglich sein, AML-Dateien mit einer neuen Version von "AR APC V1.0.0" zu exportieren und zu importieren.

TIA Portal V16 kann AML-Dateien mit "AR APC V1.0.0" und "AR APC V1.1.0" importieren.

Bei AML-Dateien, die mit älteren Versionen (wie V14 SP1, V15, V15.1) des TIA Portals erzeugt wurden, müsste es nach wie vor möglich sein, diese Dateien mit "AR APC V1.0.0" zu exportieren und zu importieren.

8.6 Hardware-Daten importieren/exportieren

Die Exportdatei umfasst die folgenden zwei Abschnitte:

- Weitere Informationen

<WriterHeader> enthält Informationen zum Export- oder Importprozess. Der Import ignoriert den Inhalt des Abschnitts <AdditionalInformation>.

Zum Beispiel können Sie einen Baustein mit <AdditionalInformation>...</AdditionalInformation> einfügen und darin zusätzliche Informationen über die Validierung unterbringen. Nach Weiterleitung der AML-Datei kann der Empfänger anhand dieses Bausteins vor dem Import prüfen, ob die AML-Datei geändert wurde.

```
<xml version="1.0" encoding="utf-8">
<CAEXFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    FileName="CAx_asterisk_AML_03_V14.aml" SchemaVersion="2.15"
    xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
<AdditionalInformation>
    <WriterHeader>
        <WriterName>Totally Integrated Automation Portal</WriterName>
        <WriterID>1d4fccebb-1ad6-4881-b01d-bca335d94a46:V1.0</WriterID>
        <WriterVendor>Siemens AG</WriterVendor>
        <WriterVendorURL>www.siemens.com</WriterVendorURL>
        <WriterVersion>1400</WriterVersion>
        <WriterRelease>1400.100.101.16</WriterRelease>
        <LastWritingDateTime>2016-09-29T11:21:34.9551066Z</LastWritingDateTime>
    </WriterHeader>
</AdditionalInformation>
<AdditionalInformation AutomationMLVersion="2.0" />
<AdditionalInformation DocumentVersions="Recommendations">
    <Document DocumentIdentifier="AR APC" Version="1.1.0" />
</AdditionalInformation>
...
...
```

Hinweis

CAx exportiert und importiert die Version von AR APC in eine AML-Datei gemäß der installierten Version des TIA Portals.

- Instanzhierarchie

Dieser Abschnitt enthält die hierarchische Reihenfolge der exportierten internen Elemente.

```

...
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
<InternalElement ID="d4dc896a-f4a5-41b6-9c48-8d3a0a5a4343" Name="CAx_asterisk_AML_03_V14">
  <Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
  <Attribute Name="ProjectSign" AttributeDataType="xs:string" />
  <Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
  <Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
  <InternalElement ID="544f3a69-5f65-45ba-ac2f-1448db9493fd" Name="PN/IE_1">
    ...
  </InternalElement>

  <InternalElement ID="12116ac0-94b7-49d2-888d-7d39bbc0caf5" Name="S71500/ET200MP station_1">
    ...
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath=
    "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
  <InternalLink Name="Link To Port_1" RefPartnerSideA=
    "726148ce-de5b-4728-8886-4ba273435479:CommunicationPortInterface" RefPartnerSideB=
    "cb9d24f3-8200-4c89-9b40-24ae850e293e:CommunicationPortInterface" />
  <InternalLink Name="Link To Port_2" RefPartnerSideA=
    "cb9d24f3-8200-4c89-9b40-24ae850e293e:CommunicationPortInterface" RefPartnerSideB=
    "726148ce-de5b-4728-8886-4ba273435479:CommunicationPortInterface" />
  ...
  <InternalLink Name="Link To IoSystem_3" RefPartnerSideA=
    "d8f6e006-3778-4a05-aab1-df844fe822fe:LogicalEndPoint_Interface" RefPartnerSideB=
    "2344b7af-329c-4215-92d1-6143b4627b56:LogicalEndPoint_IoSystem" />
</InternalElement>
</InstanceHierarchy>
</CAEXFile>

```

Interne Elemente

Alle Objekte innerhalb der Instanzhierarchie der AML-Datei sind InternalElements. Das interne Element AutomationProject enthält alle internen Elemente aller Rollenklassifizierungen. Jedes interne Element unterstützt ein Set von Attributen.

Das Attribut <TypeIdentifier> identifiziert jeden Objekttyp eines Hardware-Objekts, das über TIA Portal Openness erstellt werden kann.

Hinweis

Automatisch erzeugte Objekte

Automatisch erzeugte Objekte können nur von anderen Objekten erzeugt werden. Sie haben keine Attribute und keine Typkennung. Sie werden in die exportierte Datei aufgenommen, doch Sie können den Export eines bestimmten automatisch erstellten Objekts nicht anstoßen.

8.6 Hardware-Daten importieren/exportieren

Am Ende des AML-Elements eines internen Elements wird Folgendes definiert:

- Rollenklassifizierung

Das Element `SupportedRoleClass` definiert den Objekttyp eines internen Elements. Der Objekttyp wird in der Rollenklassifizierungsbibliothek definiert, die den Standard-AML zu dem Objektmodell von TIA Portal Openness und TIA Portal abbildet.

```
...
<InternalElement ID="1d1a37ed-19d9-4a23-bc91-51f5a8e0244b" Name="Ungrouped devices">
  <InternalElement ID="ab193f5d-0375-4a6d-a576-a903e2b77cca" Name="ET 200SP station_1">
    ...
      <InternalElement ID="72d41729-90a7-4de3-9708-a8eeda6b1886" Name="IO device_1">
        ...
          <SupportedRoleClass RefRoleClassPath=
            "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
        </InternalElement>
        ...
        <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/Device" />
      </InternalElement>
      <SupportedRoleClass RefRoleClassPath=
        "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
    </InternalElement>
  ...

```

- Interne Links

Das Element `InternalLink` definiert die Kommunikationspartner einer Verbindung.

```
...
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
  <InternalElement ID="d4dc896a-f4a5-41b6-9c48-8d3a0a5a4343" Name="CAx_asterisk_AML_03_V14">
    <Attribute Name="ProjectManufacturer" Attribute DataType="xs:string" />
    <Attribute Name="ProjectSign" Attribute DataType="xs:string" />
    <Attribute Name="ProjectRevision" Attribute DataType="xs:string" />
    <Attribute Name="ProjectInformation" Attribute DataType="xs:string" />
    ...
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
    <InternalLink Name="Link To Port_1" RefPartnerSideA=
      "726148ce-de5b-4728-8886-4ba273435479:CommunicationPortInterface" RefPartnerSideB=
      "cb9d24f3-8200-4c89-9b40-24ae850e293e:CommunicationPortInterface" />
    <InternalLink Name="Link To Port_2" RefPartnerSideA=
      "cb9d24f3-8200-4c89-9b40-24ae850e293e:CommunicationPortInterface" RefPartnerSideB=
      "726148ce-de5b-4728-8886-4ba273435479:CommunicationPortInterface" />
    <InternalLink Name="Link To Port_3" RefPartnerSideA=
      "65307e3e-95fd-41ac-9982-e5e4fffc2fb15:CommunicationPortInterface" RefPartnerSideB=
      "58b1a3f2-f94b-48d1-ab5e-fbc4857cdfbc:CommunicationPortInterface" />
    <InternalLink Name="Link To Port_4" RefPartnerSideA=
      "58b1a3f2-f94b-48d1-ab5e-fbc4857cdfbc:CommunicationPortInterface" RefPartnerSideB=
      "65307e3e-95fd-41ac-9982-e5e4fffc2fb15:CommunicationPortInterface" />
    ...
  </InternalElement>
</InstanceHierarchy>
</CAEXfile>
```

Attribute

Attribute werden internen Elementen wie folgt zugeordnet:

```
...
<InternalElement ID="1d1a37ed-19d9-4a23-bc91-51f5a8e0244b" Name="Ungrouped devices">
  <InternalElement ID="ab193f5d-0375-4a6d-a576-a903e2b77ccaa" Name="ET 200SP station_1">
    <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
      <Value>System:Device.ET200SP</Value>
    </Attribute>
    <InternalElement ID="7636c362-a7af-47bb-8a18-e6428a6d61ff" Name="Rack_0">
      <Attribute Name="TypeName" AttributeDataType="xs:string">
        <Value>Rack</Value>
      </Attribute>
      <Attribute Name="PositionNumber" AttributeDataType="xs:int">
        <Value>0</Value>
      </Attribute>
      <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
        <Value>False</Value>
      </Attribute>
      <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
        <Value>System:Rack.ET200SP</Value>
      </Attribute>
    ...
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath=
    "AutomationProjectConfigurationRoleClassLib/Device" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath=
  "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
<InternalLink Name="Link To Port_1" RefPartnerSideA=
  "5758e2ff-3974-41e9-8bcc-b61a23f1bb58:CommunicationPortInterface"
  RefPartnerSideB="46683602-5129-4504-a9d1-48e6421e2cf0:CommunicationPortInterface" />
...
</InternalElement>
```

Handhabung von Attributen

Die Handhabung von Attributen wird für jedes Attribut einzeln wie folgt definiert:

- Ignoriert
Das Attribut wird beim Import ignoriert und ist in der Exportdatei nicht vorhanden.
- Obligatorisch
Das Attribut muss in einer Importdatei vorhanden sein und kann in der Exportdatei nicht gelöscht werden.
- Optional
Falls dieses Attribut in der Importdatei fehlt, wird der Standardwert für das Attribut festgelegt. Dieses Attribut fehlt in der Exportdatei, wenn es für ein Objekt nicht verwendbar ist, z. B. haben nicht alle Module eine Firmware-Version.

8.6 Hardware-Daten importieren/exportieren

- Nur Exportieren
Das Attribut wird intern durch das TIA Portal bestimmt, z. B. der Typename des Geräteelements. Wenn dieses in einer Importdatei vorhanden ist, wird es durch das TIA Portal während des Imports ignoriert.
- Nur Importieren
Das Attribut kann das Importverhalten beeinflussen. Falls dieses Attribut in der Importdatei fehlt, wird das Verhalten dem Standardwert für das Attribut entsprechen.

Siehe auch

[AML-Typkennungen \(Seite 928\)](#)

8.6.5 AML-Typkennungen

Interne Elemente

Der String `TypeIdentifier` besteht aus den folgenden Teilen:

- `<TypeIdentifierType>:<Identifier>`

Die folgenden Werte für `TypeIdentifierType` sind hier zulässig:

- `OrderNumber` dient der Angabe von physischen Modulen
- `GSD` dient der Angabe von GSD/GSDML-basierten Geräten
- `System` dient der Angabe von Systemen und generischen Geräten

Typkennung: OrderNumber

`OrderNumber` ist die allgemeine Typkennung für alle Module im Hardware-Katalog, ausgenommen GSD. AML-Typkennungen entsprechen nicht immer den TIA Portal Openness-Typkennungen. AML-Typkennungen haben keine Information zur `FirmwareVersion`. Die Information zur Firmwareversion ist im getrennten AML-Attribut "FirmwareVersion" enthalten.

Das Format für diese Typkennung ist eines der folgenden:

- <OrderNumber>
Beispiel: Bestellnummer: 3RK1 200-0CE00-0AA2

Hinweis

Platzhalter in den Bestellnummern

Es gibt einige wenige Module im Hardware-Katalog, die „Platzhalter“-Zeichen in ihren Bestellnummern verwenden, die eine bestimmte Gruppe realer Hardware repräsentieren, z. B. die verschiedenen Längen der S7-300-Baugruppenträger.

In diesem Fall können die spezifischen OrderNumber und die „Platzhalter“ OrderNumber beide verwendet werden, um eine Instanz des Hardware-Objekts zu erstellen. Sie können die Platzhalter jedoch nicht beliebig an jeder Stelle verwenden. Beispiel: Ein S7-300-Baugruppenträger kann auf die folgenden Arten erstellt werden:

Bestellnummer: 6ES7 390-1***0-0AA0

oder

Bestellnummer: 6ES7 390-1AE80-0AA0

Beachten Sie, dass Sie die folgende Struktur beispielsweise nicht verwenden dürfen:

Bestellnummer: 6ES7 390-1AE80-0A*0

Der Rückgabewert der ausgelesenen Typkennung ist immer die Bestellnummer aus dem Hardwarekatalog.

Beispiel: Bestellnummer: 6ES7 390-1AE80-0AA0 entspricht Bestellnummer: 6ES7 390-1***0-0AA0

Typkennung: GSD

Die Typkennung für GSD- und GSDML-basierte Geräte lautet TypeIdentifier = GSD:<Identifier>

Die Kennung besteht aus folgenden Elementen

- GsdName: Name der GSD oder GSDML in Großbuchstaben
- GsdType: Ist eines der folgenden:
 - D: Gerät (Device)
 - R: Baugruppenträger (Rack)
 - DAP: Kopfmodul
 - M: Modul
 - SM: Submodul
- GsdId: ID der GSD oder GSDML

8.6 Hardware-Daten importieren/exportieren

Die folgenden Formate für die Typkennungen werden beim CAx-Import/-Export unterstützt:

- GSD.<GsdName>/<GsdType>

Beispiele:

GSD:SIEM8139.GSD/DAP

GSD:GSDML-V2.31-SIEMENS-SINAMICS_DCP-20140313.XML/D

- <GsdName>/<GsdType>/<GsdId>

Beispiel:

GSD:SIEM8139.GSD/M/4

GSD:GSDML-V2.31-SIEMENS-SINAMICS_G110M-20140704.XML/M/IDM_DRIVE_47

Typkennung: System

System. ist die Kennung für Objekte, die nicht durch andere Kennungen bestimmt werden können. Die Formate für diese TypeIdentifierType sind die folgenden:

- <SystemTypeIdentifier>

Beispiele:

System:Device.S7300

System:Subnet.Ethernet

- <SystemTypeIdentifier>/<AdditionalTypeIdentifier>

Die AdditionalTypeIdentifier ist erforderlich, wenn SystemTypeIdentifier nicht eindeutig ist.

Die SystemTypeIdentifier hat ein Präfix für bestimmte Objekttypen:

Subnet.

Device.

Rack.

Beispiel: System:Rack.S71600/Large

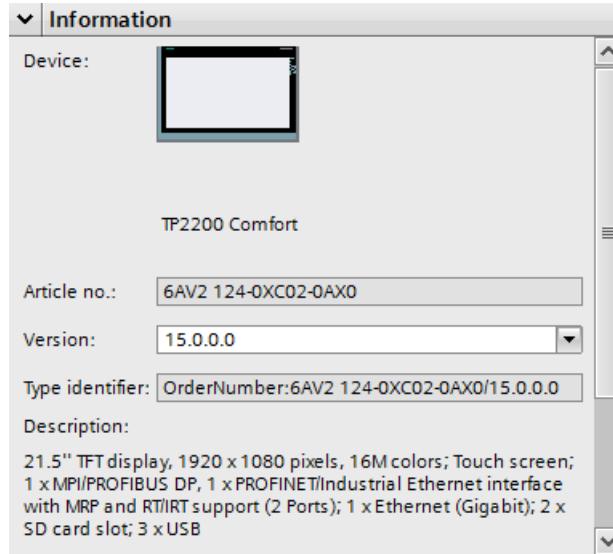
Ein Baugruppenträger mit Bestellnummer wird über die OrderNumber identifiziert.

Typkennungen im TIA Portal anzeigen

Wenn Sie eine Typkennung kennen müssen, ermitteln Sie sie im TIA Portal wie folgt:

1. Aktivieren Sie unter "Optionen > Einstellungen > Hardware-Konfiguration > Anzeige der Typkennung" die Einstellung "Anzeige der Typkennung für Geräte und Module".
2. Öffnen Sie den Editor "Geräte & Netze".
3. Wählen Sie ein Gerät im Katalog aus.

Die Typkennung wird unter "Information" angezeigt.



8.6.6 Exportieren/Importieren von Informationen zur Basiseinheit über AML

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Die PLC ist offline.

Anwendung

Im TIA Portal können Sie Informationen zur Basiseinheit aus dem TIA Portal exportieren und in das TIA Portal importieren, um den Informationsaustausch mit anderen Tools wie EPLAN zu vereinfachen.

Beim Export aus und Import in ein TIA Portal-Projekt werden zwei Arten von Basiseinheiten unterstützt:

- Einzelne Basiseinheit
- Doppelte Basiseinheit

Export einer Basiseinheit

Beispiel: CAx exportiert für ein Modul, das mit Informationen zur Basiseinheit in TIA Portal konfiguriert ist, die Informationen zur Basiseinheit als "Submodul" unter einem Modul in der AML-Datei.

Das Submodul der Basiseinheit wird stets exportiert mit:

- PositionNumber: 0
- DeviceItemType: Zubehör
- BuiltIn: False
- TypeIdentifier: "Typ-ID der Basiseinheit"
- ID: Immer ein zufällig generierter GUID

Export einer einzelnen Basiseinheit

Das untenstehende Beispiel zeigt eine AML-Datei für ein DI-Modul, das mit einer einzelnen Basiseinheit im TIA Portal konfiguriert ist.

```
<InternalElement ID="6f76c890-5c5d-41c4-9ade-96543b0222ac" Name="DI 8x24VDC ST_1">
...
<InternalElement ID="69233c1f-7ef7-4999-8e84-691d0ff3a210" Name="BaseUnit">
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>Accessory</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 193-6BP00-0DA0</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
...

```

Export einer doppelten Basiseinheit

Das untenstehende Beispiel zeigt eine AML-Datei für zwei DI-Module, die mit doppelter Basiseinheit im TIA Portal konfiguriert sind. Beispiel: Das erste Modul mit der Basiseinheit wird mit dem Präfix IX300 und das zweite mit derselben doppelten Basiseinheit wird mit dem Präfix IX301 konfiguriert.

Beim Export wird nur das erste Modul der Basiseinheit mit dem Präfix IX300 mit dem Submodul der Basiseinheit in die AML-Datei exportiert. Das zweite Modul, das mit dem Präfix IX301, wird in kein untergeordnetes Submodul eingefügt.

```

<InternalElement ID="6f76c890-5c5d-41c4-9ade-96543b0222ac" Name="DI 8x24VDC ST_1">
...
<InternalElement ID="3a1bee8a-12d0-4ec4-849c-333d45113d9c" Name="BaseUnit">
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>Accessory</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 193-6BP60-0DA0</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
...
<InternalElement ID="5f843491-b053-4dc8-b879-9ac327ee2a7e" Name="DI 8x24VDC ST_2">
...
<InternalElement ID="55c30280-6f8a-4c37-9b2d-41bb90941258" Name="DI 8x24VDC ST_2">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value> </Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
...

```

Import einer Basiseinheit

Es ist möglich, ein mit einem Submodul einer Basiseinheit konfigurierten Modul in eine AML-Datei zu importieren.

Während des Imports einer Basiseinheit

- haben GUID, PositionNumber, BuiltIn und DeviceItem keine Relevanz und werden deshalb nicht im TIA Portal angezeigt.
- Wenn ein unbeabsichtigtes Modul in der AML-Datei ein Submodul einer Basiseinheit hat, gibt Openness das Attribut "BaseUnit" dafür nicht zurück. Somit zeigt CAx in der Protokolldatei eine entsprechende Warnung an.
- CAx prüft nicht auf die Richtigkeit der MLFB der Basiseinheit in der AML-Datei (außer zur Ermittlung, ob es sich um eine einzelne oder doppelte Basiseinheit handelt). CAx versucht, die MLFB der Basiseinheit über Openness festzulegen. Tritt in Openness ein Fehler auf, wird ein entsprechender Fehler angezeigt.
- Der Import der Vorgängerversion der AML-Datei mit/ohne Informationen zur Basiseinheit ist erfolgreich, wenn Informationen ins TIA Portal importiert werden.

Import einer einzelnen Basiseinheit

Beim Import einer einzelnen Basiseinheit wird nur das Modul, das das Submodul der Basiseinheit in der AML-Datei enthält, mit Informationen zur Basiseinheit ins TIA Portal importiert.

Die einzelne Basiseinheit wird anhand des Typelidifier und dem nachfolgenden Muster in der AML-Datei identifiziert:

OrderNumber:xxxx 193-6[B|U|T]xYx-xxxx, dabei liegt der Wert von Y (11. Stelle) im Bereich von 0 bis 5.

Import einer doppelten Basiseinheit:

Beim Import einer doppelten Basiseinheit werden zwei (benachbarte) Module mit Informationen zur Basiseinheit im TIA Portal angezeigt. Das erste Modul mit einem Submodul einer doppelten Basiseinheit in der AML-Datei wird mit einer doppelten Basiseinheit importiert, indem das Suffix |X300 an die MLFB der Basiseinheit angehängt wird. Das zweite Modul, das kein untergeordnetes Submodul einer Basiseinheit hat, wird mit derselben Basiseinheit importiert, indem das Suffix |X301 an die MLFB der Basiseinheit angehängt wird. Die doppelte Basiseinheit wird anhand des Typelidifier und dem nachfolgenden Muster in der AML-Datei identifiziert:

OrderNumber:xxxx 193-6[B|U|T]x6x-xxxx

Siehe auch

Verbindung zum TIA Portal aufbauen (Seite 79)

Projekt öffnen (Seite 111)

8.6.7 Export/Import einer AML-Datei mit Verbindung zum Erweiterungsbaugruppenträger

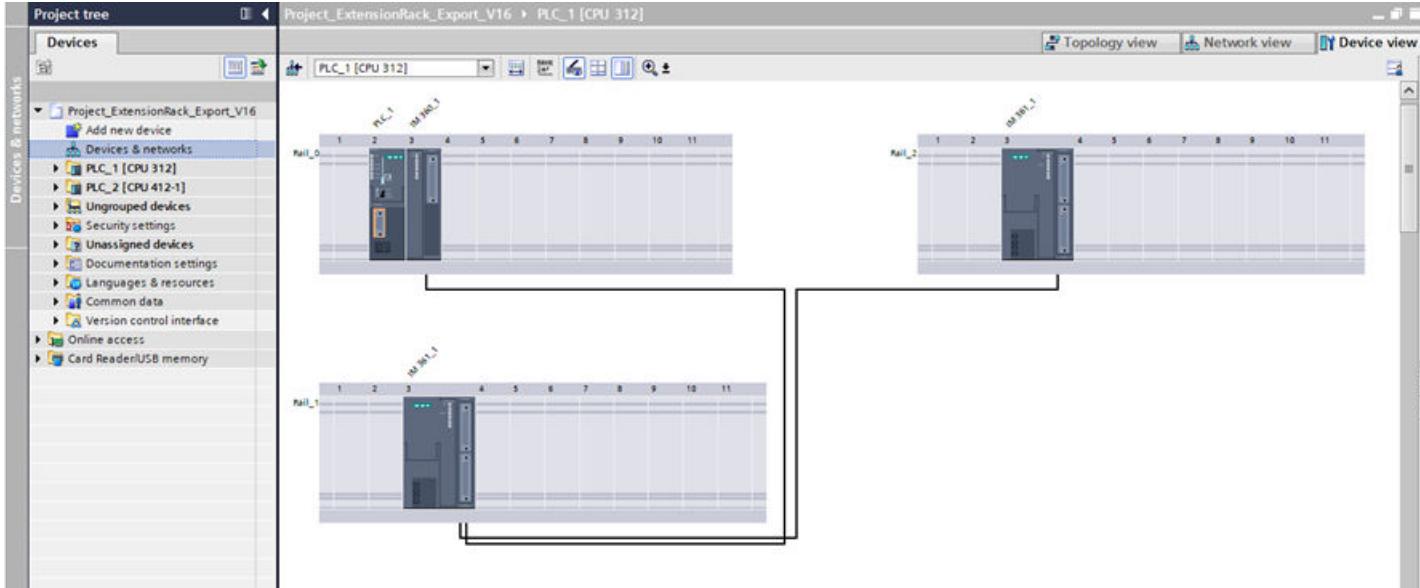
Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Im TIA Portal können Sie die Geräte mit mehreren Baugruppenträgern, die eine Erweiterungsbaugruppenträgerverbindung zu einer AML-Datei haben, exportieren und auch wieder importieren, um die gleiche Gerätekonfiguration zu erhalten, die im TIA Portal-Projekt angelegt wurde.

Beispiel: Gerät mit mehreren Baugruppenträgern mit Erweiterungsbaugruppenträgerverbindungen



AML-Struktur

Im TIA Portal werden Erweiterungsbaugruppenträgerverbindungen zwischen mehreren Baugruppenträgern direkt unter Sender- und Empfängermodulen modelliert (beide sind DeviceItem-Objekte). Entsprechend der AR APC-Empfehlung werden diese Verbindungen jedoch als Port-zu-Port-Verbindungen unter einem Kommunikationsport modelliert. Um diese Empfehlung zu erfüllen, wird unter dem IM-Modul ein Dummy-CommunicationInterface mit Dummy-CommunicationPort eingefügt.

8.6 Hardware-Daten importieren/exportieren

Das folgende Beispiel zeigt einen Ausschnitt der Elementstruktur der exportierten AML-Datei für obige Gerätekonfiguration:

```

<InternalElement ID="1ddb8d5c-d6cc-42c9-b1d8-621219b139f6" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>ExtensionRack</Value>
</Attribute>
...
<InternalElement ID="f25e531a-1793-4896-ade5-a87bd98de06e" Name="IM 46x SenderPort_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="9a824a06-89b9-4ba8-bee0-83c89b1f5e53"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
...
<InternalElement ID="d841278f-558a-41ab-9f03-91eeb454dc6b" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>ExtensionRack</Value>
</Attribute>
...
<InternalElement ID="2e1ff3b2-8a3e-4d5b-a95c-3ed027287db9" Name="IM 46x ReceiverPort_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="98460c75-a05d-4c23-8f88-33878ccd79c5"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="7615a32e-09dc-4171-88ed-118026357bae" Name="IM 46x SenderPort_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X2</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>2</Value>
</Attribute>

```

8.6 Hardware-Daten importieren/exportieren

```
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="846093a9-6473-4946-acf4-95a7813924df"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
...
<InternalElement ID="f4eb31c6-41d4-4a6e-ac33-de9c054a8c74" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>ExtensionRack</Value>
</Attribute>
...
<InternalElement ID="c11d2227-91db-41ae-9d94-d822e3ab9c7a" Name="IM 46x ReceiverPort_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="a9b2cce1-7078-4347-b5f2-428dalad5326"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
...
<InternalLink Name="Link To Port_1" RefPartnerSideA="f25e531a-1793-4896-ade5-
a87bd98de06e:CommunicationPortInterface" RefPartnerSideB="2e1ff3b2-8a3e-4d5b-
a95c-3ed027287db9:CommunicationPortInterface" />
<InternalLink Name="Link To Port_2"
RefPartnerSideA="7615a32e-09dc-4171-88ed-118026357bae:CommunicationPortInterface"
RefPartnerSideB="c11d2227-91db-41ae-9d94-d822e3ab9c7a:CommunicationPortInterface" />
```

Hinweis

Die Erweiterungsbaugruppenträgerschnittstelle wird nur dann exportiert, wenn Erweiterungsbaugruppenträgerverbindungen vorhanden sind. Auch basiert die Anzahl der Dummy-Ports unterhalb der Dummy-Erweiterungsbaugruppenträgerschnittstelle auf den an der Baugruppenträgerverbindung auf Modulebene teilnehmenden Ports. Im obigen Beispiel unterstützt das Modul "IM 460-0_1" zwei Ports (IM 46x SenderPort_1 und IM 46x SenderPort_2). Im TIA Portal ist jedoch nur ein Port mit Verbindung konfiguriert. Deshalb enthält die exportierte AML-Datei nur einen Port unterhalb der Erweiterungsbaugruppenträgerschnittstelle.

Erweiterungsbaugruppenträgerverbindung

Die XML-Darstellung von Erweiterungsbaugruppenträgerverbindungen zwischen mehreren Baugruppenträgern geschieht im nachfolgend dargestellten Format.

ExternalInterface-

Das interne Element <ExternalInterface> muss unter dem internen Element <CommunicationPort> hinzugefügt werden, das an der Verbindung teilnimmt.

```
<InternalElement ID="[IM Module Unique ID]" Name="[IM Module Name]">
...
<InternalElement ID="[Dummy Interface Unique ID]" Name="RackExtension">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>ExtensionRack</Value>
</Attribute>
...
<InternalElement ID="[Dummy Port Unique ID]" Name="[IM Module Sender/Receiver Name]">
...
<ExternalInterface ID="[External Interface Unique ID]" Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="[Dummy Port Unique ID]" Name="[IM Module Sender/Receiver Name]">
...
<ExternalInterface ID="[External Interface Unique ID]" Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
```

8.6 Hardware-Daten importieren/exportieren

Interner Link

Erweiterungsbaugruppenträgerverbindungen werden mit Hilfe von <InternalLink>-Variablen dargestellt. Die <InternalLink>-Variablen müssen unter dem gemeinsamen übergeordneten Element der verschiedenen Baugruppenträger (z. B. Gerät) hinzugefügt werden. Der Name des internen Links muss im gemeinsamen übergeordneten Element eindeutig sein.

```
<InternalLink Name="Link To [Internal link Name]" RefPartnerSideA="[Communication Port UniqueID]:[Communication Port External Interface Name]" RefPartnerSideB="[Communication Port UniqueID]:[Communication Port External Interface Name]" />
```

8.6.8 Verwalten von Verbindungen bei Erweiterungsbaugruppenträgern

Voraussetzung

- Die Anwendung TIA Portal Openess ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)

Anwendung

Sie können mithilfe von TIA Portal Openess Verbindungen für Erweiterungsbaugruppenträger abrufen, hinzufügen und entfernen. Dadurch können Sie auch die Unterstützung von TIA Portal Openess für die Realisierung von Verbindungen für Erweiterungsbaugruppenträger beim CAx-Export/Import nutzen.

Programmcode

```
ImConnection imConnection = portDeviceItem.GetService<ImConnection>();  
imConnection.Connect(partnerport);  
imConnection.Disconnect();  
imConnection.GetPartnerPort();  
var imConnectionOwner = imConnection.OwnedBy;
```

Siehe auch

Verbindung zum TIA Portal aufbauen (Seite 79)

Projekt öffnen (Seite 111)

8.6.9 Export von CAx-Daten

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Im TIA Portal können Sie Ihre Konfiguration im Editor "Geräte & Netze" in eine AML-Datei exportieren. Mit dieser auf TIA Portal Openness basierenden Funktion lassen sich Hardwaredaten von der Projekt- oder Geräteebene exportieren.

TIA Portal Openness bietet die folgenden Möglichkeiten zum Exportieren von CAx-Daten:

- Exportfunktion
Die Exportfunktion ist über den Dienst `CaxProvider` zugänglich. Für den Dienst `CaxProvider` rufen Sie die Methode `GetService` am Objekt `Project` auf.
- Befehlszeilenschnittstelle
Sie führen "Siemens.Automation.Cax.AmiHost.exe" unter "C:\Programme\Siemens\Automation\Portal V..\\Bin" aus, indem Sie spezifische Argumente an der Befehlszeile übergeben.

Einschränkungen für CAx in Bezug auf das Exportieren und Importieren

Nicht unterstützt von CAx wird das Exportieren und Importieren von

- Port-Port-Verbindungen
- Verbindungen zu und zwischen Erweiterungsbaugruppenträgern
- Multi-CPUs
- HMI-Geräten außer Push Button Panels und Key Panels
- Antrieben
- Ausgabemodus und -bereich von analogen Kanälen
- Gepackten Adressen

Nicht unterstützt von CAx wird das Exportieren und Importieren folgender Geräte und Antriebe:

- 6AV2 104-0XXXX-XXXX
- 6AV2 155-0XXXX-XXXX
- 6ES7 XXX-XXXXX-XXXX
- 6ES7 370-0AA01-0AA0
- 6ES7 451-3AL00-0AE0
- 6GK5 414-3FC00-2AA2
- 6GK5 414-3FC10-2AA2

8.6 Hardware-Daten importieren/exportieren

- 6GK5 495-8BA00-8AA2
- 6GK5 496-4MA00-8AA2
- 6GK5 602-0BA00-2AA3
- 6GK5 602-0BA10-2AA3
- 6GK5 612-0BA00-2AA3
- 6GK5 612-0BA10-2AA3
- 6GK5 613-0BA00-2AA3
- 6GK5 623-0BA10-2AA3
- 6GK5 627-2BA10-2AA3
- System:Device.Scalance/S627
- System:IPProxy.Device
- System:IPProxy.Rack

Programmcode: Auf den Dienst **CaxProvider** zugreifen

Ändern Sie den folgenden Programmcode, um auf den Dienst **CaxProvider** zuzugreifen:

```
Project project = tiaPortal.Projects.Open(...);
CaxProvider caxProvider = project.GetService<CaxProvider>();

if(caxProvider != null)
{
    // Perform CAx export and import operation
}
```

CAx-Export auf Projektebene

Um CAx-Daten auf Projektebene zu exportieren, verwenden Sie die Methode `Export` mit den folgenden Parametern:

Name	Beispiel	Beschreibung
ProjectToExport	tiaPortal.Projects[0]	Projektobjekt zum Exportieren
ExportFilePath	new FileInfo(@"D:\Temp\ProjectExport.aml")	Vollständiger Exportdateipfad der AML-Datei
LogFilePath	new FileInfo(@"D:\Temp\ProjectExport_Log.log")	Vollständiger Dateipfad der Protokolldatei

Um CAx-Daten auf Projektebene zu exportieren, ändern Sie folgenden Programmcode:

```
caxProvider.Export(project, new FileInfo(@"D:\Temp\ProjectExport.aml"),
new FileInfo(@"D:\Temp\ProjectExport_Log.log"));
```

CAx-Export auf Geräteebene

Um CAx-Daten auf Geräteebene zu exportieren, verwenden Sie die Methode `Export` mit den folgenden Parametern:

Name	Beispiel	Beschreibung
DeviceToExport	project.Devices[0]	Geräteobjekt zum Exportieren
ExportFilePath	new FileInfo(@"D:\Temp\ProjectExport.aml")	Vollständiger Exportdateipfad der AML-Datei
LogFilePath	new FileInfo(@"D:\Temp\ProjectExport_Log.log")	Vollständiger Dateipfad der Protokolldatei

Um CAx-Daten auf Projektebene zu exportieren, ändern Sie folgenden Programmcode:

```
caxProvider.Export(device, new FileInfo(@"D:\Temp\DeviceExport.aml"), new
FileInfo(@"D:\Temp\DeviceExport_Log.log"));
```

CAx-Export über Befehlszeile

Um CAx-Daten über die Befehlszeile zu exportieren, verwenden Sie `Siemens.Automation.Cax.AmiHost.exe` mit den folgenden Parametern:

Parameter	Beispiel	Beschreibung
-p	-p "D:\Temp\MyProject.ap*"	Gibt einen vollständigen Pfadnamen zu einem TIA Portal-Projekt an.
-d	-d "S7300/ET200M station_1"	Optionaler Parameter. Sofern kein Gerät angegeben ist, erfolgt der Export auf Projektebene. Gibt den Namen des Geräts oder der Station innerhalb des angegebenen TIA-Projekts an, das exportiert werden soll.
-m	-m "AML"	Gibt den Export-/Importmodus an (Format für Export/Import):. "AML" exportiert in das AML-Format
-e	-e "D:\Import" -e "D:\Import\CAx_Export.aml"	Gibt den vollständigen Pfad der zu exportierenden AML-Datei an. Der Projektname wird als Name der exportierten Datei verwendet, falls nur ein Pfad angegeben ist.

8.6 Hardware-Daten importieren/exportieren

Um CAx-Daten auf Projektebene über die Befehlszeile zu exportieren, ändern Sie folgenden Programmcode:

```
//please adapt the path and the extension apx to the installed version of  
TIA Portal  
Siemens.Automation.Cax.AmiHost.exe -p "D:\Temp\MyProject.apx" -m "AML" -e  
"D:\Import\CAx_Export.aml"
```

Um CAx-Daten auf Geräteebene über die Befehlszeile zu exportieren, ändern Sie folgenden Programmcode:

```
//please adapt the path and the extension apx to the installed version of  
TIA Portal  
Siemens.Automation.Cax.AmiHost.exe -p "D:\Temp\MyProject.apx" -d "S7300/  
ET200M station_1" -m "AML" -e "D:\Import\CAx_Export.aml"
```

8.6.10 Import von CAx-Daten

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Verwendung

Im TIA Portal können Sie Ihre Konfiguration im Editor "Geräte & Netze" aus einer AML-Datei importieren. Mit dieser Funktion lassen sich Hardwaredaten von der Projekt- oder Geräteebene importieren.

TIA Portal Openness bietet die folgenden Möglichkeiten zum Exportieren von CAx-Daten:

- Importfunktion
Die Importfunktion ist über den Dienst `CaxProvider` zugänglich. Für den Dienst `CaxProvider` rufen Sie die Methode `GetService` am Objekt `Project` auf.
- Befehlszeile
Sie führen "Siemens.Automation.Cax.AmiHost.exe" unter "C:\Programme\Siemens\Automation\Portal V..\Bin\" aus, indem Sie spezifische Argumente an der Befehlszeile übergeben:

Programmcode: Auf den Dienst CaxProvider zugreifen

Ändern Sie folgenden Programmcode:

```
//Access the CaxProvider service
Project project = tiaPortal.Projects.Open(...);
CaxProvider caxProvider = project.GetService<CaxProvider>();

if(caxProvider != null)

{
    // Perform Cax export and import operation
}
```

Cax-Import

Um CAx-Daten in ein TIA Portal-Projekt zu importieren, verwenden Sie die Methode `Import` mit den folgenden Parametern:

Name	Beispiel	Beschreibung
ImportFilePath	<code>new FileInfo(@"D:\Temp\ProjectExport.aml")</code>	Vollständiger Importdateipfad der AML-Datei
LogFilePath	<code>new FileInfo(@"D:\Temp\ProjectExport_Log.log")</code>	Vollständiger Dateipfad der Protokolldatei
ImportOptions	<code>CaxImportOptions.MoveToParkingLot</code> <code>CaxImportOptions.RetainTiaDevice</code> <code>CaxImportOptions.OverwriteTiaDevice</code>	Strategien zur Konfliktlösung beim Import in ein bereits vorhandenes, nicht leerer Projekt.

Um CAx-Daten zu importieren, ändern Sie folgenden Programmcode:

```
caxProvider.Import(new FileInfo(@"D:\Temp\ProjectImport.aml"), new
FileInfo(@"D:\Temp\ProjectImport_Log.log"),
CaxImportOptions.MoveToParkingLot);
```

Die folgenden `CaxImportOptions` sind gegeben:

Importoption	Beschreibung
<code>MoveToParkingLot</code>	Gerät(e) mit Namenskonflikt im Projekt beibehalten und diejenigen aus den CAx-Daten in einen Parkplatzordner importieren
<code>RetainTiaDevice</code>	Gerät(e) mit Namenskonflikt im Projekt beibehalten und diejenigen aus den CAx-Daten nicht importieren
<code>OverwriteTiaDevice</code>	Gerät(e) mit Namenskonflikt im Projekt durch diejenigen aus den CAx-Daten überschreiben

CAX-Import über Befehlszeile

Um CAX-Daten über die Befehlszeile zu importieren, verwenden Sie Siemens.Automation.Cax.AmiHost.exe mit den folgenden Parametern:

Parameter	Beispiel	Beschreibung
-p	-p "D:\Temp\MyProject.apx"	Gibt einen vollständigen Pfadnamen zu einem TIA Portal-Projekt an.
-m	-m "AML"	Gibt den Export-/Importmodus an (Format für Export/Import):. "AML" exportiert in das AML-Format
-i	-i "D:\Import\CAX_Export.aml"	Gibt den vollständigen Pfad der zu importierenden AML-Datei an.
-c	-c "ParkingLot"	Gibt unterschiedliche Strategien an, wenn es nach den Importoptionen einen Konflikt beim Gerätenamen gibt.

Um CAX-Daten über die Befehlszeile zu importieren, ändern Sie folgenden Programmcode:

```
//please adapt the path and the extension apx to the installed version of
TIA Portal
Siemens.Automation.Cax.AmiHost.exe -p "D:\Temp\MyProject.apx" -m "AML" -i
"D:\Import\CAX_Export.aml"
```

Folgende Importoptionen stehen zur Verfügung:

Importoption	Beschreibung
ParkingLot	Gerät(e) mit Namenskonflikt im Projekt beibehalten und diejenigen aus den CAX-Daten in einen Parkplatzordner importieren
RetainTia	Gerät(e) mit Namenskonflikt im Projekt beibehalten und diejenigen aus den CAX-Daten nicht importieren
OverwriteTia	Gerät(e) mit Namenskonflikt im Projekt durch diejenigen aus den CAX-Daten überschreiben

8.6.11 Export/Import von Submodulen

Voraussetzungen

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden. Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet. Siehe Projekt öffnen (Seite 111)
- Der PLC ist offline.

Anwendung

Zwischen dem TIA Portal und anderen Engineering-Tools, z. B. einem CAD-Tool wie EPLAN, können Submoduldaten im Round-Trip-Verfahren ausgetauscht werden, wobei während des Exports und Imports eine gemeinsame Hierarchie der Submodule in der AML-Datei beibehalten werden muss. Beispielsweise müssen die Submodule, wie etwa Busadapter, in TIA Portal eine andere interne Hierarchie aufweisen als in anderen Anwendungen (z. B. CAD-Tools wie EPLAN).

AML-Struktur der Exportdatei

Sie können Submoduldaten aus der TIA Portal-Hierarchie in die AML-Dateihierarchie exportieren.

8.6 Hardware-Daten importieren/exportieren

Das folgende Beispiel zeigt einen Ausschnitt der AML-Dateistruktur, die beim Exportieren eines Busadapters als Submodul aus dem TIA Portal erzeugt wird.

```

<?xml version="1.0" encoding="utf-8"?>
<CAEXFile FileName="Project4.aml" SchemaVersion="2.15"
xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
<AdditionalInformation>
<WriterHeader>
<WriterName>Totally Integrated Automation Portal</WriterName>
<WriterID>1d4fccebb-1ad6-4881-b01d-bca335d94a46:V1.0</WriterID>
<WriterVendor>Siemens AG</WriterVendor>
<WriterVendorURL>www.siemens.com</WriterVendorURL>
<WriterVersion>15</WriterVersion>
<WriterRelease>1500.0100.0.0</WriterRelease>
<LastWritingDateTime>2018-05-03T11:23:10.3011329Z</LastWritingDateTime>
</WriterHeader>
</AdditionalInformation>
<AdditionalInformation AutomationMLVersion="2.0" />
<AdditionalInformation DocumentVersions="Recommendations">
<Document DocumentIdentifier="AR APC" Version="1.1.0" />
</AdditionalInformation>
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
<InternalElement ID="6cd7f80f-e049-4958-ba67-630481805bf0" Name="Project4">
<Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
<Attribute Name="ProjectSign" AttributeDataType="xs:string" />
<Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
<Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
<InternalElement ID="b27045c4-9cb3-4b8d-916b-85f8100d1602" Name="Ungrouped devices">
<InternalElement ID="3f770698-940d-49c2-9f77-06fc458e1340" Name="ET 200SP station_1">
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>System:Device.ET200SP</Value>
</Attribute>
<InternalElement ID="6f52fbab-a221-4d54-9368-84c392ca7fec" Name="Rack_0">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>Rack</Value>
...
<InternalElement ID="f7445c0b-1c52-4a84-915f-2c8bee13af70" Name="BA 2xRJ45">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>BA 2xRJ45</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>127</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 193-6AR00-0AA0</Value>
</Attribute>
<Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
<Value>V0.0</Value>
</Attribute>
<InternalElement ID="40f8bbce-35d3-4d65-907a-bece3e0144e0" Name="PROFINET interface">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">

```

8.6 Hardware-Daten importieren/exportieren

```
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<InternalElement ID="8fb775eb-96c6-48d6-af8a-96ba72418830" Name="IE1">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Ethernet</Value>
</Attribute>
<Attribute Name="NetworkAddress" AttributeDataType="xs:string">
<Value>192.168.0.1</Value>
</Attribute>
<Attribute Name="SubnetMask" AttributeDataType="xs:string">
<Value>255.255.255.0</Value>
</Attribute>
<Attribute Name="IpProtocolSelection" AttributeDataType="xs:string">
<Value>Project</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
</InternalElement>
<InternalElement ID="f28a3d93-d821-4556-9df1-a45f0e4ff6a6" Name="Port_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P1 R</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="ad6a0faa-3b70-4528-8c54-8183018b6714" Name="Port_2">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P2 R</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>2</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
...
...
```

Hinweis

CAx exportiert und importiert die Version von AR APC in eine AML-Datei gemäß der installierten Version des TIA Portals.

Import von Submodulen

Sie können Submodule aus AML-Dateien importieren, die beim vorhergehenden Export erzeugt wurden.

Hinweis

- Das veränderte Verhalten der Exporthierarchie gilt erst ab Version V15.1 aufwärts.
- Die Hierarchie in der AML-Datei darf die interne Hierarchie in TIA Portal nach dem Import nicht beeinflussen.
- Mit älteren TIA Portal-Versionen erzeugte AML-Dateien sollten ebenfalls fehlerfrei importiert werden.
- Dieses Verhalten in Bezug auf die Änderung/Umwandlung der Hierarchie gilt sowohl für eingebaute als auch für nicht eingebaute Submodule.

Mehrere Submodule an einer gemeinsamen Schnittstelle

In manchen Fällen sind mehrere Submodule an einer gemeinsamen Schnittstelle vorhanden. Beispiel: IO-Device: IM 155-6 PN/3 HF 6ES7 155-6AU30-0CN0/V4.2. Dieses Modul verfügt über zwei nicht eingebaute Busadapter an der gleichen Schnittstelle. In diesem Fall muss es möglich sein, die Busadapter aus der TIA Portal-Hierarchie in die erforderliche AML-Dateihierarchie zu exportieren. In diesem Beispiel aus der TIA Portal-Hierarchie hat die PROFINET-Schnittstelle zwei Busadapter, drei Ports und einen Knoten. Hier gehören Port_1 und Port_2 logisch zu BA 2xRJ45 und Port_3 gehört logisch zu BA 2xRJ45_1, obwohl alle drei Ports an einer Schnittstelle zusammengeführt sind.

Beim Export:

- Nur dem ersten Submodul wird die Original-Schnittstelle mit den verbindungsrelevanten Informationen zugewiesen. Hier wird BA 2xRJ45 die Original-Schnittstelle mit Knoten 'IE1', 'Port_1' und 'Port_2' zugewiesen.
- Die anderen Submodule werden einer 'duplicierten' Schnittstelle mit Ports zugewiesen, die logisch zum Submodul gehören. Hier wird BA 2xRJ45_1 eine 'duplicierte' Schnittstelle und Port_3 zugewiesen.
- Wenn das Kopfmodul mit einem Subnetz/IO-System verbunden ist, wird die zugehörige Verbindungsinformation (wie ExternalInterface-Links) nur als Teil des ersten Submoduls exportiert (ExternalInterface-Link zum Subnetz unter 'Node' und ExternalInterface-Link zum IO-System unter 'Interface').
- Die Verbindungsinformationen zur Topologieverschaltung gehören zum jeweiligen Port.

Beim Import:

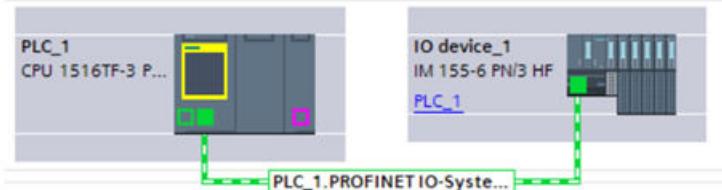
- Es ist möglich, mehrere Submodule aus einer AML-Datei zu importieren, die beim vorhergehenden Export erzeugt wurde.
- Es ist möglich, dass eine von EPLAN erzeugte AML-Datei in der sekundären Schnittstelle Knoteninformationen enthält.

8.6 Hardware-Daten importieren/exportieren

- Knotendetails, die ein Duplikat aus dem Knoten der primären Schnittstelle sind, werden wie folgt gehandhabt:
 - Knotenattribute: Details zu Knotenattributen, die während der Verarbeitung der primären Schnittstelle festgelegt wurden, werden überschrieben.
 - Subnetzverbindung: Wird stillschweigend übersprungen, wenn die Verbindung bereits besteht, andernfalls wird die Verbindung hergestellt.
- Wenn die AML-Datei Details zur IO-Systemverbindung an der sekundären Schnittstelle enthält, dann
 - Wird die Verbindungsherstellung übersprungen, sofern die Verbindung bereits vorhanden ist, und der Anwender wird über eine entsprechende Fehlermeldung in der Registerkarte "Info" benachrichtigt.
 - Wird die Verbindung hergestellt, sofern sie noch nicht vorhanden ist.

Die folgende Konfiguration zeigt eine IO-Device-Konfiguration mit Master-Slave-Verbindungen und Topologieverbindungen.

Network View



Topology View



Das folgende Beispiel zeigt einen Ausschnitt einer AML-Datei, die beim Export für obige Konfiguration erzeugt wird:

8.6 Hardware-Daten importieren/exportieren

```
<?xml version="1.0" encoding="utf-8"?>
<CAEXfile FileName="MultipleBA_01.aml" SchemaVersion="2.15"
xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
<AdditionalInformation>
<WriterHeader>
<WriterName>Totally Integrated Automation Portal</WriterName>
<WriterID>1d4fcebb-1ad6-4881-b01d-bca335d94a46:V1.0</WriterID>
<WriterVendor>Siemens AG</WriterVendor>
<WriterVendorURL>www.siemens.com</WriterVendorURL>
<WriterVersion>15</WriterVersion>
<WriterRelease>1501.0000.0.0</WriterRelease>
<LastWritingDateTime>2018-05-17T09:36:46.9230179Z</LastWritingDateTime>
</WriterHeader>
</AdditionalInformation>
<AdditionalInformation AutomationMLVersion="2.0" />
<AdditionalInformation DocumentVersions="Recommendations">
<Document DocumentIdentifier="AR APC" Version="1.1.0" />
</AdditionalInformation>
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
<InternalElement ID="e005c094-1b0a-42c4-92a0-67c981508c1a" Name="Project45">
<Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
<Attribute Name="ProjectSign" AttributeDataType="xs:string" />
<Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
<Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
<InternalElement ID="2782e61d-8c27-46cb-93ea-6b804157ae60" Name="PN/IE_1">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Ethernet</Value>
</Attribute>
<ExternalInterface ID="2d901881-a2bf-4fe7-915f-b2542b346988" Name="LogicalEndPoint_Subnet">
RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
Subnet" />
...
<InternalElement ID="dc5cf410-2516-4b0b-ad1a-c43117d8c9b3" Name="BA 2xRJ45">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>BA 2xRJ45</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>127</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 193-6AR00-0AA0</Value>
</Attribute>
<Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
<Value>V0.0</Value>
</Attribute>
<InternalElement ID="f04874a8-2d35-47c4-93ae-d6fdc2668479" Name="PROFINET interface">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
```

```
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="81a7d9df-99b8-4eca-8e72-404b22bd05e7"
Name="LogicalEndPoint_Interface" RefBaseClassPath="CommunicationInterfaceClassLib/
LogicalEndPoint" />
<InternalElement ID="83eb7d69-8cd5-4217-a07a-0c656d215ec7" Name="IE1">
...

```

Pruning von AML-Dateien

Das Pruning (Bereinigen) beschreibt den Vorgang der Optimierung des Inhalts durch Entfernen bestimmter Dinge, die nicht unbedingt angegeben werden müssen. Informationen über das Pruning von AML-Dateien enthält Pruning von AML-Dateien (Seite 920).

In manchen Fällen kann die Hierarchie der Submodule in TIA Portal und CAD-Tools (wie EPLAN) auf Grund von Pruning der Submodule voneinander abweichen. In diesen Fällen muss TIA Portal den Import von AML-Dateien sowohl mit als auch ohne Pruning unterstützen.

Hinweis

- Der Export von AML-Dateien ohne Pruning wird in TIA Portal immer unterstützt.
 - Der Import von AML-Dateien, ob mit oder ohne Pruning, wird in TIA Portal immer unterstützt.
-

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

8.6.12 CAx-Daten ohne logische Adresse importieren

Voraussetzung

- Die Anwendung TIA Portal Openess ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Beim CAx-Import in TIA Portal können Sie die Verbindung zwischen Kanal und Variable konfigurieren, ohne dass Sie die Anfangsadresse eines E/A-Moduls und/oder die logische Adresse der Variablen in der AML-Datei angegeben haben.

8.6 Hardware-Daten importieren/exportieren

Das folgende AML-Beispiel zeigt die XML-Datei, die ohne Attribute für Anfangsadresse und logische Adresse erzeugt werden soll.

```

<?xml version="1.0" encoding="utf-8"?><CAEXFile FileName="TagsExport.aml"
SchemaVersion="2.15" xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
...
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
<InternalElement ID="fff25423-fe9e-4334-9331-4cec118e06f7" Name="Project1">
...
<InternalElement ID="59c0b48b-aa6c-45c3-8dc8-bba5367bd4fb" Name="S7300/ET200M station_1">
...
<InternalElement ID="1b7b2b24-243b-4348-831e-bc46bc35957f" Name="Rail_0">
...
<InternalElement ID="974ca791-ad8d-482b-be80-2cf4e8dcedaf" Name="PLC_1">
...
<InternalElement ID="9564bcc2-8ea0-4be7-a950-5c55b34e474a" Name="Default tag table">
<ExternalInterface ID="7fd969e6-c2c9-45a8-b573-68833df327f5" Name="Tag_1"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Tag">
<Attribute Name="DataType" Attribute DataType="xs:string">
<Value>Bool</Value>
</Attribute>
</ExternalInterface>
<ExternalInterface ID="33899862-86c1-4171-832a-1136b6e59b9d" Name="Tag_2"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Tag">
<Attribute Name="DataType" Attribute DataType="xs:string">
<Value>Byte</Value>
</Attribute>
</ExternalInterface>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
TagTable" />
</InternalElement>
...
<Attribute Name="PositionNumber" Attribute DataType="xs:int">
<Value>8</Value>
</Attribute>
<Attribute Name="BuiltIn" Attribute DataType="xs:boolean">
<Value>true</Value>
</Attribute>
<Attribute Name="Address">
<RefSemantic CorrespondingAttributeName="OrderedListType" />
<Attribute Name="1">
<Attribute Name="StartAddress" Attribute DataType="xs:int">
<Value>832</Value>
</Attribute>
<Attribute Name="Length" Attribute DataType="xs:int">
<Value>128</Value>
</Attribute>
<Attribute Name="IoType" Attribute DataType="xs:string">
<Value>Input</Value>
</Attribute>
</Attribute>
<Attribute Name="2">
<Attribute Name="StartAddress" Attribute DataType="xs:int">
<Value>832</Value>
</Attribute>
<Attribute Name="Length" Attribute DataType="xs:int">
<Value>128</Value>
</Attribute>

```

8.6 Hardware-Daten importieren/exportieren

```
<Attribute Name="IoType" AttributeDataType="xs:string">
<Value>Output</Value>
</Attribute>
</Attribute>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<InternalElement ID="29e0bb63-0050-46e3-968a-fcecf4eb050a" Name="DI 16x24VDC_1">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>DI16 x 24VDC</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>4</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 321-1BH02-0AA0</Value>
</Attribute>
<Attribute Name="Address">
<RefSemantic CorrespondingAttributePath="OrderedListType" />
<Attribute Name="1">
<Attribute Name="Length" AttributeDataType="xs:int">
<Value>16</Value>
</Attribute>
<Attribute Name="IoType" AttributeDataType="xs:string">
<Value>Input</Value>
</Attribute>
</Attribute>
</Attribute>
<ExternalInterface ID="175dc9c9-f9a3-4b10-b43e-68dfc14811fc" Name="Channel_DI_0"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Channel">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Digital</Value>
</Attribute>
<Attribute Name="IoType" AttributeDataType="xs:string">
<Value>Input</Value>
</Attribute>
<Attribute Name="Number" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="Length" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
</ExternalInterface>
<ExternalInterface ID="23e99053-906c-4548-9bd2-e975cacf01b2" Name="Channel_DI_1"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Channel">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Digital</Value>
</Attribute>
<Attribute Name="IoType" AttributeDataType="xs:string">
```

```

<Value>Input</Value>
</Attribute>
<Attribute Name="Number" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="Length" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
</ExternalInterface>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
<InternalLink Name="Link To Tag_1" RefPartnerSideA="29e0bb63-0050-46e3-968a-
fcecf4eb050a:Channel_DI_0" RefPartnerSideB="9564bcc2-8ea0-4be7-a950-5c55b34e474a:Tag_1" />
<InternalLink Name="Link To Tag_2" RefPartnerSideA="29e0bb63-0050-46e3-968a-
fcecf4eb050a:Channel_DI_0" RefPartnerSideB="9564bcc2-8ea0-4be7-a950-5c55b34e474a:Tag_2" />
<InternalLink Name="Link To Tag_3" RefPartnerSideA="29e0bb63-0050-46e3-968a-
fcecf4eb050a:Channel_DI_1" RefPartnerSideB="9564bcc2-8ea0-4be7-a950-5c55b34e474a:Tag_2" />
<InternalLink Name="Link To Tag_4" RefPartnerSideA="29e0bb63-0050-46e3-968a-
fcecf4eb050a:Channel_DI_2" RefPartnerSideB="9564bcc2-8ea0-4be7-a950-5c55b34e474a:Tag_2" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
Device" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
AutomationProject" />
</InternalElement>
</InstanceHierarchy>
</CAEXFile>

```

Variable vom Datentyp Bool (%I0.0)

Im obigen Beispiel einer AML-Datei wird beim Import die logische Adresse für die Variable mit Hilfe des folgenden Algorithmus berechnet:

$$\text{Logische Adresse} = \text{ChannelType} + \text{ByteAddress} + \text{BitAddress}$$

Name	Beschreibung
ChannelType	Eingang (E) oder Ausgang (A)
ByteAddress	ByteAddress wird wie folgt berechnet: StartAddress des E/A-Moduls * 8 + BitOffsetAddress des E/A-Moduls + (ChannelNumber * ChannelLength) / 8
BitAddress	BitAddress wird wie folgt berechnet: (BitOffsetAddress des E/A-Moduls + (ChannelNumber * ChannelLength)) % 8

Hinweis

- Wenn eine Variable modulübergreifend ist: Der Algorithmus oben liefert mehrere Byte-Adressen basierend auf der Anfangsadresse der Module sowie ein Array aus Bitadressen pro Byte-Adresse entsprechend den einzelnen Kanalnummern. Dann wählt der Algorithmus die niedrigste Byte-Adresse und die niedrigste Bitadresse im Array entsprechend diesem Byte aus, um die logische Adresse der Variablen zu berechnen. Sobald die logische Adresse der Variablen zugeordnet ist, wird die übergreifende Eigenschaft beim Import automatisch vom TIA Portal berücksichtigt.
- Im TIA Portal unterstützen nur wenige Gerätekonfigurationen (zum Beispiel ASI-Module) das BitOffset-Addressattribut. Für Module, die das BitOffset-Addressattribut nicht unterstützen, wird für obige Berechnung der Standardwert "0" angenommen.

Die folgende Liste enthält die Variablen-Datentypen, die die Bit-Adresse im TIA Portal unterstützen:

Variablen-Datentypen	Wert der Bitadresse
Bool	0 bis 7
LReal	Die höchsten und niedrigsten Werte für Bitnummern im TIA Portal sind 0.
LWord	
LInt	
ULInt	
LTime	
LDT	
LTime_Of_Day	

Wenn beim CAx-Import für einen Kanaltyp eine Boolesche Variable konfiguriert ist, umfasst die Berechnung der logischen Adresse die Bit-Offset-Adresse. Die logische Adresse zusammen mit dem Bit-Offset wird als logische Adresse der Variablen in der Oberfläche des TIA Portals aktualisiert.

Variablen mit anderen Datentypen, die die BitOffset-Adresse unterstützen

Wenn zwischen zwei verschiedenen Datentypen ein Variablenkanal konfiguriert ist, wobei ein Kanal des Typs Bool einer mehrere Kanäle überspannenden Variablen des Typs LDT zugeordnet ist, umfasst die Berechnung der logischen Adresse die Bit-Offset-Adresse für die Berechnung der logischen Adresse der Variablen des Datentyps "LDT". Die logische Adresse der Variablen wird in der Oberfläche im TIA Portal mit einem Fehler aktualisiert, wenn der Wert der Bitadresse ein anderer als "0" ist; die Kanalkonfiguration der Variablen ist dann nicht möglich.

Es muss sichergestellt werden, dass die Variablen-Kanal-Konfiguration zwischen gleichen Datentypen ausgeführt wird.

Variable mit anderem Datentyp (%IB0)

Logische Adresse = ChannelloType + TagDataType + ByteAddress

Name	Beschreibung
ChannelloType	Eingang (E) oder Ausgang (A)
TagDataType	TagDataType ist eine Abkürzung des VariablenTyps. Beispiel: W für Wort und B für Byte
ByteAddress	ByteAddress wird wie folgt berechnet: StartAddress des E/A-Moduls + (ChannelNumber * ChannelLength) / 8

Der oben beschriebene Algorithmus wird für die genaue Berechnung der logischen Adresse einer Variablen in folgenden Fällen verwendet:

- Die Länge des in der Variablen angegebenen Datentyps sollte gleich der Länge des Kanals sein, dem er zugewiesen ist.
- Wird beispielsweise eine Variable vom Datentyp "Byte" einem analogen Kanal mit einer Länge von 2 Bytes zugewiesen, muss beim Import einer AML-Datei ohne logische Adresse der Variablen die Variable im TIA Portal immer dem ersten Byte des Kanals zugewiesen werden, unabhängig davon, welchem Byte sie ursprünglich zugewiesen war.

Hinweis

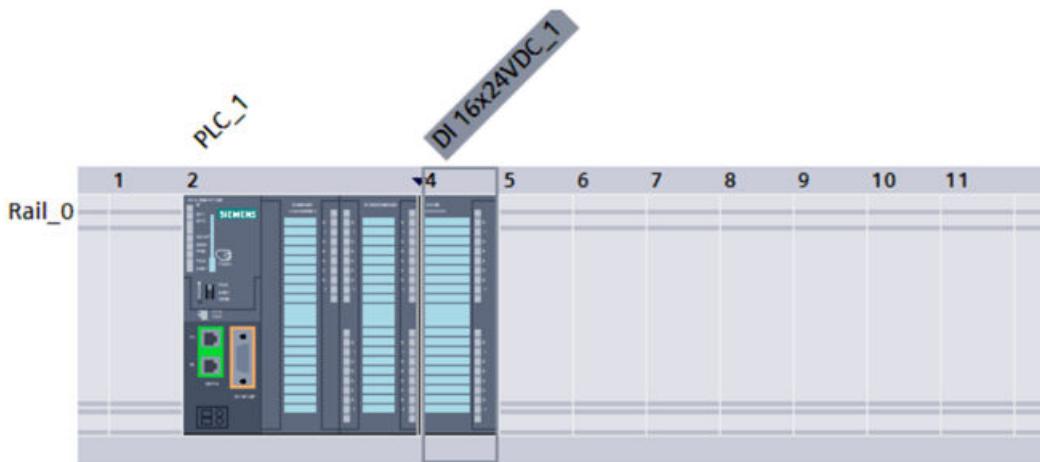
- Wenn eine Variable modulübergreifend ist: Der Algorithmus oben liefert mehrere Byte-Adressen basierend auf der Anfangsadresse der Module. Dann wählt der Algorithmus die niedrigste Byte-Adresse aus, um die logische Adresse der Variablen zu berechnen. Sobald die logische Adresse mit der niedrigsten Byte-Adresse der Variablen zugeordnet ist, wird die übergreifende Eigenschaft beim Import automatisch vom Portal gehandhabt. Wenn das Attribut StartAddress für das E/A-Modul in der AML-Datei nicht angegeben ist, wird vom TIA Portal der Standardwert zugeordnet, der dann für obige Berechnung verwendet wird.
- Wenn das Attribut StartAddress für das E/A-Modul in der AML-Datei nicht angegeben ist, wird vom TIA Portal der Standardwert zugeordnet, der dann für obige Berechnung verwendet wird.

Nach erfolgreichem Import wurde im TIA Portal die folgende Variablenkonfiguration für das oben beschriebene Beispiel erstellt.

Default tag table						
	Name	Data type	Address	Retain	Access...	Visibl...
1	Tag_1	Bool	%IO.0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
2	Tag_2	Byte	%IB0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
3	<Add new>			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

Die Kanäle sind mit den Variablen wie unten dargestellt zu konfigurieren.

8.6 Hardware-Daten importieren/exportieren



Siehe auch

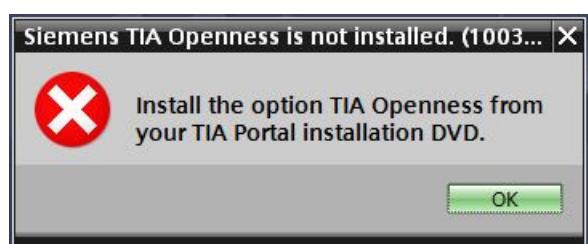
- Verbindung zum TIA Portal aufbauen (Seite 79)
- Projekt öffnen (Seite 111)

8.6.13 Ausnahmen beim Import und Export von CAx-Daten

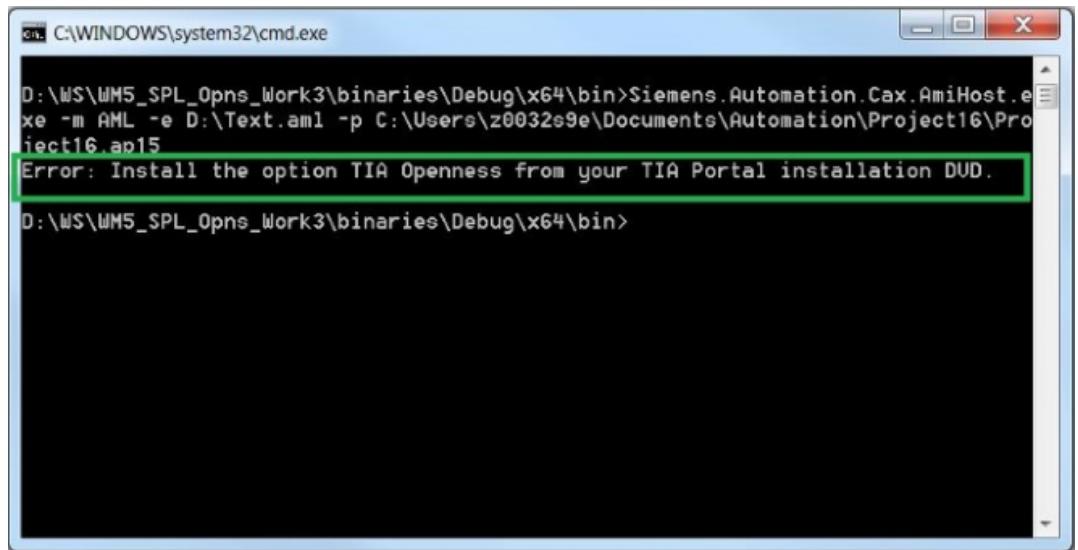
Ausnahme aufgrund von Nichtverfügbarkeit von TIA Openness

Die CAx-Implementierung basiert auf TIA Openness Public APIs. Openness Public APIs sind nur verfügbar, wenn der Anwender das Openness-Optionspaket während der Installation des TIA Portals installiert hat. Deshalb muss vor der Ausführung von CAx-bezogenen Funktionen geprüft werden, ob Openness verfügbar ist. (Siehe TIA Portal Openness installieren (Seite 30))

Immer wenn der Anwender eine CAx-Exportaktion oder eine CAx-Importaktion in der Benutzeroberfläche des TIA Portals auslöst, wird eine Prüfung durchgeführt, um zu ermitteln, ob TIA Openness im System vorhanden ist. Wenn keine Installation von TIA Openness gefunden wird, wird dem Anwender ein TIA Portal-Dialog mit der folgenden Fehlermeldung angezeigt.



Wird die CAx-Operation über die Befehlszeile ausgeführt, wird bei Nichtverfügbarkeit von TIA Openness der folgende Fehlerdialog angezeigt.



The screenshot shows a Windows Command Line window titled 'C:\WINDOWS\system32\cmd.exe'. The command entered is:
D:\WS\WM5_SPL_Opns_Work3\binaries\Debug\x64\bin>Siemens.Automation.Cax.AmiHost.exe -m AML -e D:\Text.aml -p C:\Users\z0032s9e\Documents\Automation\Project16\Project16.ap15
The output shows an error message in a red-bordered box: 'Error: Install the option TIA Openness from your TIA Portal installation DVD.' Below the error message, the command line prompt is visible again: 'D:\WS\WM5_SPL_Opns_Work3\binaries\Debug\x64\bin>'.

Bild 8-3 OpennessNotInstalled-Commandline

8.6.14 Round-Trip-Geräte und -Module

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Die PLC ist offline.

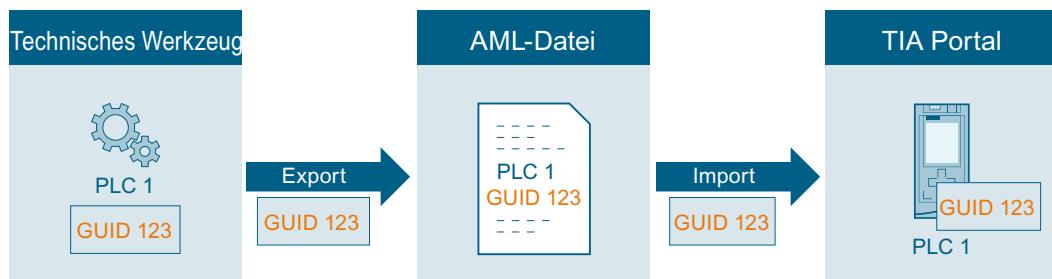
Verwendung

Sie können zwischen dem TIA Portal und anderen Engineering Tools, beispielsweise einem Elektroplanungs-Tool wie EPLAN oder dem TIA Selection Tool, Konfigurationsdaten austauschen. Zur Identifikation der importierten und exportierten Geräte wird eine eindeutige globale Kennung, die AML GUID, verwendet.

Während der Round-Trips wird die AML GUID für physische Objekte wie Geräte und Geräteelemente, die nicht integriert sind, z. B. CPUs oder Module, stabil gehalten, jedoch nicht für virtuelle Objekte wie Variablen, Kanäle usw.

Während des ersten Exports aus dem TIA Portal wird die AML GUID für ein Gerät oder ein nicht integriertes Geräteelement zufällig generiert, danach jedoch stabil gehalten.

8.6 Hardware-Daten importieren/exportieren

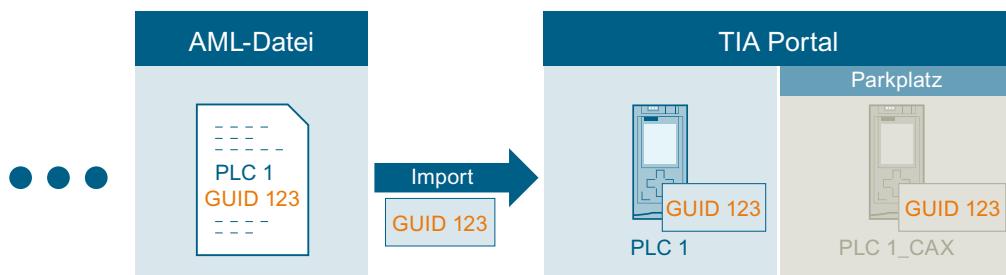


Wenn Sie ein Gerät aus einem technischen Werkzeug in ein leeres TIA Portal-Projekt exportieren, wird der AML-GUID in den Kommentar des Hardware-Objekts hinzugefügt. Wenn im TIA Portal unter "Werkzeuge > Einstellungen > CAx > Importeinstellungen" die entsprechende Einstellung aktiviert ist, wird der AML-GUID in der aktuellen Bearbeitungssprache hinzugefügt. Der Round-Trip-Prozess unterstützt nur eine Editorsprache für die AML-GUIDs. Wenn Sie Daten importieren oder exportieren, verwenden Sie stets die Bearbeitungssprache, in der Sie den Round-Trip begonnen haben.

Für alle folgenden Importe und Exporte bleibt der AML-GUID für dieses Hardware-Objekt gleich. Änderungen am Hardware-Objekt werden übernommen.

Innerhalb eines TIA Portal-Projekts müssen die Objektnamen eindeutig sein. Der Import eines Geräts oder eines Geräteelements in ein TIA Portal-Projekt, in dem ein Objekt mit dem gleichen Namen bereits vorhanden ist, würde zu einem Namenskonflikt führen. Während des Imports haben Sie die Möglichkeit, die Objekte mit den Namenskonflikten in den benutzerdefinierten Parkplatz zu verschieben. Der Name des importierten Objekts wird um "_CAX" erweitert.

Beim CAx-Export der GUI



Um den AML-Round-Trip für Vorgänger-Projektversionen zu unterstützen, werden AML-GUIDs beim Projekt-Upgrade jetzt unter "CustomIdentity" (App-ID) gespeichert und nicht mehr als "Comment" des Geräts/Geräteelements. Die AML-GUIDs des Geräts/Geräteelements müssen eindeutig sein und können in jeder beliebigen Bearbeitungssprache gespeichert werden. Es wird somit davon ausgegangen, dass alle Kommentarbearbeitungssprachen beim Projekt-Upgrade den eindeutigen AML-GUID erhalten. Der AML-GUID des Geräts/Geräteelements ist eindeutig, doch wenn ein neuer GUID mit passendem [AR_APPC:ID:] - Regex in einer beliebigen Bearbeitungssprache dem Kommentar hinzugefügt wird, gilt der erste GUID, der aus den Bearbeitungssprachen ausgewählt wird, als der AML-GUID für das Gerät/Geräteelement.

Sobald die AML-GUIDs im Kommentar in CustomIdentity verschoben werden, wird der Kommentar zum Gerät/Geräteelement durch Entfernen von [AR_APPC:ID:] aktualisiert. Das wird im folgenden Bild gezeigt.

Wenn der Kommentar mehrere Abschnitte [AR_APc:ID:] enthält, wird der erste GUID, der dem Muster entspricht, auf das CustomIdentity-Repository gesetzt und aus dem Kommentar entfernt. Der restliche Text wird als Kommentar betrachtet.

Name:	PLC_2
Author:	IC660850
Comment:	TestCPU

Hinweis

Kopieren eines importierten Geräts

Wenn Sie ein Gerät oder ein Geräteelement mit einem AML-GUID kopieren, müssen Sie den AML-GUID im Kommentar des kopierten Objekts löschen. Andernfalls existieren Geräte oder Geräteelemente mit identischen AML GUIDs in Ihrem Projekt und führen zu einer ungültigen AML-Datei.

Importeinstellungen

1. Legen Sie den Namen des Parkplatzordners unter "Optionen > Einstellungen > CAx > Einstellungen für Konfliktlösung" fest.
Der Parkplatzordner dient zum Speichern von Objekten mit Namenskonflikten.
2. Aktivieren Sie "Optionen > Einstellungen > CAx > Importeinstellungen > GUIDs beim Import speichern".

Beim Import:

- Die GUIDs eines physischen Asset werden als Bestandteil des CustomIdentity-Abschnitts gespeichert.
- Beim Importieren einer AML-Datei werden die GUIDs als Bestandteil der CustomIdentity gespeichert.

Hinweis

Gültige AML-GUIDs

Wenn Sie einen AML-GUID vor dem Import bearbeiten, wird der AML-GUID ungültig und der CAx-Import wird abgebrochen und entsprechende Informationen werden protokolliert.

Hinweis

Kommentar zu lang

Sollte der Anhang des AML GUID Kommentars die maximale Länge von 500 Zeichen überschreiten, wird der Benutzerkommentar auf 500 Zeichen begrenzt. Eine entsprechende Information wird protokolliert.

Exportieren

Beim Export:

- Der zu exportierende GUID wird für den Schlüssel AR_APPC:ID aus CustomIdentity exportiert.
- Wenn der GUID nicht als Bestandteil von CustomIdentity verfügbar ist, wird der Mehrbenutzer-GUID abgerufen und als AML GUID exportiert.
- Wenn es durch keinen der beiden obigen Schritte gelingt, den GUID eines physischen Asset bereitzustellen, wird vom Exportvorgang ein neuer zufälliger GUID erzeugt, der als AML-GUID für physische Assets behandelt wird.

AML-Struktur

Die erstellte Kennung wird wie in dem folgenden Code-Ausschnitt beschrieben in eine AML Datei exportiert:

```
<InternalElement ID="23aeef0-ce05-4116-a644-e33d43901eaf"  
Name="PLC_1"
```

8.6.15 AML importieren und dabei unbekannte Artefakte ignorieren

Voraussetzung

- Die Anwendung TIA Portal Openess ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Der PLC ist offline.

Anwendung

Es muss möglich sein, eine AML-Datei mit unbekannten Artefakten in das TIA Portal zu importieren, wobei die unbekannten Artefakte ignoriert und nur relevante Informationen aufgenommen werden. Unbekannte Artefakte enthalten unnötige Informationen, die bei CAx-Importvorgängen für das TIA Portal nicht relevant sind. Mögliche unbekannte Artefakte können Attribute, Subattribute, interne Elemente und externe Schnittstellen sein.

Hier zwei mögliche Reaktionen darauf, dass während des Importvorgangs ein unbekannter Artefakt angetroffen wird:

- Der unbekannte Artefakt ist "bekannt", aber bedeutungslos für das TIA Portal, und kann stillschweigend ignoriert werden
- Der unbekannte Artefakt ist "unbekannt" und wird ignoriert, aber durch eine Warnung gemeldet

Whitelist – Klassifizierung bekannter/unbekannter Artefakte

Mit Hilfe einer speziellen XML-Metadatei "Automation-Cax-WhiteList.xml" wird entschieden, ob ein unbekannter Artefakt für das TIA Portal bekannt oder unbekannt ist. Berücksichtigen können Sie Artefakte, die sich innerhalb eines Automatisierungsprojekts befinden. Somit wird von der Metadatei auch die Konfiguration von Artefakten innerhalb der Grenzen von Automatisierungsprojekten unterstützt.

Nachfolgend finden Sie die Typen von Rollenklassen, die für die Konfiguration unbekannter Artefakte, die ignoriert werden sollen, verfügbar sind.

Element in Whitelist	Beschreibung
ItemType	<p>Repräsentiert den Typ der Rollenklasse. Mögliche Werte für ItemType:</p> <ul style="list-style-type: none"> • AutomationProjectConfigurationRoleClassLib/AutomationProject • AutomationProjectConfigurationRoleClassLib/Device • AutomationProjectConfigurationRoleClassLib/DeviceItem • AutomationProjectConfigurationRoleClassLib/DeviceUserFolder • AutomationProjectConfigurationRoleClassLib/Subnet • AutomationProjectConfigurationRoleClassLib/CommunicationInterface • AutomationProjectConfigurationRoleClassLib/CommunicationPort • AutomationProjectConfigurationRoleClassLib/Node • AutomationProjectConfigurationRoleClassLib/IoSystem • AutomationProjectConfigurationRoleClassLib/TagTable • AutomationProjectConfigurationRoleClassLib/TagUserFolder • AutomationProjectConfigurationInterfaceClassLib/Channel • AutomationProjectConfigurationInterfaceClassLib/Tag
Name	Repräsentiert den Namen des unbekannten Attributs/Subattributs

Struktur der Whitelist

Das folgende Beispiel zeigt die Whitelist-Metadatei für Attribute:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Copyright © Siemens AG 2018. All rights reserved. -->
<WhiteList>
...
<WhiteListItem ItemType="AutomationProjectConfigurationRoleClassLib/DeviceItem">
<AttributeList>
<Attribute Name="UnknownAttribute" /> <!--This is how an attribute as white list item is
configured-->
</AttributeList>
</WhiteListItem>
...
</WhiteList>
```

8.6 Hardware-Daten importieren/exportieren

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Copyright © Siemens AG 2018. All rights reserved. -->
<WhiteList>
...
<WhiteListItem ItemType="*">
<AttributeList>
<Attribute Name="UnknownAttribute" />
<!--This is how an attribute as white list item is configured-->
</AttributeList>
</WhiteListItem>
...
</WhiteList>
```

Das folgende Beispiel zeigt die Whitelist-Metadatei für Subattribute:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Copyright © Siemens AG 2018. All rights reserved. -->
<WhiteList>
...
<WhiteListItem ItemType="AutomationProjectConfigurationRoleClassLib/DeviceItem">
<SubAttributeList>
<Attribute Name ="Address">
<Attribute Name ="2">
<SubAttribute Name="UnknownSubAttribute"/> <!--This is how sub attribute as white list item
is configured-->
</Attribute>
</Attribute>
</SubAttributeList>
</WhiteListItem>
...
</WhiteList>
```

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Copyright © Siemens AG 2018. All rights reserved. -->
<WhiteList>
...
<WhiteListItem ItemType="*">
<SubAttributeList>
<Attribute Name ="Address">
<Attribute Name ="2">
<SubAttribute Name="UnknownSubAttribute"/> <!--This is how sub attribute as white list item
is configured-->
</Attribute>
</Attribute>
</SubAttributeList>
</WhiteListItem>
...
</WhiteList>
```

Hinweis

Der Wert "*" in ItemType ist zu verwenden, um alle Vorkommen eines bestimmten Attributs/Subattributs unter allen internen Elementen/externen Schnittstellen innerhalb einer AML-Datei zu ignorieren.

Das folgende Strukturbispiel zeigt die Whitelist-Metadatei für Verknüpfungen:

```
<?xml version="1.0" encoding="utf-8" ?>
<!--Copyright @ Siemens AG 2018. All rights reserved.-->
<WhiteList>
...
<WhiteListItem ItemType="AutomationProjectConfigurationRoleClassLib/DeviceItem">
<LinkList>
<Link RefBaseClassPath ="Unknown external interface RefBaseClassPath"/> --><!--This is how
an link as white list item is configured--><!--
</LinkList>
</WhiteListItem>
...
</WhiteList>

<?xml version="1.0" encoding="utf-8" ?>
<!--Copyright @ Siemens AG 2018. All rights reserved.-->
<WhiteList>
...
<WhiteListItem ItemType="*>
<LinkList>
<Link RefBaseClassPath ="Unknown external interface RefBaseClassPath"/> --><!--This is how
an link as white list item is configured--><!--
</LinkList>
</WhiteListItem>
...
</WhiteList>
```

Hinweis

- **ItemType** repräsentiert den Typ der Rollenklasse.
 - Der Wert "*" in ItemType ist zu verwenden, um alle Vorkommen einer bestimmten externen Schnittstelle unter allen internen Elementen einer AML-Datei zu ignorieren.
 - **RefBaseClassPath** repräsentiert RefBaseClassPath einer unbekannten externen Schnittstelle.
-

8.6 Hardware-Daten importieren/exportieren

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Copyright © Siemens AG 2018. All rights reserved. -->
<WhiteList>
...
<WhiteListItem ItemType="*">
<LinkList>
<!--This is how ModuleAssignment is white list configured-->
<Link RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
ModuleAssignment" />
</LinkList>
</WhiteListItem>
...
</WhiteList>
```

Hinweis

ModuleAssignment wird an allen Stellen der AML-Datei ignoriert.

Das folgende Beispiel zeigt die Whitelist-Metadatei für Objekte:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Copyright © Siemens AG 2018. All rights reserved. -->
<WhiteList>
...
<WhiteListItem ItemType="AutomationProjectConfigurationRoleClassLib/DeviceItem">
<ObjectList>
<!--This is how an object as white list item is configured-->
<Object WhiteListKey="RefBaseClassPath"
WhiteListValue="AutomationProjectConfigurationRoleClassLib/DeviceItem"/>
</ObjectList>
</WhiteListItem>
...
</WhiteList>
```

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Copyright © Siemens AG 2018. All rights reserved. -->
<WhiteList>
...
<WhiteListItem ItemType="*">
<ObjectList>
<!--This is how an object as white list item is configured-->
<Object WhiteListKey="RefBaseClassPath"
WhiteListValue="AutomationProjectConfigurationRoleClassLib/DeviceItem"/>
</ObjectList>
</WhiteListItem>
...
</WhiteList>
```

Hinweis

- **ItemType** repräsentiert den Typ der Rollenklasse.
 - Wird für ItemType ein * eingesetzt, wird dieses interne Element an allen Stellen der AML-Datei ignoriert.
 - **WhiteListKey** repräsentiert RefBaseClassPath
 - **WhiteListValue** repräsentiert die Rollenklasse des internen Elements
-

Siehe auch

Verbindung zum TIA Portal aufbauen (Seite 79)

Projekt öffnen (Seite 111)

8.6.16 Export/Import-Topologie

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Die PLC ist offline.

Verwendung

Im TIA Portal können Sie die Geräte mit ihren Topologieinformationen in eine AML-Datei exportieren. Beim Importieren in ein leeres TIA Portal-Projekt behalten die importierten Geräteelemente die Topologieinformationen bei.

Das Element <InternalLink> liefert Verknüpfungsdetails von Port-zu-Port-Verschaltungen zwischen den Geräteelementen. Es erscheint unter dem gemeinsamen übergeordneten Objekt der verschalteten Geräte und enthält eindeutige Variablennamen.

Attribute eines Elements "InternalLink"

Die folgende Tabelle zeigt die zugehörigen Attribute der Objekte von rechnergestützten Import- und Exportdateien:

Attribut	Handha-bung	Kommentar
Name	Obligato- risch	Die Variablennamen sind als "Link to Port_n" formatiert (dabei steht n für eine Zahl von 1 bis zu der Anzahl von Port-zu-Port-Verschaltungen).
RefPartnerSideA	Obligato- risch	Bezeichnet den verknüpften Port. Formatiert als UniqueIDofPort:CommunicationPortIn- terface
RefPartnerSideB	Obligato- risch	Bezeichnet den verknüpften Port. Formatiert als UniqueIDofPort:CommunicationPortIn- terface

Beispiel: Topologiesicht



AML-Struktur

Die folgende Abbildung zeigt eine partielle Elementstruktur der exportierten AML-Datei. Sie enthält zwei eindeutige IDs für die Ports in PLCs.

```

...
<InternalElement ID="e1966b52-b8b3-47b4-8866-a754ebb77648" Name="Port_1">
  <Attribute Name="Label" Attribute DataType="xs:string">
    ...
  </Attribute>
</InternalElement>
...
<InternalElement ID="75f31daf-575f-48a2-ab35-8f07a376eb1b" Name="Port_1">
  <Attribute Name="Label" Attribute DataType="xs:string">
    ...
  </Attribute>
</InternalElement>

```

Das Element <InternalLink> enthält drei obligatorische Attribute.

```

<InternalLink Name="Link to Port_1"
  RefPartnerSideA="e1966b52-b8b3-47b4-8866-a754ebb77648:CommunicationPortInterface"
  RefPartnerSideB="75f31daf-575f-48a2-ab35-8f07a376eb1b:CommunicationPortInterface" />

```

8.6.17 Import eines Geräts mit Bibliotheksreferenzen

Voraussetzung

- Die Anwendung TIA Portal Openess ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)

Anwendung

Sie können Geräte (Stationen)/Geräteelemente (Module, Submodule) in einer AML-Datei mit Hilfe von Bibliotheksreferenzen in TIA Portal importieren.

AML-Struktur

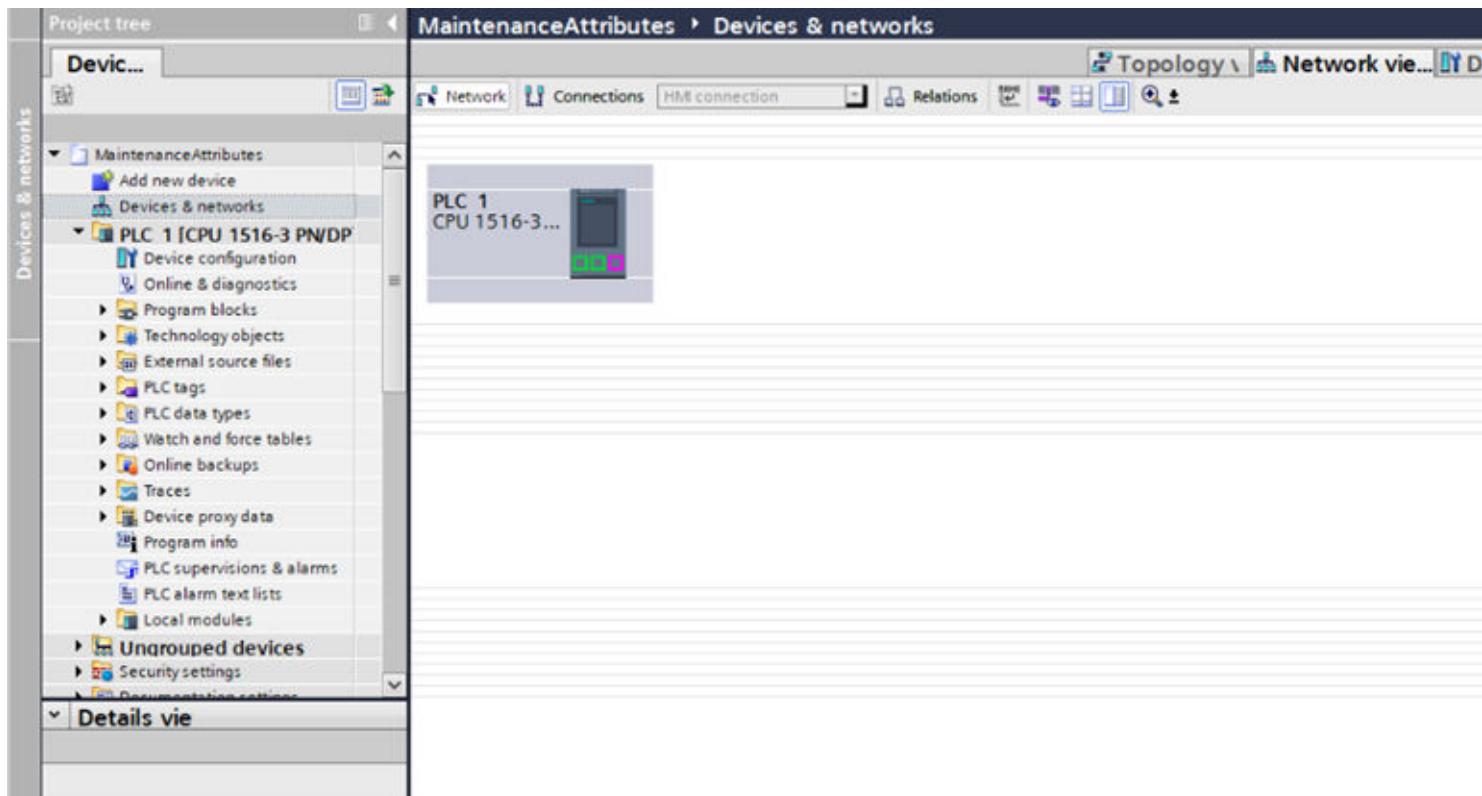
Die folgende AML-Struktur beschreibt die XML-Datei, die während des Imports in ein TIA Portal-Projekt zu verwenden ist.

```
...
<InternalElement ID="bed34f88-7a3f-4e37-a32f-df1a6dcb954a" Name="S71500/ET200MP station_1">
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>System:Device.S71500</Value>
<Attribute Name="TemplateReference" AttributeDataType="xs:string">
<Value>GlobalLib://StationPlcLibrary/Master copies/DeviceFolder/S71500ET200MP station_1</Value>
</Attribute>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
Device" />
...
<InternalElement ID="5082f437-4198-4dcb-b794-35b6b9fc104" Name="PLC_1">
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 214-1BE30-0XB0</Value>
<Attribute Name="TemplateReference" AttributeDataType="xs:string">
<Value>GlobalLib://StationPlcLibrary/Master copies/PLC_1</Value>
</Attribute>
</Attribute>
</InternalElement>
...
</InternalElement>
...
```

8.6 Hardware-Daten importieren/exportieren

Beispiel: Importierte Konfiguration

Obiger Abschnitt der AML-Datei entspricht der Station und dem PLC im TIA Portal wie unten dargestellt:



Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

8.6.18 Export eines Geräteelements

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
[Siehe Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
[Siehe Projekt öffnen \(Seite 111\)](#)
- Die PLC ist offline.

Verwendung

Das Objekt `Device` ist ein Behälterobjekt für eine zentrale oder dezentrale Konfiguration. Es ist das übergeordnete Objekt für `DeviceItem`-Objekte und das interne Element oberster Ebene der Instanzhierarchie des TIA Portal-Projekts innerhalb der Struktur einer AML-Datei.

Der CAx-Datenexport unterstützt folgende Arten von Geräten, angegeben über die AML-Typkennung:

- Physische Module
- GSD-/GSDML-basierte Geräte
- Systeme

Geräte können in ein `DeviceUserFolder` Objekt gruppiert werden.

Hinweis

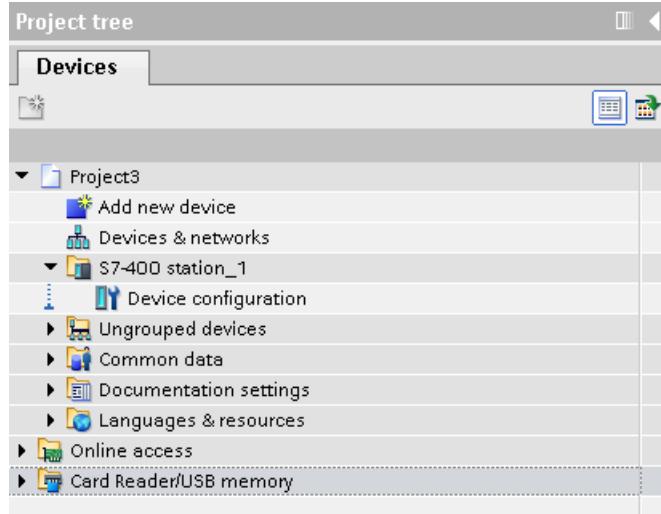
Beim Export eines Einzelgeräts werden auch alle Subnetze des Projekts exportiert.

Attribute

Die folgende Tabelle zeigt die zugehörigen Attribute des Geräteobjekts rechnergestützter Import- und Exportdateien:

Attribut	Handha-bung	Kommentar
Name	Obligato-risch	
TypeIdentifier	Obligato-risch für Export	Optional für Import ab AR APC V1.1
Comment	Optional	Standardeinstellung: „

Beispiel: Exportierte Konfiguration



AML-Struktur der Exportdatei

Das folgende Strukturbispiel stellt den Export des Einzelgeräts "S7-400 station_1" ohne Baugruppenträger und Module dar:

```
...
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
  <InternalElement ID="288b7850-688e-43b3-941e-d615ba900a02" Name="Project3">
    <Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
    <Attribute Name="ProjectSign" AttributeDataType="xs:string" />
    <Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
    <Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
    <InternalElement ID="57611cfcd-6da4-444e-ac78-5fbcea20a4e1" Name="S7-400 station_1">
      <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
        <Value>System:Device.S7400</Value>
      </Attribute>
      <Attribute Name="Comment" AttributeDataType="xs:string">
        <Value>S7400 station</Value>
      </Attribute>
      <SupportedRoleClass RefRoleClassPath=
        "AutomationProjectConfigurationRoleClassLib/Device" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
  </InternalElement>
</InstanceHierarchy>
</CAEXfile>
```

Gerät ohne Typkennung und mit generischer Typkennung

Der CAx-Import muss in der Lage sein, Geräte ohne Typkennung oder mit generischen Typkennungen ("System:Device.Generic") zu verarbeiten. Es kann möglich sein, dass die AMI-Datei Gerätetypen ohne Typkennung oder mit generischer Typkennung ('System:Device.Generic') enthält.

Aber mit dem CAx-Import können diese bearbeitet und Geräte korrekt erstellt werden.

Die folgenden Geräte unterstützen den Ersatz generischer Gerätetypen oder fehlender Typkennungen:

- GSD- und GSDML-Geräte
- MDD-basierte Geräte (keine GSD/GSDML-Geräte)

Für generische Geräte oder Geräte ohne Typkennung und mit generischem Baugruppträger und Ersatz der Typkennung muss das Kopfmodul (bei dezentralen Geräten) oder der PLC (bei zentralen Geräten) in dem in der AML-Datei beschriebenen Baugruppträger vorhanden sein, sonst werden Geräte ohne Typkennungen oder der Ersatz der Typkennung für generische Geräte und generische Baugruppträger nicht verarbeitet.

Die folgende XML-Struktur zeigt eine Gerätetypkonfiguration mit nicht-generischer Typkennung:

```
<InternalElement ID="04f5d5f08-316a-4a1d-9290-9bfd75b2b2ca" Name="S7-400 station_1">
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>System:Device.S7400</Value>
  </Attribute>
  <Attribute Name="Comment" AttributeDataType="xs:string">
    <Value>S7400 station</Value>
  </Attribute>
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Device"/>
</InternalElement>
```

Die folgende XML-Struktur zeigt eine Gerätetypkonfiguration mit generischer Typkennung:

```
<InternalElement ID="04f5d5f08-316a-4a1d-9290-9bfd75b2b2ca" Name="S7-400 station_1">
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>System:Device.Generic</Value>
  </Attribute>
  <Attribute Name="Comment" AttributeDataType="xs:string">
    <Value>S7400 Device</Value>
  </Attribute>
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Device"/>
</InternalElement>
```

Die folgende XML-Struktur zeigt eine Gerätetypkonfiguration mit Typkennung für das Gerät.

```
<InternalElement ID="a887601f-3ced-4f50-88ff-a9ec6eabb682" Name="S7-400 station_1">
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>System:Device.S7400</Value>
  </Attribute>
  <Attribute Name="Comment" AttributeDataType="xs:string">
    <Value>S7400 Device</Value>
  </Attribute>
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Device"/>
</InternalElement>
```

8.6 Hardware-Daten importieren/exportieren

Die folgende XML-Struktur zeigt eine Gerätekonfiguration, in der keine Typkennung für das Gerät verfügbar ist.

```
<InternalElement ID="a887601f-3ced-4f50-88ff-a9ec6eabb682" Name="S7-400 station_1">
  <Attribute Name="Comment" AttributeDataType="xs:string">
    <Value>S7400 Device</Value>
  </Attribute>
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
Device"/>
</InternalElement>
```

Siehe auch

[Struktur der CAx-Daten zum Import/Export \(Seite 923\)](#)

[AML-Typkennungen \(Seite 928\)](#)

8.6.19 Import eines Geräteobjekts

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
[Siehe Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
[Siehe Projekt öffnen \(Seite 111\)](#)
- Der PLC ist offline.

Verwendung

Das Objekt `Device` ist ein Behälterobjekt für eine zentrale oder dezentrale Konfiguration. Es ist das übergeordnete Objekt für `DeviceItem`-Objekte und das interne Element oberster Ebene der Instanzhierarchie des TIA Portal-Projekts innerhalb der Struktur einer AML-Datei.

Der CAx-Datenimport unterstützt folgende Arten von Geräten, angegeben über die AML-Typkennung:

- Physische Module
- GSD-/GSDML-basierte Geräte
- Systeme
- Generische Geräte

Wenn ein `DeviceUserFolder` Objekt bereits in einem TIA Portal-Projekt existiert, werden die Geräte in dem entsprechenden Ordner gruppiert.

Wenn Sie nur die Kennung (`TypeIdentifier`) eines Kopfmoduls oder eines PLC und nicht die des entsprechenden Baugruppenträgers und Geräts kennen, können Sie einen generischen Baugruppenträger importieren.

Beispiel: TypeIdentifier = System:<Prefix>.Generic

Für den Austausch generischer Geräte müssen die folgenden Elemente in dem in der AML Datei beschriebenen Baugruppträger vorhanden sein.

- Zentrale Geräte: SPS
- Dezentrale Geräte: Kopfmodul

Sind die Geräte generisch, definiert das Attribut BuiltIn die Art des Baugruppträgers oder Moduls:

- Physisch: BuiltIn = True
- Generisch: BuiltIn = False

Beispiel: Generische Geräte importieren

Das folgende Strukturbispiel stellt den Import des generischen "S7-400 station" -Geräts ohne Baugruppträger und Module dar.

```
...
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
<InternalElement ID="d4dc896a-f4a5-41b6-9c48-8d3a0a5a4343" Name="MyProject">
  <Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
  <Attribute Name="ProjectSign" AttributeDataType="xs:string" />
  <Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
  <Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
  <InternalElement ID="3e6277d1-1b12-4c18-b00e-25e3eac3ac35" Name="S7400 station_1">
    <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
      <Value>System:Device.Generic</Value>
    </Attribute>
    <Attribute Name="Comment" AttributeDataType="xs:string">
      <Value>S7400 station_1</Value>
    </Attribute>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/Device" />
  </InternalElement>
  ...
  <SupportedRoleClass RefRoleClassPath=
    "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
</InternalElement>
</InstanceHierarchy>
</CAEXFile>
```

Beispiel: Importieren der Hierarchie eines Anwenderordners eines Geräts

Das folgende Strukturbeispiel zeigt den Import einer Ordnerhierarchie.

```
...
<InternalElement ID="4fe37f4f-2661-4492-95f0-3d8a8160c851" Name="Project1">
    <Attribute Name="ProjectManufacturer" AttributeDataType="xs:string" />
    <Attribute Name="ProjectSign" AttributeDataType="xs:string" />
    <Attribute Name="ProjectRevision" AttributeDataType="xs:string" />
    <Attribute Name="ProjectInformation" AttributeDataType="xs:string" />
    <InternalElement ID="1ee1615f-9c67-432d-a7cc-b795babf67b6" Name="Group_1">
        <SupportedRoleClass RefRoleClassPath=
            "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
    </InternalElement>
    <InternalElement ID="ce14c85a-28de-41aa-ad08-2eb7d0fb755f" Name="Group_2">
        <InternalElement ID="852347e8-3c48-4eb9-8bd8-349d0c7caf34" Name="Group_3">
            <SupportedRoleClass RefRoleClassPath=
                "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
        </InternalElement>
        <SupportedRoleClass RefRoleClassPath=
            "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
    </InternalElement>
    <InternalElement ID="97cf7924-1756-4e32-8716-ac18990e4762" Name="Group_4">
        <SupportedRoleClass RefRoleClassPath=
            "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
        "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
</InternalElement>
...

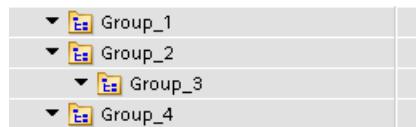
```

Importierte Anwenderordnerhierarchie

Der Name der Ordner für nicht gruppierte und nicht zugeordnete Geräte wird sprachspezifisch vergeben. Deshalb wird empfohlen, einen Import mit derselben Benutzeroberflächensprache wie beim Export durchzuführen. Andernfalls werden nicht gruppierte und nicht zugeordnete Geräte in Ordner importiert, die entsprechend der Sprache des Exports benannt sind.

Beispiel: Sie haben ein Projekt exportiert, das die Gerätesystemgruppe "Ungrouped devices" (englisch) enthält, und importieren die AML-Datei in einer deutschen Benutzeroberfläche. Das Ergebnis: Im Projekt wird eine Gerätesystemgruppe "Nicht gruppierte Geräte" (deutsch) angelegt, gleichzeitig wird beim CAx-Import eine Benutzergruppe "Ungrouped device" (englisch) erstellt.

Die folgende Hierarchie wird in die Projektnavigation importiert:



Siehe auch

[Struktur der CAx-Daten zum Import/Export \(Seite 923\)](#)

[AML-Typkennungen \(Seite 928\)](#)

8.6.20 Export/Import eines Geräts mit festgelegter Adresse

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Die PLC ist offline.

Verwendung

Im TIA Portal können Sie die Adressobjekte von IO-Geräteelementen in eine AML-Datei exportieren. Beim Importieren in ein leeres TIA Portal-Projekt behalten die importierten Geräteelemente die Adressobjekte in den entsprechenden IO-Geräteelementen bei.

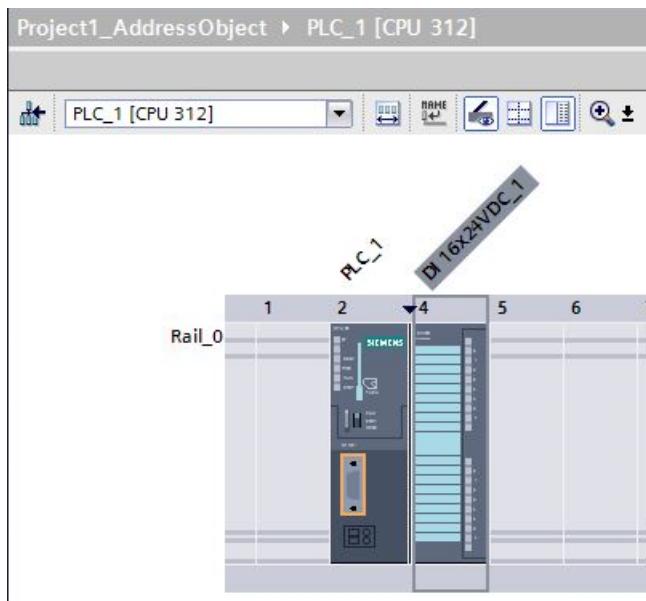
Das Attribut Address in der AML-Datei enthält RefSemantic mit der obligatorischen Einstellung auf den angegebenen Wert OrderedListType.

Attribute eines Elements "Address"

Die folgende Tabelle zeigt die zugehörigen Attribute der Objekte von rechnergestützten Import- und Exportdateien:

Attribut	Handhabung	Kommentar
Io-Type	Obligatorisch	Eingang oder Ausgang
Länge	Optional	Kanalbreite
Start-Address	Obligatorisch	Anfangsadresse des IO-Geräts

Beispiel: IO-Geräteelemente mit Adressobjekten



AML-Struktur

Die folgende Abbildung zeigt eine partielle Elementstruktur der exportierten AML-Datei. Sie enthält die Address-Elemente und deren Attribute.

```
<Attribute Name="Address">
  <RefSemantic CorrespondingAttributePath="OrderedListType" />
<Attribute Name="1">
  <Attribute Name="StartAddress" Attribute DataType="xs:int">
    <Value>0</Value>
  </Attribute>
  <Attribute Name="Length" Attribute DataType="xs:int">
    <Value>16</Value>
  </Attribute>
  <Attribute Name="IoType" Attribute DataType="xs:string">
    <Value>Input</Value>
  </Attribute>
</Attribute>
</Attribute>
```

Pruned XML

Pruning beschreibt den Vorgang der Optimierung des Inhalts durch Entfernen bestimmter Dinge, die in der XML nicht unbedingt angegeben werden müssen. In dieser reduzierten XML werden die automatisch erstellten Submodulinformationen nicht angegeben, und die entsprechenden Adressobjekte werden unter dem direkt übergeordneten Element angeordnet.

Die folgende Abbildung zeigt eine partielle Elementstruktur der exportierten AML-Datei vor dem Pruning.

```
<InternalElement ID="5511a117-42c6-44b7-be5d-0f33cd46e932" Name="AS-i Master_1">
  <Attribute Name="PositionNumber" AttributeDataType="xs:int">
    <Value>4</Value>
  </Attribute>
  <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
    <Value>True</Value>
  </Attribute>
  <Attribute Name="Address">
    <RefSemantic CorrespondingAttributePath="OrderedListType" />
    <Attribute Name="1">
      <Attribute Name="StartAddress" AttributeDataType="xs:int">
        <Value>20</Value>
      </Attribute>
      <Attribute Name="Length" AttributeDataType="xs:int">
        <Value>256</Value>
      </Attribute>
      <Attribute Name="IoType" AttributeDataType="xs:string">
        <Value>Input</Value>
      </Attribute>
    </Attribute>
  </Attribute>
```

In der Pruned-AML-Datei werden die Submodulinformationen like <InternalElement> entfernt und die entsprechenden Adressobjekte beibehalten.

```
<Attribute Name="Address">
  <RefSemantic CorrespondingAttributePath="OrderedListType" />
  <Attribute Name="1">
    <Attribute Name="StartAddress" AttributeDataType="xs:int">
      <Value>20</Value>
    </Attribute>
    <Attribute Name="Length" AttributeDataType="xs:int">
      <Value>256</Value>
    </Attribute>
    <Attribute Name="IoType" AttributeDataType="xs:string">
      <Value>Input</Value>
    </Attribute>
  </Attribute>
</Attribute>
```

Siehe auch

[Pruned AML \(Seite 920\)](#)

8.6.21 Export/Import eines Geräts mit Kanälen

Voraussetzung

- Die TIA Portal Openess-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Die PLC ist offline.

Verwendung

Im TIA Portal können Sie die Kanalobjekte von IO-Geräteelementen in eine AML-Datei exportieren. Beim Importieren in ein leeres TIA Portal-Projekt behalten die importierten Geräteelemente die Kanalobjekte in den entsprechenden IO-Geräteelementen bei.

Das Element <ExternalInterface> wird in internen Elementen von Knoten und Subnetzen dargestellt und gibt an, dass Knoten und Subnetze verbunden sind.

Attribute eines Elements "ExternalInterface"

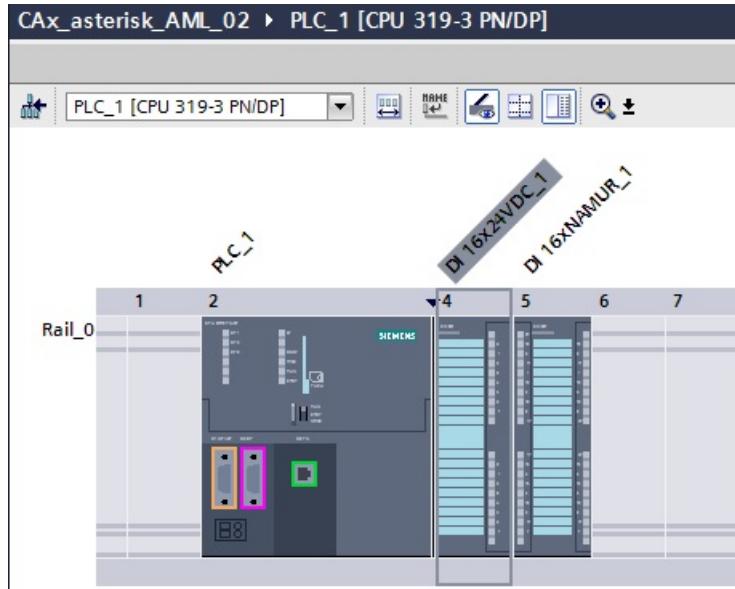
Die folgende Tabelle zeigt die zugehörigen Attribute der Objekte von rechnergestützten Import- und Exportdateien:

Attribut	Handhabung	Kommentar
Io-Type	Optional	Eingang oder Ausgang
Lengt h	Optional	Kanalbreite (1 für digitale und 16 für analoge Signale)
Num- ber	Optional	Kanalnummer beginnt bei 0
Type	Optional	Analog oder digital

Kanalnummierung

Digitaleingangs-, Digitalausgangs-, Analogeingangs-, Analogausgangs- und Technologiekanäle werden als DI_0, DO_0, AI_0, AO_0, TO_0 nummeriert. Die Kanäle auf den Geräteelementen selbst werden zuerst nummeriert, und die Kanäle auf Untergeräteelementen werden nachfolgend nummeriert (Tiefe zuerst). Jedes weitere Geräteelement hat eine eigene Kanalnummer, beginnend bei 0.

Beispiel: Geräte mit Kanälen



AML-Struktur

Die folgende Abbildung zeigt eine partielle Elementstruktur der exportierten AML-Datei.

```
<ExternalInterface ID="31ca16d3-6322-43b6-95bc-e2d7d7bfc7b7" Name="Channel_DI_0"
    RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/Channel">
    <Attribute Name="Type" AttributeDataType="xs:string">
        <Value>Digital</Value>
    </Attribute>
    <Attribute Name="IoType" AttributeDataType="xs:string">
        <Value>Input</Value>
    </Attribute>
    <Attribute Name="Number" AttributeDataType="xs:int">
        <Value>0</Value>
    </Attribute>
    <Attribute Name="Length" AttributeDataType="xs:int">
        <Value>1</Value>
    </Attribute>
</ExternalInterface>
```

8.6.22 Export von Geräteelementobjekten

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Die PLC ist offline.

Verwendung

Der Export von Geräteelementobjekten ist nur auf PLC-Geräte anwendbar.

DeviceItem-Objekte sind verschachtelte untergeordnete Elemente eines Device-Objekts:
Ein Objekt vom Typ DeviceItem kann ein Baugruppenträger oder ein Einschubmodul sein.

- Das erste Nachfolgeelement eines Geräts ist vom Typ „Baugruppenträger“.
Die PositionNumber eines Baugruppenträgers beginnt bei 0. Gibt es mehrere Baugruppenträger, werden diese fortlaufend nummeriert (1, 2, 3, ...).
Es gibt keine Einschränkungen bezüglich der Reihenfolge innerhalb eines Hierarchielevels in der AML Datei.
- Alle weiteren Nachfolger des Typs „Baugruppenträger“ sind Module.

Der CAx-Datenexport unterstützt folgende Arten von Geräteelementen, angegeben über die AML-Typkennung:

- Physische Module
- GSD/GSDML-Module

Attribute

Die folgende Tabelle zeigt die zugehörigen Attribute der Objekte von rechnergestützten Import- und Exportdateien:

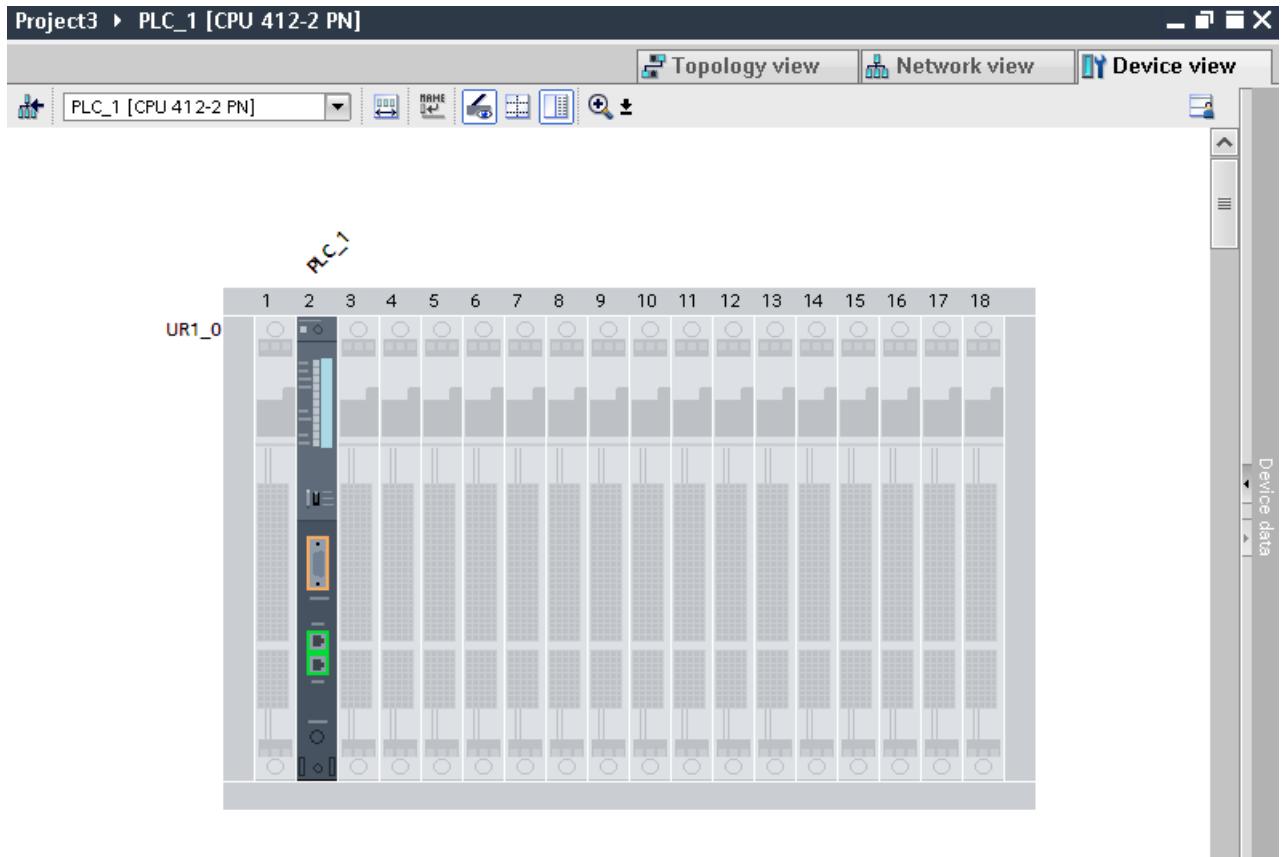
Attribut	Handhabung	Kommentar
Name	Obligatorisch Nur Exportieren für „BuiltIn“ = TRUE	
TypeName	Nur Exportieren für „BuiltIn“ = FALSE	
DeviceItemType	Nur Exportieren	Nur für PLCs (Zentralgeräte) und Geräteelemente (physische Baugruppenträger, Module, Kopfmodul). Optional beim Import, doch jedes Geräteelement außer Basiseinheiten mit DeviceItemType als Zubehör wird stillschweigend ignoriert.
PositionNumber	Obligatorisch	
BuiltIn	Optional	Standardeinstellung: FALSE

Attribut	Handhabung	Kommentar
TypeIdentifier	Obligatorisch für „BuiltIn“ = FALSE Ignoriert für „BuiltIn“ = TRUE	Bei integrierten Geräteelementen wird dieses Attribut mit der Typkennung seines steckbaren übergeordneten Elements exportiert. Es hat keine Relevanz beim Import und ist somit optional. Bei nicht integrierten Geräteelementen ist dieses Attribut optional.
FirmwareVersion	Optional, obligatorisch wenn das Objekt Firmware-Versionen unterstützt	
PlantDesignation IEC	Optional	Standardeinstellung: „“
LocationIdentifier IEC	Optional	Standardeinstellung: „“
Comment	Optional für „BuiltIn“ = FALSE	Standardeinstellung: „“
ProductDesignation IEC	Obligatorisch wenn das Objekt dieses Attribut in TIA Portal unterstützt und nicht leer ist	Das Importieren wird nicht unterstützt, wenn die Länge des ProductDesignation IEC größer als 54 Zeichen ist. Export/Import unter Geräteelementen als Teil des AR APC V1.1.0 Supports für TIA Portal V16
InstallationDate	Obligatorisch wenn das Objekt dieses Attribut in TIA Portal unterstützt	Export/Import unter Geräteelementen als Teil des AR APC V1.1.0-Supports für TIA Portal V16
Address	Optional	"Address" besitzt verschachtelte Attribute

Die folgende Tabelle zeigt die verschachtelten Attribute der „Address“ Attribute des Objekts „DeviceItem“:

Attribute für „Address“	Handhabung	Kommentar
StartAddress	Obligatorisch	
Length	Nur Exportieren	Der Export/Import von Adressen mit der Länge = 0 wird nicht unterstützt.
IoType	Obligatorisch	Eingang oder Ausgang

Beispiel: Exportierte Konfiguration



AML-Struktur der Exportdatei

Das folgende Strukturbispiel stellt den Export von „UR1_0“ und dem Modul „PLC_1“ dar.

8.6 Hardware-Daten importieren/exportieren

```

<InternalElement ID="7624ed42-3db7-4ba5-8726-e719d7b09969" Name="S7-400_station_1">
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>System:Device.S7400</Value>
  </Attribute>
  <Attribute Name="Comment" AttributeDataType="xs:string">
    <Value>S7400 station</Value>
  </Attribute>
<InternalElement ID="1fff247e6-6b94-42a2-bcd5-d673fc6e9ba5" Name="UR1_0">
  <Attribute Name="TypeName" AttributeDataType="xs:string">
    <Value>UR1</Value>
  </Attribute>
  <Attribute Name="PositionNumber" AttributeDataType="xs:int">
    <Value>0</Value>
  </Attribute>
  <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
    <Value>False</Value>
  </Attribute>
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>OrderNumber:6ES7 400-1TA01-0AA0</Value>
  </Attribute>
  <Attribute Name="Comment" AttributeDataType="xs:string">
    <Value>S7 400 rack</Value>
  </Attribute>
<InternalElement ID="202421bf-a4b9-4399-a563-9f154f0eabab" Name="PLC_1">
  <Attribute Name="TypeName" AttributeDataType="xs:string">
    <Value>CPU 412-2 PN</Value>
  </Attribute>
  <Attribute Name="DeviceItemType" AttributeDataType="xs:string">
    <Value>CPU</Value>
  </Attribute>
  <Attribute Name="PositionNumber" AttributeDataType="xs:int">
    <Value>2</Value>
  </Attribute>
  <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
    <Value>False</Value>
  </Attribute>
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>OrderNumber:6ES7 412-2EK06-0AB0</Value>
  </Attribute>
  <Attribute Name="Comment" AttributeDataType="xs:string">
    <Value>S7 400 plc</Value>
  </Attribute>
  <Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
    <Value>V6.0</Value>
  </Attribute>
  <Attribute Name="ProductDesignation IEC" AttributeDataType="xs:string">
    <Value>Additional Information</Value>
  </Attribute>
  <Attribute Name="InstallationDate" AttributeDataType="xs:dateTime">
    <Value>2018-11-26T05:09:02.803Z</Value>
  </Attribute>
  <Attribute Name="PlantDesignation IEC" AttributeDataType="xs:string">
    <Value>PD</Value>
  </Attribute>
  <Attribute Name="LocationIdentifier IEC" AttributeDataType="xs:string">
    <Value>LI</Value>
  </Attribute>
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Device" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/AutomationProject" />
</InternalElement>

```

Siehe auch

Export/Import von GSD/GSDML-basierten Geräten und Geräteelementen (Seite 994)

Struktur der CAx-Daten zum Import/Export (Seite 923)

AML-Typkennungen (Seite 928)

8.6.23 Import von Geräteelementobjekten

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal aufbauen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Projekt öffnen (Seite 111)
- Die PLC ist offline.

Verwendung

Der Import von Geräteelementobjekten ist nur auf PLC-Geräte anwendbar.

DeviceItem-Objekte sind verschachtelte untergeordnete Elemente eines Device-Objekts:
Ein Objekt vom Typ DeviceItem kann ein Baugruppenträger oder ein Einschubmodul sein.

- Das erste Nachfolgeelement eines Geräts ist vom Typ Baugruppenträger.
Die PositionNumber eines Baugruppenträgers beginnt bei 0. Gibt es mehrere Baugruppenträger, werden diese fortlaufend nummeriert (1, 2, 3, ...).
Es gibt keine Einschränkungen bezüglich der Reihenfolge innerhalb eines Hierarchielevels in der AML Datei.
- Alle weiteren Nachfolger des Typs Baugruppenträger sind Module.

Der CAx-Datenimport unterstützt folgende Arten von Geräteelementen, angegeben über die AML-Typkennung:

- Physische Module
- GSD/GSDML-Module
- Generische Module

Wenn Sie nur die Kennung (`TypeIdentifier`) eines Kopfmoduls oder eines PLC und nicht die des entsprechenden Baugruppenträgers und Geräts kennen, können Sie einen generischen Baugruppenträger importieren.

Beispiel: `TypeIdentifier = System:Rack.Generic`

Für den Austausch generischer Baugruppenträger müssen die folgenden Elemente in dem in der AML Datei beschriebenen Baugruppenträger vorhanden sein:

- Zentrale Geräte: SPS
- Dezentrale Geräte: Kopfmodul

8.6 Hardware-Daten importieren/exportieren

Ein generischer Baugruppenträger stammt aus dem Device-Typ. Demnach kann eine AML-Datei, die in das TIA Portal importiert wird, die Typkennung dieses Baugruppenträgers verwenden:

In diesem Fall bestimmt das TIA Portal die Typkennung für den Baugruppenträger.

Sind Baugruppenträger und Module generisch, definiert das Attribut `BuiltIn` die Art des Baugruppenträgers oder Moduls:

- **Physisch:** `BuiltIn = True`
- **Generisch:** `BuiltIn = False`

Einschränkungen

Während des Imports ist das Attribut `DeviceItemType` nicht relevant und ist optional.

Hinweis

Attribut "FirmwareVersion"

Wenn in der Importdatei keine `FirmwareVersion` angegeben ist, wird beim CAx-Import die letzte Firmwareversion verwendet, die im TIA Portal verfügbar ist.

Wenn das Attribut `FirmwareVersion` in der Importdatei mit einem leeren Wert vorhanden ist, schlägt der Import des Geräteelements fehl und es wird eine Fehlermeldung protokolliert.

Beispiel: Generische Geräte importieren

Das folgende Strukturbispiel stellt den Import des generischen Baugruppenträgers „Rack_1“ dar.

```

...
<InternalElement ID="6563466e-2de9-42ca-951d-eb8f2545958d" Name="S7-400_station_1">
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>System:Device.S7400</Value>
  </Attribute>
  <InternalElement ID="96930368-14ec-43e2-b9b7-c1fefc4b0534" Name="UR1_0">
    <Attribute Name="TypeName" AttributeDataType="xs:string">
      <Value>UR1</Value>
    </Attribute>
    <Attribute Name="PositionNumber" AttributeDataType="xs:int">
      <Value>0</Value>
    </Attribute>
    <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
      <Value>False</Value>
    </Attribute>
    <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
      <Value>System:Rack.Generic</Value>
    </Attribute>
    <InternalElement ID="a1de449e-4f89-45af-8bbc-f77c28bccd04" Name="PLC_1">
      <Attribute Name="TypeName" AttributeDataType="xs:string">
        <Value>CPU 412-2 PN</Value>
      </Attribute>
      <Attribute Name="DeviceItemType" AttributeDataType="xs:string">
        <Value>CPU</Value>
      </Attribute>
      <Attribute Name="PositionNumber" AttributeDataType="xs:int">
        <Value>2</Value>
      </Attribute>
      <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
        <Value>False</Value>
      </Attribute>
      <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
        <Value>OrderNumber:6ES7 412-2EK06-0AB0</Value>
      </Attribute>
      <Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
        <Value>V6.0</Value>
      </Attribute>

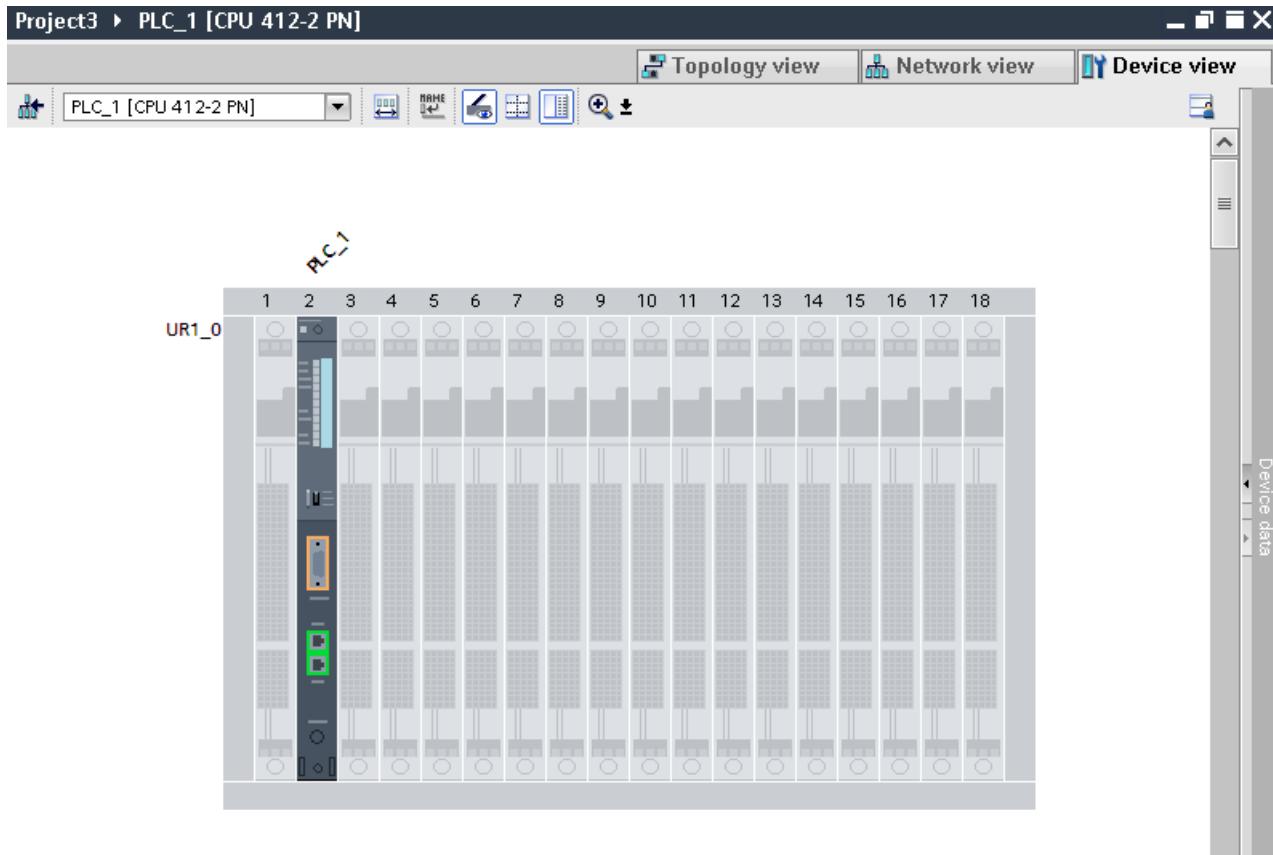
      <SupportedRoleClass RefRoleClassPath=
        "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath=
    "AutomationProjectConfigurationRoleClassLib/Device" />
</InternalElement>
...

```

8.6 Hardware-Daten importieren/exportieren

Importierte Konfiguration

Die folgende Abbildung zeigt die importierte Konfiguration in der TIA Portal-Nutzerschnittstelle:



Siehe auch

Struktur der CAx-Daten zum Import/Export (Seite 923)

AML-Typkennungen (Seite 928)

8.6.24 Export/Import von GSD/GSDML-basierten Geräten und Geräteelementen

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Die PLC ist offline.

Verwendung

Der CAx-Import/Export von GSD/GSDML-basierten Geräten und Geräteelementen ist mit dem Import/Export von Standardgeräten vergleichbar.

Bei GSD/GSDML-basierten Geräten und Geräteelementen weichen die exportierbaren Attribute voneinander ab, z. B. bei GSD/GSDML ist das Attribut "Label" vorhanden.

Generischer Import von Geräten und Baugruppenträgern ist möglich. Verwenden Sie für den Import die gleiche Kennung wie für Standardgeräte:

- Generische Geräte importieren: TypeIdentifier = System:Device.Generic
- Generische Baugruppenträger importieren: TypeIdentifier = System:Rack.Generic

Sind die Geräte generisch, definiert das Attribut BuiltIn die Art:

- Physisch: BuiltIn = True
- Generisch: BuiltIn = False

Attribute für ein Gerät

Die folgende Tabelle zeigt die zugehörigen Attribute von Geräten von CAx-Import- und -Exportdateien:

Attribut	Handhabung von Attribut	Kommentar
Name	Obligatorisch für Export und Import	
TypeIdentifier	Obligatorisch für Export und optional für Import ab AR APC V1.1	
Comment	Optional für Import	

Attribute für ein Geräteelement

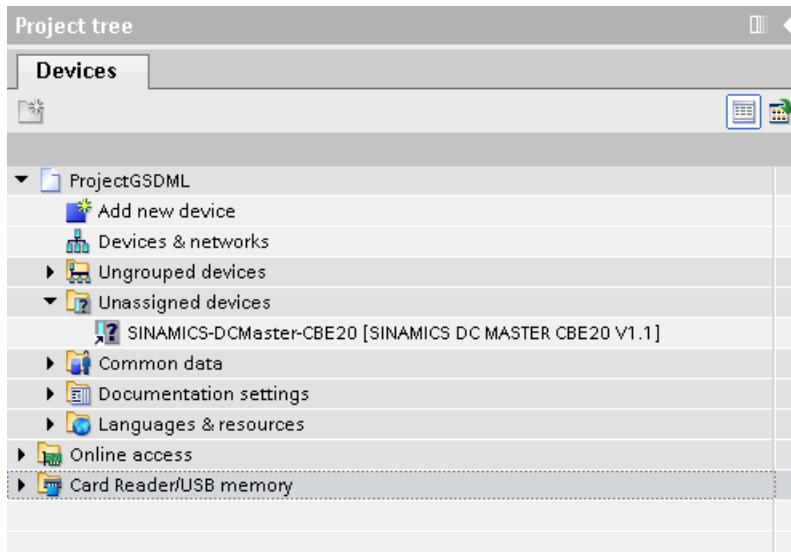
Die folgende Tabelle zeigt die zugehörigen Attribute eines Geräteelements von CAx-Import- und -Exportdateien:

Attribut	Handhabung von Attribut Integriert = FALSCH Generische Geräteelemente	Handhabung von Attribut Integriert = WAHR Physische Geräteelemente	Kommentar
Name	Obligatorisch	Nur Exportieren	
TypeName	Nur Exportieren	-/-	
DeviceItem-Type	Nur Exportieren	Nur Exportieren	Nur für SPS- (zentrale Geräte) und Kopfmodul- (dezentrale Geräte) Geräteelemente Optional beim Import, doch jedes Geräteelement außer Basiseinheiten mit DeviceItemType als Zubehör wird ignoriert.

8.6 Hardware-Daten importieren/exportieren

Attribut	Handhabung von Attribut Integriert = FALSCH Generische Geräteelemente	Handhabung von Attribut Integriert = WAHR Physische Geräteelemente	Kommentar
PositionNumber	Obligatorisch	Obligatorisch für Export Ausnahmefälle: Geräteelemente vom Typ Schnittstelle: Optional für Import Geräteelemente vom Typ Port: Optional	
BuiltIn	Optional		Standardeinstellung: FALSE
TypeIdentifier	Obligatorisch für „BuiltIn“ = FALSE	Ignoriert für „BuiltIn“ = TRUE	Bei integrierten Geräteelementen wird dieses Attribut mit der Typkennung seines steckbaren übergeordneten Elements exportiert. Das Attribut hat beim Import keine Relevanz und ist somit optional. Bei nicht integrierten Geräteelementen hat dieses Attribut keine Relevanz.
Comment	Optional	-	
Label	-	- Geräteelemente vom Typ Schnittstelle: Obligatorisch Geräteelemente vom Typ Port: Obligatorisch	

Beispiel: Exportiertes GSD-/GSDML-Gerät



AML-Struktur der Exportdatei

Die folgende Abbildung zeigt die Struktur der exportierten AML-Datei.

```
...
<InternalElement ID="9ae02cde-dfb4-4d43-a649-68b9ede7fc3d" Name="Ungrouped devices">
  <InternalElement ID="12d4ce0f-346d-4bfa-b139-c9d0db64c794" Name="GSD_device_1">
    <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
      <Value>GSD:GSDML-V2.31-SIEMENS-SINAMICS_DCMASTER-20140704.XML/D</Value>
    </Attribute>
    <InternalElement ID="ccb1cb62-67b2-4b8c-951f-10c7ffb4d787" Name="Rack">
      <Attribute Name="TypeName" AttributeDataType="xs:string">
        <Value>Rack</Value>
      </Attribute>
    ...
    <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
      <Value>GSD:GSDML-V2.31-SIEMENS-SINAMICS_DCMASTER-20140704.XML/R/IDD_14</Value>
    </Attribute>
    <InternalElement ID="74f25b5c-0c09-46d0-9011-f341a3e98a0d" Name="SINAMICS_DCMaster-CBE20">
      <Attribute Name="TypeName" AttributeDataType="xs:string">
        <Value>SINAMICS DC MASTER CBE20 V1.1</Value>
      </Attribute>
      <Attribute Name="DeviceItemType" AttributeDataType="xs:string">
        <Value>HeadModule</Value>
      </Attribute>
      <Attribute Name="PositionNumber" AttributeDataType="xs:int">
        <Value>0</Value>
      </Attribute>
      <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
        <Value>False</Value>
      </Attribute>
      <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
        <Value>GSD:GSDML-V2.31-SIEMENS-SINAMICS_DCMASTER-20140704.XML/DAP/IDD_14</Value>
      </Attribute>
      <InternalElement ID="94f34bb9-fe47-4904-b8a1-62e8fb6b1b74"
        Name="SINAMICS DC MASTER CBE20 V1.1">
        <Attribute Name="PositionNumber" AttributeDataType="xs:int">
          <Value>0</Value>
        </Attribute>
        <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
          <Value>True</Value>
        </Attribute>
        <SupportedRoleClass RefRoleClassPath=
          "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
      </InternalElement>
    ...
    <SupportedRoleClass RefRoleClassPath=
      "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
  </InternalElement>
  <SupportedRoleClass RefRoleClassPath=
    "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath=
  "AutomationProjectConfigurationRoleClassLib/Device" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath=
  "AutomationProjectConfigurationRoleClassLib/DeviceUserFolder" />
</InternalElement>
...

```

Generischer und nicht-generischer Baugruppenträger mit und ohne TypeIdentifier

Der CAx-Import muss in der Lage sein, Geräte ohne Typkennung oder mit generischen Typkennungen, d. h. 'System:Device.Generic', und Baugruppenträger mit generischen Typkennungen, d. h. 'System:Rack.Generic', zu verarbeiten.

Während des Importierens kann es möglich sein, dass die AML-Datei Gerätetypen ohne Typkennung oder mit generischer Typkennung, d. h. 'System:Device.Generic', und Baugruppenträger-Geräteelemente mit generischer Typkennung, d. h. 'System:Rack.Generic', enthält. Aber mit dem CAx-Import können diese bearbeitet sowie Geräte und Baugruppenträger-Geräteelemente korrekt erstellt werden.

Die folgenden Geräte unterstützen den Ersatz generischer Geräte, von Geräten ohne Typkennung und generischer Baugruppenträger:

- GSD- und GSXML-Geräte - Alle GSD- und GSXML-Geräte mit GSD/GSXML-Baugruppenträgern.
- MDD-basierte Geräte (keine GSD/GSXML-Geräte) - Geräte mit Systembaugruppenträger-Typkennungen.

Für die Verarbeitung generischer Baugruppenträger ist der CAx-Export nicht relevant. CAx exportiert immer Typkennungen für nicht-generische Baugruppenträger.

Für generische Geräte oder Geräte ohne Typkennung und mit generischem Baugruppenträger und Ersatz der Typkennung muss das Kopfmodul (bei dezentralen Geräten) oder der PLC (bei zentralen Geräten) in dem in der AML-Datei beschriebenen Baugruppenträger vorhanden sein, sonst werden Geräte ohne Typkennungen oder der Ersatz der Typkennung für generische Geräte und generische Baugruppenträger nicht verarbeitet.

Die folgende XML-Struktur zeigt eine GSXML-Gerätekonfiguration mit Gerät (mit Typkennung) und Baugruppenträger (mit Typkennung):

```
<InternalElement ID="bc3b50fa-5cfa-4bf3-a496-8e46080d4f86" Name="GSD device_1">
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>GSD:GSXML-V2.32-SIEMENS-SINAMICS_DCMASTER-20160531.XML/D</Value>
</Attribute>
<InternalElement ID="c80f2d97-66c9-4f31-bf6b-17e3e0de0509" Name="Rack">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>Rack</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>GSD:GSXML-V2.32-SIEMENS-SINAMICS_DCMASTER-20160531.XML/R/IDD_14</Value>
</Attribute>
<InternalElement ID="f519b4b9-b1f7-4011-bed2-25be85c8c2a8" Name="SINAMICS-DCMaster-CBE20">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>SINAMICS DC MASTER CBE20 V1.1</Value>
</Attribute>
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>HeadModule</Value>
</Attribute>
```

Die folgende XML-Struktur zeigt eine GSDML-Gerätekonfiguration mit Gerät (ohne Typkennung) und Baugruppenträger (generische Typkennung):

```
<InternalElement ID="bc3b50fa-5cfa-4bf3-a496-8e46080d4f86" Name="GSD device_1">
<InternalElement ID="c80f2d97-66c9-4f31-bf6b-17e3e0de0509" Name="Rack">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>Rack</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>System:Rack.Generic</Value>
</Attribute>
<InternalElement ID="f519b4b9-b1f7-4011-bed2-25be85c8c2a8" Name="SINAMICS-DCMaster-CBE20">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>SINAMICS DC MASTER CBE20 V1.1</Value>
</Attribute>
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>HeadModule</Value>
</Attribute>
```

8.6 Hardware-Daten importieren/exportieren

Die folgende XML-Struktur zeigt eine GSDML-Gerätekonfiguration mit Gerät (generische Typkennung) und Baugruppträger (generische Typkennung):

```
<InternalElement ID="bc3b50fa-5cfa-4bf3-a496-8e46080d4f86" Name="GSD device_1">
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>System:Device.Generic</Value>
</Attribute>
<InternalElement ID="c80f2d97-66c9-4f31-bf6b-17e3e0de0509" Name="Rack">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>Rack</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>0</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>System:Rack.Generic</Value>
</Attribute>
<InternalElement ID="f519b4b9-b1f7-4011-bed2-25be85c8c2a8" Name="SINAMICS-DCMaster-CBE20">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>SINAMICS DC MASTER CBE20 V1.1</Value>
</Attribute>
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>HeadModule</Value>
</Attribute>
</Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>GSD:GSDML-V2.32-SIEMENS-SINAMICS_DCMASTER-20160531.XML/DAP/IDD_14</Value>
</Attribute>
```

Siehe auch

- [AML-Typkennungen \(Seite 928\)](#)
- [Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)
- [Projekt öffnen \(Seite 111\)](#)

8.6.25 Exportieren/Importieren der Gerätekonfiguration mit virtueller Schnittstelle

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Die PLC ist offline.

Anwendung

Im TIA Portal werden Ports für die Kommunikation zwischen den Geräten verwenden und sind unter den Schnittstellen zu finden. Es gibt jedoch bestimmte Arten von Geräten, bei denen sich die Ports direkt unter den Geräteelementen befinden, bei denen es sich nicht um Schnittstellen handelt. Das entspricht nicht dem AML-Standard, der vorgibt, dass Ports immer unter den Schnittstellen zu finden sind.

CAx verwendet eine imaginäre Schnittstelle, die als virtuelle Schnittstelle bezeichnet wird, um die Gerätekonfiguration zu exportieren und zu importieren, wenn sich die Ports direkt unter Geräteelementen befinden, die keine Schnittstellen sind.

8.6 Hardware-Daten importieren/exportieren

Export einer AML-Datei

Das folgende Beispiel zeigt eine AML-Datei mit virtueller Schnittstelle:

```
<InternalElement ID="822622a0-056a-494c-a802-2463c5e1b47d" Name="SCALANCE interface_1">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<InternalElement ID="5a604a57-bc2d-4763-8df0-7d2000faf1b" Name="VirtualInterface_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Ethernet</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<InternalElement ID="3c4862a1-b8ed-4610-88f3-29dc8328e748" Name="Port_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="220dd49d-8d23-44b1-bdc1-878516540313" Name="Port_2">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P2</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>2</Value>
</Attribute> <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<ExternalInterface ID="99c6253b-c546-4720-af54-92db926b8231"
Name="CommunicationPortInterface"
RefBaseClassPath="AutomationProjectConfigurationInterfaceClassLib/
CommunicationPortInterface" />
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
```

Die virtuelle Schnittstelle wird bis TIA Portal V16 mit den folgenden Attributen exportiert:

- Name: ScalanceInterface_1
- Bezeichnung: Switch

Ab TIA Portal V16 wird die virtuelle Schnittstelle mit den folgenden Attributen exportiert:

- Name: VirtualInterface_1
- Bezeichnung: X1
- Typ: Ethernet

Importieren

CAx unterstützt den Import von virtuellen Schnittstellen. Dann werden die Ports unter der virtuellen Schnittstelle in der AML-Datei verarbeitet, unter dem richtigen übergeordneten Element im TIA Portal. Hier wird jede Schnittstelle mit der Bezeichnung "Switch" als virtuelle Schnittstelle betrachtet. Ab TIA Portal V16 ist die Bezeichnung jedoch keine Kennung mehr, da sie in X1 geändert wurde, was einer echten Schnittstelle entspricht.

Für das Typattribut wurde für die virtuelle Schnittstelle trotzdem "Ethernet" festgelegt, es ist optional. Ab TIA Portal V16 werden also alle Schnittstellenelemente in der AML-Datei, die nicht als "Switch" bezeichnet sind, auf allgemeine Weise behandelt, und CAx versucht bei jedem Vorkommen in der AML-Datei den Abruf aus dem TIA Portal.

Bei virtuellen Schnittstellen misslingt CAx die Suche, doch die darin enthaltenen Ports werden verarbeitet. Findet CAx jedoch eine echte Schnittstelle nicht, werden die darin enthaltenen Ports nicht verarbeitet.

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

8.6.26 Exportieren/Importieren von Subnetzen

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe [Verbindung zum TIA Portal herstellen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
Siehe [Öffnen eines Projekts \(Seite 111\)](#)
- Die PLC ist offline.

AML-Struktur

Subnetze beschreiben ein physisches Netzwerk, insbesondere, welche Geräte an dem gleichen Netzwerk vom Typ PROFIBUS, PROFINET, MPI oder ASI angeschlossen sind.

8.6 Hardware-Daten importieren/exportieren

Die Verbindungen zwischen einem Netzwerk und den Geräteelementen werden als Referenz für das Netzwerkobjekt abgebildet. Es gibt keine Referenz von den Netzwerkobjekten zu den verbundenen Geräteelementen. Die Netzwerkparameter werden im Netzwerkobjekt gespeichert. Die Parameter, die eine Netzwerkschnittstelle eines bestimmten Geräteelements betreffen, das mit einem Netzwerk verbunden ist, werden in einem Netzwerknotenobjekt in diesem Geräteelement gespeichert. Die Kommunikation wird oft mit „Kanälen“, „Ports“ und „Schnittstellen“ geregelt.

Subnetze werden als interne Elemente mit der Rollenklassifizierung „Subnet“ in der Instanzhierarchie in der AML-Datei exportiert.

Ein Subnetz hat folgende verbundene Elemente in der AML-Struktur:

- Internes Element mit der Rollenklassifizierung „Knoten“
Legt die Schnittstelle auf einem Geräteelement fest.
- <InternalLink>
Legt die verbundenen Partner des Subnetzes fest. Der <InternalLink> Variablenname ist einzigartig und wird immer unter dem internen Element des Projekts in der AML-Datei hinzugefügt.

```
.....
<SupportedRoleClass RefRoleClassPath=
  "AutomationProjectConfigurationRoleClassLib/AutomationProject" />
<InternalLink Name="Link To Port_1"
  RefPartnerSideA="1e3e4c5b-04c1-4d2c-9aee-cad53cc92dba:CommunicationPortInterface"
  RefPartnerSideB="d45aa36a-a7f2-4862-a266-d6727b9cf75:CommunicationPortInterface" />
<InternalLink Name="Link To Subnet_1"
  RefPartnerSideA="beb4eb8e-1a45-45ce-a703-1acfaf73e5f3:LogicalEndPoint_Node"
  RefPartnerSideB="1062a384-d3ca-4183-9ac2-0934a5ab7286:LogicalEndPoint_Subnet" />
<InternalLink Name="Link To Subnet_2"
  RefPartnerSideA="a3e85a0ed-580a-45c8-943e-da7de8280b7c:LogicalEndPoint_Node"
  RefPartnerSideB="1062a384-d3ca-4183-9ac2-0934a5ab7286:LogicalEndPoint_Subnet" />
</InternalElement>
</InstanceHierarchy>
</CAEXfile>
```

- <ExternalInterface>
Stellt in internen Elementen von Knoten und Subnetzen fest, dass Knoten und Subnetze verbunden sind. Wenn die Knoten oder Subnetze nicht verbunden sind, existieren die <ExternalInterface> Elemente für Knoten und Subnetz nicht.

```
...
<InternalElement ID="1062a384-d3ca-4183-9ac2-0934a5ab7286" Name="PN/IE_1">
  <Attribute Name="Type" AttributeDataType="xs:string">
    <Value>Ethernet</Value>
  </Attribute>
  <ExternalInterface ID="55ecf2cb-e72a-4974-8ed0-1d4e1ade3509"
    Name="LogicalEndPoint_Subnet"
    RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Subnet" />
</InternalElement>
... 
```

Verwendung

In TIA Portal V16 unterstützt der CAx-Import Ethernet und Profinet und der Export erfolgt für den Subnetztyp und die Knoten-Attribute immer über Ethernet.

Das rechnergestützte Importieren/Exportieren unterstützt folgende Arten von Subnetzen:

- Ethernet
- PROFIBUS
- MPI
- ASi

Hinweis

- Der CAx-Import unterstützt Ethernet und Profinet für die Attribute Subnetztyp und Knotentyp.
 - Der CAx-Import einer AML-Datei mit Stringwerten ohne Unterscheidung von Groß-/Kleinschreibung für die Attribute (Beispiel: Profinet, PROFINET, pROFINET, ProFINET usw.) wird unterstützt.
 - Der CAx-Export ist immer Ethernet.
-

Attribute eines „Subnetz“-Elements

Die folgende Tabelle zeigt die zugehörigen Attribute der Objekte von rechnergestützten Import- und Exportdateien:

Attribut	Handha-bung	Kommentar
Name	Obligato-risch	
Type	Obligato-risch	Ethernet oder PROFIBUS oder MPI oder ASi

Attribute von Elementen "CommunicationInterface"

Die folgende Tabelle zeigt die zugehörigen Attribute der Objekte für CAx-Import- und -Exportdateien:

Attribut	Verarbei-tung	Kommentar
Name	Obligato-risch	Keine Relevanz bei "festen" Geräteelementen.
Label	Obligato-risch	Beschriftung fehlt möglicherweise, wenn "BuiltIn" = TRUE und "PositionNumber" für das zugehörige Objekt "DeviceItem" angegeben sind.
TypIdentifier	Obligato-risch	Für integrierte Geräteelemente muss dieses Attribut mit der Typkennung seines steckbaren übergeordneten Elements exportiert werden. Während des Imports ist dieses Attribut nicht relevant. Bei nicht integrierten Geräteelementen hat dieses Attribut keine Relevanz.
FirmwareVersion	Obligato-risch	
TypeName	Nur Export	Keine Relevanz bei "integrierten" Geräteelementen.

8.6 Hardware-Daten importieren/exportieren

Attribut	Verarbei-tung	Kommentar
DeviceItemType	Nur Export	Nur für CPU und Kopfmodul Dieses Attribut ist beim Import optional, doch jedes Geräteelement außer Basiseinheiten mit DeviceItemType als Zubehör wird stillschweigend ignoriert.
PositionNumber	Obligato-risch	Keine Relevanz beim Import von "integrierten" Geräteelementen.
BuiltIn	Obligato- risch für Ex- port Optional für Import	Keine Relevanz beim Import von "nicht integrierten" Geräteelementen. Für den Import standardmäßig "False".
Comment	Optional	Nicht zutreffend bei "integrierten" Geräteelementen.

Attribute von Elementen "CommunicationPort"

Die folgende Tabelle zeigt die zugehörigen Attribute der Objekte von rechnergestützten Import- und Exportdateien:

Attribut	Handha-bung	Kommentar
Name	Obligato- risch	Keine Relevanz bei "integrierten" Geräteelementen.
Label	Obligato- risch	Das Label kann fehlen, wenn „BuiltIn“ = TRUE“ und „PositionNumber“ für das zugehörige „DeviceItem“-Objekt festgelegt sind. Beim Importieren des Werts für ein Port-Label wird beim Vergleich des Label-Werts nicht nach Groß- und Kleinschreibung unterschieden (z. B.: p1 r, P1 R, P1, p1 usw.). Bei Ports, die für eine Ringtopologie freigegeben und mit Suffix "R" gekennzeichnet sind, ist der Import "gleichwertig" mit Ports zu behandeln, deren Label-Werte kein Suffix enthalten. Beispiel: Die folgenden Port-Label werden als gleich behandelt: <ul style="list-style-type: none">• P1 ist gleich P1• P1 ist gleich P1 R• P1 R ist gleich P1• P1 R ist gleich P1 R
TypeIdentifier	Obligato- risch	Für integrierte Geräteelemente muss dieses Attribut mit der Typkennung seines steckbaren übergeordneten Elements exportiert werden. Während des Imports ist dieses Attribut nicht relevant. Für nicht integrierte Geräteelemente ist dieses Attribut nicht relevant.
FirmwareVersion	Obligato- risch	
TypeName	Nur Export	Keine Relevanz bei "integrierten" Geräteelementen.
DeviceItemType	Nur Export	Nur für CPU und Kopfmodul. Optional beim Import, doch jedes Geräteelement außer Basiseinheiten mit DeviceItemType als Zubehör wird stillschweigend ignoriert.

Attribut	Handha-bung	Kommentar
PositionNumber	Obligato- risch	Nur relevant beim Import von "integrierten" Geräteelementen, wenn das Attribut "Label" nicht angegeben ist. Wenn "PositionNumber" und "Label" beide konfiguriert sind, erhält "PositionNumber" die höhere Priorität.
BuiltIn	Obligato- risch für Ex- port Optional für Import	Keine Relevanz beim Import von "nicht integrierten" Geräteelementen. Für den Import standardmäßig "False".
Comment	Optional	Nicht zutreffend bei "integrierten" Geräteelementen.

Attribute eines „Knoten“-Elements

Die folgende Tabelle zeigt die zugehörigen Attribute der Objekte von rechnergestützten Import- und Exportdateien:

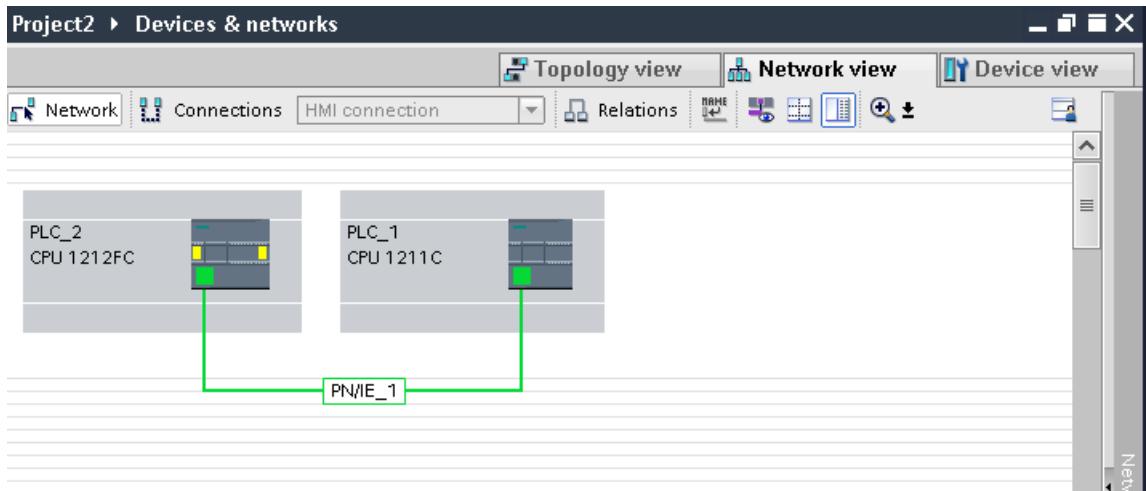
Attribut	Handhabung	Kommentar
Name	Nur Export	MPI, PROFIBUS, PROFINET
Type	Nur Exportieren	Ethernet oder PROFIBUS oder MPI oder ASI
NetworkAddress	Obligatorisch	
SubnetMask	Optional	PROFINET Beim Import wird der Standardwert beibehalten, wenn kein Wert festgelegt ist.
RouterAddress	Optional	PROFINET Beim Import wird der Standardwert beibehalten, wenn kein Wert festgelegt ist.
DhcpClientId	Optional	PROFINET Beim Import wird der Standardwert beibehalten, wenn kein Wert festgelegt ist.
IpProtocolSelection	Optional	PROFINET Beim Import wird der Standardwert beibehalten, wenn kein Wert festgelegt ist. Werte: Project, Dhcpc, UserProgram, OtherPath

Attribute eines „Kanal“-Elements

Die folgende Tabelle zeigt die zugehörigen Attribute der Objekte von rechnergestützten Import- und Exportdateien:

Attribut	Handha-bung	Kommentar
Type	Obligato- risch	Digital oder analog
IoType	Obligato- risch	Eingang oder Ausgang
Number	Obligato- risch	
Length	Nur Export- ieren	

Beispiel: Exportiertes Subnetz



AML-Struktur

Die folgenden Abbildungen zeigen die Struktur der exportierten AML-Datei.

```
...
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
  <InternalElement ID="e9d2bedb-f8c1-4148-acda-c3c68836c7dd" Name="Project2">
    ...
      <InternalElement ID="1062a384-d3ca-4183-9ac2-0934a5ab7286" Name="PN/IE_1">
        <Attribute Name="Type" AttributeDataType="xs:string">
          <Value>Ethernet</Value>
        </Attribute>
        <ExternalInterface ID="55ecf2cb-e72a-4974-8ed0-1d4e1ade3509">
          Name="LogicalEndPoint_Subnet"
          RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
        <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Subnet" />
      </InternalElement>
      <InternalElement ID="b011dbb1-efa4-46c0-a26f-f9bd047cda4f" Name="S7-1200 station_1">
        ...
        <InternalElement ID="d006e41b-05ff-44ab-baab-fca15f99e86c" Name="PROFINET interface_1">
          ...
            <InternalElement ID="beb4eb8e-1a45-45ce-a703-1acfac73e5f3" Name="E1">
              ...
                <ExternalInterface ID="a365b498-20cc-4e0b-99ca-5c5257632b96">
                  Name="LogicalEndPoint_Node" RefBaseClassPath=
                  "CommunicationInterfaceClassLib/LogicalEndPoint" />
                <SupportedRoleClass RefRoleClassPath=
                  "AutomationProjectConfigurationRoleClassLib/Node" />
              </InternalElement>
              <InternalElement ID="d45aa36a-a7f2-4862-a266-d6727b9cfcd75" Name="Port_1">
                ...
                  <ExternalInterface ID="32c6ba4a-b01f-4678-b721-ea284779e96c">
                    Name="CommunicationPortInterface"
                    RefBaseClassPath=
                    "AutomationProjectConfigurationInterfaceClassLib/CommunicationPortInterface" />
                  <SupportedRoleClass RefRoleClassPath=
                    "AutomationProjectConfigurationRoleClassLib/CommunicationPort" />
                </InternalElement>
                <SupportedRoleClass RefRoleClassPath=
                  "AutomationProjectConfigurationRoleClassLib/CommunicationInterface" />
              </InternalElement>
              <SupportedRoleClass RefRoleClassPath=
                "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
            </InternalElement>
            <SupportedRoleClass RefRoleClassPath=
              "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
          </InternalElement>
          <SupportedRoleClass RefRoleClassPath=
            "AutomationProjectConfigurationRoleClassLib/Device" />
        </InternalElement>
      ...

```

8.6 Hardware-Daten importieren/exportieren

```
...
<InternalElement ID="7cf0ea2b-b66f-4ad4-8a03-5a8691cbe04d" Name="PLC_2">
...
<InternalElement ID="b287020d-667b-483d-a8e0-c5466ac2f5c3" Name="PROFINET interface_1">
  <Attribute Name="Label" AttributeDataType="xs:string">
    <Value>X1</Value>
  </Attribute>
  <InternalElement ID="a3e85aed-580a-45c8-943e-da7de8280b7c" Name="E1">
    <Attribute Name="Type" AttributeDataType="xs:string">
      <Value>Ethernet</Value>
    </Attribute>
    <Attribute Name="NetworkAddress" AttributeDataType="xs:string">
      <Value>192.168.0.2</Value>
    </Attribute>
    <Attribute Name="SubnetMask" AttributeDataType="xs:string">
      <Value>255.255.255.0</Value>
    </Attribute>
    <Attribute Name="DeviceNumber" AttributeDataType="xs:string">
      <Value>0</Value>
    </Attribute>
    <Attribute Name="IpProtocolSelection" AttributeDataType="xs:string">
      <Value>Project</Value>
    </Attribute>
    <ExternalInterface ID="6ae8eb93-09d3-4f8c-b529-a12148c71bf4"
      Name="LogicalEndPoint_Node"
      RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
    <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
  </InternalElement>
<InternalElement ID="1e3e4c5b-04c1-4d2c-9aee-cad53cc92dba" Name="Port_1">
...
  <ExternalInterface ID="1f5b2a3d-fcd1-460a-b846-30dadc8726d1"
    Name="CommunicationPortInterface"
    RefBaseClassPath=
      "AutomationProjectConfigurationInterfaceClassLib/CommunicationPortInterface" />
  <SupportedRoleClass RefRoleClassPath=
    "AutomationProjectConfigurationRoleClassLib/CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath=
  "AutomationProjectConfigurationRoleClassLib/CommunicationInterface" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath=
  "AutomationProjectConfigurationRoleClassLib/DeviceItem" />
</InternalElement>
...

```

Siehe auch

Struktur der CAx-Daten zum Import/Export (Seite 923)

Verbindung zum TIA Portal aufbauen (Seite 79)

Projekt öffnen (Seite 111)

8.6.27 Export/Import von IO-Systemen

Voraussetzung

- Die TIA Portal Openness-Anwendung ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Die PLC ist offline.

AML-Struktur

IO-Systeme werden in der AML-Struktur als <InternalElement> dargestellt.

IO-Systeme eines Masters oder IO-Controllers werden unter dem Element <CommunicationInterface> des Geräteelements einer Schnittstelle hinzugefügt.

```
...
<InternalElement ID="[Communication Interface UniqueID]"
    Name="[Communication Interface Name]">
...
<!--Node-->
<InternalElement ID="[Node UniqueID]" Name="[Node Name]">
...
<ExternalInterface ID="[External Interface UniqueID]"
    Name="LogicalEndPoint_Node"
    RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
<SupportedRoleClass
    RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
</InternalElement>
<!--IoSystem-->
<InternalElement ID="[IoSystem UniqueID]" Name="[IoSystem Name]">
<Attribute Name="Number" Attribute DataType="xs:integer">
    <Value>[IoSystem Number]</Value>
</Attribute>
<ExternalInterface ID="[External Interface UniqueID]"
    Name="LogicalEndPoint_Interface"
    RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
<SupportedRoleClass
    RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/IoSystem" />
</InternalElement>
<SupportedRoleClass
    RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/CommunicationInterface" />
</InternalElement>
```

Als Slave oder IO-Device verbundene IO-Systeme werden als Elemente <ExternalInterface> unter dem Element <CommunicationInterface> des Geräteelements einer Schnittstelle hinzugefügt.

8.6 Hardware-Daten importieren/exportieren

```
<InternalElement ID="[Communication Interface UniqueID]"  
    Name="[Communication Interface Name]">  
...  
<ExternalInterface ID="[External Interface UniqueID]"  
    Name="LogicalEndPoint_Interface"  
    RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />  
<!--Node-->  
<InternalElement ID="[Node UniqueID]" Name="[Node Name]">  
    <ExternalInterface ID="[External Interface UniqueID]"  
        Name="LogicalEndPoint_Node"  
        RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />  
    <SupportedRoleClass  
        RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />  
</InternalElement>  
<SupportedRoleClass  
    RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/CommunicationInterface" />  
</InternalElement>
```

Die verbundenen Partner des IO-Systems werden als Elemente `<InternalLink>` dargestellt. Die Variablen `<InternalLink>` werden unter dem gemeinsamen übergeordneten Element eines IO-Systems und des verbundenen Slave-Geräteelements hinzugefügt, z. B. unter Project, DeviceFolder, DeviceItem.

Der Variablenname `<InternalLink>` ist im gemeinsamen übergeordneten Element eindeutig.

Attribute eines Elements "IO-System"

Die folgende Tabelle zeigt die zugehörigen Attribute der Objekte von rechnergestützten Import- und Exportdateien:

Attribut	Handha-bung	Kommentar
Name	Obligato-risch	Der Name des IO-Systems. Wird ein leerer String importiert, wird das IO-System mit dem Standardnamen erstellt.
Number	Optional	Erfolgt keine Angabe für den Import, wird der Standardwert verwendet.

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

8.6.28 Exportieren/Importieren mehrsprachiger Kommentare

Voraussetzung

- Die Anwendung TIA Portal Openess ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Die PLC ist offline.

Verwendung

Export- und Importkommentare und mehrsprachige Kommentare zum CAx-Datenaustausch der folgenden Hardwareobjekte:

- Geräte (Gerät)
- Module (Geräteelemente)
- Variablen (Variable)

Das Importieren/Exportieren mehrsprachiger Kommentare umfasst alle Sprachen des TIA Portals.

Einschränkungen

- Export
 - Nur, wenn ein Kommentar existiert, wird ein Attribut „Comment“ in die AML-Datei exportiert.
- Import
 - Das Attribut "Comment" ist optional.
 - Für virtuelle Geräteelemente kann kein Kommentar importiert werden.

Beispiel: Exportierte Konfigurierung mit mehrsprachigen Kommentaren

Das folgende Bild zeigt die Konfiguration eines SIMATIC S7 1500 (Gerät) mit PLC_1 (Geräteelemente). Bei beiden Objekten werden Kommentare in Englisch, Französisch, Deutsch und Chinesisch festgelegt.

S7-1200 station_1 [S7-1200 Station]					Properties	Info	Diagnostics
General	IO tags	System constants	Texts				
English (United States)	French (France)	German (Germany)	Chinese (People's Republic of Chi...)	Reference			
Profinet_Module	Profinet_Module_fr	Profinet_Module_de	Profinet_Module_cs	PROFINET interf...			
Device_01	machine_01	Gerät_01	Device_01_chs	S7-1200 statio...			
				Project2PLC_1...			
				Project2PLC_1...			

AML-Struktur

Nach dem Exportieren dieser Konfigurierungen, werden die mehrsprachigen Kommentare als verschachtelte Attribute des Geräts, Geräteelements oder der Variablen generiert.

- Das übergeordnete Attribut "Comment" muss den in der Standardsprache verwendeten Wert besitzen.
- Das Nachfolgerattribut existiert für jeden fremdsprachigen Kommentar.

```
...
<Attribute Name="Comment" AttributeDataType="xs:string">
  <Value>English</Value>
  <Attribute Name="aml-lang=en-US" AttributeDataType="xs:string">
    <Value>English</Value>
  </Attribute>
  <Attribute Name="aml-lang=de-DE" AttributeDataType="xs:string">
    <Value>Deutsch</Value>
  </Attribute>
  <Attribute Name="aml-lang=zh-HK" AttributeDataType="xs:string">
    <Value>Chinese</Value>
  </Attribute>
  ...
  ...
</Attribute>
...
```

Siehe auch

[Struktur der CAx-Daten zum Import/Export \(Seite 923\)](#)

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

8.6.29 Unterstützung von Attributen bei AML AR APC 1.1

8.6.29.1 Exportieren/Importieren von PLC-Variablen

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Die PLC ist offline.

Verwendung

Exportierte und importierte Symbole und Variablen werden einem Geräteelement zugeordnet. Rechnergestütztes Importieren/Exportieren betrifft hardwarenahe Symbole und Variablen. Die Symbole und Variablen werden nur mit dem Geräteelement des Steuersollwerts exportiert, z. B. der CPU und nicht mit anderen Geräteelementen, auf die sie sich beziehen, z. B. einem E/A-Modul. Wie Geräte, werden die Variablen oft in Variablenklassen und in einer hierarchischen Ordnerstruktur unterteilt.

AML-Strukturelemente

PLC-Variablen, Variablenklassen und Variablen-Nutzerordner können durch die rechnergestützte Import-/Exportfunktion exportiert und importiert werden. Die Variablen-Objekte werden in den folgenden AML-Strukturelementen abgebildet:

- <InternalElement>
Variablenklassen und Variablen-Nutzerordner werden als interne Elemente der zugehörigen PLC mit der jeweiligen Rollenklassifizierung abgebildet.
- <ExternalInterface>
Stellt eine PLC-Variable dar, die zu dem internen Element der zugehörigen Variablenklasse oder dem Variablen-Nutzerordner gehört.

Ein Zuordnungskanal mit einer PLC-Variable wird über das Element <internal link> als Kommunikationspartner exportiert. Die folgende XML-Struktur zeigt ein Beispiel:

```
...
<InternalLink Name="Link To Tag_1"
  RefPartnerSideA="b33451f6-d88f-4900-8dbe-41f1be1e3535:Channel_DI_0"
  RefPartnerSideB="b2b937ee-d5db-4826-9340-027b1da22828:Tag_1" />
...
...
```

PLC-Variablen-Nutzerordner

Die Objekte „TagUserFolder“ benötigen nur das Attribut „Name“ in rechnergestützten Import- und Exportdateien.

Attribute einer PLC-Variablenklasse

Die folgende Tabelle zeigt die zugehörigen Attribute der Objekte rechnergestützter Import- und Exportdateien:

Attribut	Handhabung	Kommentar
Name	Obligatorisch, wird ignoriert, wenn „AssignToDefault“ = TRUE:	
AssignToDefault	Nur Importieren	Wird verwendet, um die Standardvariablenklasse während des Imports zu identifizieren. Wenn "AssignToDefault" = TRUE, werden alle Variablen im Rahmen der Standardvariablenklasse des TIA Portals erstellt.

8.6 Hardware-Daten importieren/exportieren

Attribute einer PLC-Variablen

Die folgende Tabelle zeigt die zugehörigen Attribute der Objekte von rechnergestützten Import- und Exportdateien:

Attribut	Handhabung	Kommentar
Name	Obligatorisch	
DataType	Obligatorisch	
LogicalAddress	Optional	Wird im internationalen Mnemonik-Format importiert und exportiert
IoType	Optional	Eingang oder Ausgang
Comment	Optional	

Hinweis

TIA Portal V16 unterstützt den Export und Import des Attributs IoType in AML-Dateien mit AR APC V1.1.0.

Beispiel: AML-Struktur

Die folgende Abbildung zeigt die Struktur der folgenden exportierten Variablenobjekte:

- leere Standardvariabentabelle
- Variablen-Nutzerordner „Gruppe_1“
- enthaltene Variabentabelle „Variabentabelle_1“
- vier Variablen

```

<InternalElement ID="a310b193-ba04-49d7-a8e3-004619c7d9d2" Name="Default tag table">
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/TagTable" />
</InternalElement>
<InternalElement ID="0feff703-9c70-4ca9-b3b3-8de8229696dd" Name="Group_1">
  <InternalElement ID="f92g9ce4-c015-459f-9f59-8f94bca3b186" Name="Tag table_1">
    <ExternalInterface ID="fc0c8c5a-fd5b-443b-b430-6435b6aa22ff" Name="Tag_1" RefBaseClassPath="AutomationProjectConfigurationRoleClassLib/TagUserFolder" />
    ...
    <ExternalInterface>
      <ExternalInterface ID="450d6a1d-81b8-49ae-a104-c0072933d669" Name="Tag_2" RefBaseClassPath="AutomationProjectConfigurationRoleClassLib/TagUserFolder" />
      ...
      <ExternalInterface>
        <ExternalInterface ID="3de17a36-b5c5-4fc7-9fc3-47e4a8f95087" Name="Tag_3" RefBaseClassPath="AutomationProjectConfigurationRoleClassLib/TagUserFolder" />
          <Attribute Name="DataType" Attribute DataType="xs:string">
            <Value>Word</Value>
          </Attribute>
          <Attribute Name="IoType" Attribute DataType="xs:string">
            <Value>Input</Value>
          </Attribute>
          <Attribute Name="LogicalAddress" Attribute DataType="xs:string">
            <Value>IWO</Value>
          </Attribute>
        </ExternalInterface>
        <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/TagTable" />
      </InternalElement>
      <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/TagUserFolder" />
    </InternalElement>
    <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/TagUserFolder" />
  </InternalElement>
...
  
```

Siehe auch

[Struktur der CAx-Daten zum Import/Export \(Seite 923\)](#)

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

8.6.29.2 Export/Import von RH/PLC

Voraussetzung

- TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Der PLC ist offline.

Anwendung

Sie können mit dem TIA Portal AML-Dateien nach AR APC V1.1 mit R/H PLC in gleicher IP-Adresskonfiguration importieren.

Attribut

TIA Portal unterstützt nur ein Attribut in der AML-Datei für R/H PLC, falls in der Benutzeroberfläche des TIA Portals verfügbar:

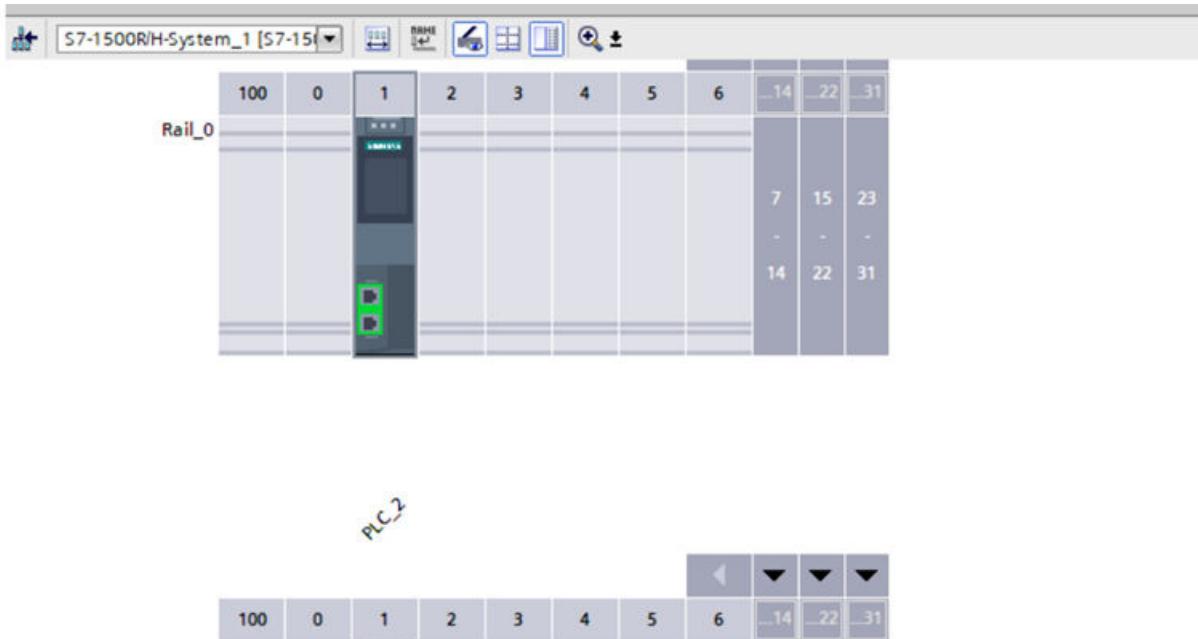
Attributname des Geräteelements	Handhabung	Kommentar
HNetworkAddress	Obligatorisch	Exportiert/Importiert, wenn der R/H PLC des Geräteelements dieses Attribut in TIA Portal unterstützt wird und nicht leer ist, andernfalls wird er übersprungen. Für den Import wird "System IP-Adresse für geschaltete Verbindung freigeben" eingestellt und die HNetworkAddress bewertet.

Einschränkung

- Exportieren
 - Nur bei Auswahl von "System IP-Adresse für geschaltete Verbindung freigeben" in TIA Portal.

Beispiel: Exportierte Konfiguration

Die folgende Konfiguration zeigt ein Geräteelement, bei dem die HNetworkAddress konfiguriert ist.



AML-Struktur der Exportdatei

Die folgende Abbildung zeigt die Struktur der exportierten AML-Datei für R/H PLC.

```

<InternalElement ID="e1879cf8-8222-4ce3-b171-60c56aed7f18" Name="PROFINET interface_1">
  <Attribute Name="Label" AttributeDataType="xs:string">
    <Value>X1</Value>
  </Attribute>
  <Attribute Name="PositionNumber" AttributeDataType="xs:int">
    <Value>32768</Value>
  </Attribute>
  <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
    <Value>true</Value>
  </Attribute>
<InternalElement ID="a32c67c7-47dc-4362-a9c4-b41b93b58eff" Name="E1">
  <Attribute Name="Type" AttributeDataType="xs:string">
    <Value>Ethernet</Value>
  </Attribute>
  <Attribute Name="NetworkAddress" AttributeDataType="xs:string">
    <Value>192.168.0.1</Value>
  </Attribute>
  <Attribute Name="SubnetMask" AttributeDataType="xs:string">
    <Value>255.255.255.0</Value>
  </Attribute>
  <Attribute Name="IpProtocolSelection" AttributeDataType="xs:string">
    <Value>Project</Value>
  </Attribute>
  <Attribute Name="HNetworkAddress" AttributeDataType="xs:string">
    <Value>192.168.0.3</Value>
  </Attribute>
<ExternalInterface ID="802676fa-212f-4bf5-b112-de94993a0340" Name="LogicalEndPoint_Node">
  RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint" />
  <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
</InternalElement>

```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

8.6.29.3 Export/Import von AML mit benutzerdefinierten Variablen und Geräteelementen

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Der PLC ist offline.

Anwendung

Sie können mit dem TIA Portal AML-Dateien nach AR APC V1.1 mit einem Subattribut des Typs 'Customized' für Variable und DeviceItem exportieren und importieren.

Attribut

Die folgende Tabelle zeigt das zugehörige Attribut, das für eine Variable beim CAx-Import und -Export von Dateien verfügbar ist:

Attributname	Handhabung	Kommentar
Customized	Optional beim Export/ Import	Subattribut von 'DataType' einer Variablen. Für 'Customized' sind nur die Werte TRUE und FALSE erlaubt. Während des Imports ist das Attribut 'Customized' nicht relevant. Während des Exports hat das Subattribut 'Customized' bei allen Datentypen, die nicht IEC 61131 entsprechen, den Wert "True". Bei Datentypen nach IEC 61131 wird das Subattribut 'Customized' nicht exportiert.

Die folgende Tabelle zeigt das zugehörige Attribut, das für ein Geräteelement beim CAx-Import und -Export von Dateien verfügbar ist:

Attributname	Handhabung	Kommentar
Customized	Optional beim Import	Es ist ein Nachfolgeattribut des übergeordneten Attributs 'DeviceItemType' für ein Geräteelement (DeviceItem). Bei Customized sind nur die Werte TRUE und FALSE erlaubt. Während des Imports und Exports ist das Attribut Customized nicht relevant.

Exportierte AML-Datei

Die folgende AML-Datei wird während des Exports einer AML-Datei mit Geräteelementen mit Attribut 'Customized' generiert.

```
<?xml version="1.0" encoding="utf-8"?>
<CAEXFile FileName="Project26.aml" SchemaVersion="2.15"
xsi:noNamespaceSchemaLocation="CAEX_ClassModel_V2.15.xsd">
<AdditionalInformation>
<WriterHeader>
...
</WriterHeader>
</AdditionalInformation>
<AdditionalInformation AutomationMLVersion="2.0" />
<AdditionalInformation DocumentVersions="Recommendations">
<Document DocumentIdentifier="AR APC" Version="1.1.0" />
</AdditionalInformation>
<InstanceHierarchy Name="APC Sample Instance Hierarchy">
<InternalElement ID="03ecf798-3e07-4976-b281-f8b98eb3a590" Name="Project26">
...
<InternalElement ID="c218aea9-93b8-4719-be6f-ccfa9517e2c6" Name="S71500/ET200MP station_1">
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>System:Device.S71500</Value>
</Attribute>
<InternalElement ID="c1ae306f-183f-47b8-91e6-cb331c559278" Name="Rail_0">
...
<InternalElement ID="8d7d5ee1-603a-4897-b47e-8954d4c21a31" Name="PLC_1">
...
<Attribute Name="DeviceItemType" AttributeDataType="xs:string">
<Value>CPU</Value>
</Attribute>
...
...
<SupportedRoleClass RefRoleClassName="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassName="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<SupportedRoleClass RefRoleClassName="AutomationProjectConfigurationRoleClassLib/
Device" />
</InternalElement>
<SupportedRoleClass RefRoleClassName="AutomationProjectConfigurationRoleClassLib/
AutomationProject" />
</InternalElement>
</InstanceHierarchy>
</CAEXFile>
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

8.6.29.4 Import/Export fehlersicherer PLCs**Voraussetzung**

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Der PLC ist offline.

Anwendung

Sie können mit dem TIA Portal eine AML-Datei nach AR APC V1.1 mit fehlersicherer PLC mit fehlersicheren Attributen exportieren und importieren.

Attribute

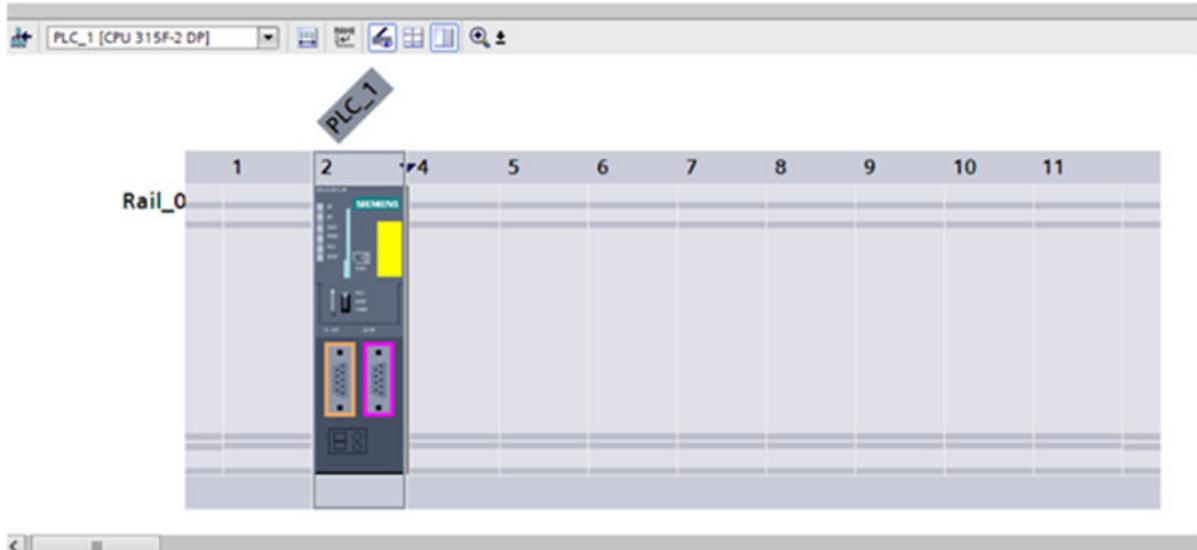
Die folgende Tabelle zeigt die Liste der fehlersicheren Attribute von AML-Dateien für den CAx-Import und CAx-Export:

Attribute in Openness	Handhabung	Kommentar	AR APC-Name in AML
Failsafe_FSourceAddress	Obligatorisch	Export/Import nur, wenn das Geräteelement ein fehlersicherer PLC ist und dieses Attribut in TIA Portal unterstützt und nicht leer ist, andernfalls wird er übersprungen.	Failsafe_FSourceAddress
Failsafe_LowerBoundForFDestinationAddresses	Obligatorisch	Export/Import nur, wenn das Geräteelement ein fehlersicherer PLC ist und dieses Attribut in TIA Portal unterstützt und nicht leer ist, andernfalls wird er übersprungen.	Failsafe_LowerBoundForFDestinationAddresses
Failsafe_UpperBoundForFDestinationAddresses	Obligatorisch	Export/Import nur, wenn das Geräteelement ein fehlersicherer PLC ist und dieses Attribut in TIA Portal unterstützt und nicht leer ist, andernfalls wird er übersprungen.	Failsafe_UpperBoundForFDestinationAddresses

Attribute in Openness	Handhabung	Kommentar	AR APC-Name in AML
Failsafe_CentralFSourceAddress	Optional	Export/Import nur, wenn das Geräteelement ein fehlersicherer PLC ist und dieses Attribut in TIA Portal unterstützt und nicht leer ist, andernfalls wird er übersprungen.	Failsafe_FSourceAddress
Failsafe_FDestinationAddress	Optional	Export/Import nur, wenn das Geräteelement ein fehlersicherer PLC ist und dieses Attribut in TIA Portal unterstützt und nicht leer ist, andernfalls wird er übersprungen.	Failsafe_FDestinationAddress

Beispiel: Exportierte Konfiguration

Die folgende Konfiguration zeigt ein Geräteelement, bei dem die Attribute konfiguriert sind.



AML-Struktur der Exportdatei

Die folgenden Abbildungen zeigen die Struktur der exportierten AML-Datei.

```

<InternalElement ID="9c944d2c-e0ae-4f39-b35a-a63faaf35be7" Name="PLC_1">
  <Attribute Name="TypeName" AttributeDataType="xs:string">
    <Value>CPU 1511TF-1 PN</Value>
  </Attribute>
  <Attribute Name="DeviceItemType" AttributeDataType="xs:string">
    <Value>CPU</Value>
  </Attribute>
  <Attribute Name="PositionNumber" AttributeDataType="xs:int">
    <Value>1</Value>
  </Attribute>
  <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
    <Value>false</Value>
  </Attribute>
  <Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
    <Value>OrderNumber:6ES7 511-1UK01-0AB0</Value>
  </Attribute>
  <Attribute Name="InstallationDate" AttributeDataType="xs:dateTime">
    <Value>2019-02-28T08:12:12.987Z</Value>
  </Attribute>
  <Attribute Name="Failsafe_FSourceAddress" AttributeDataType="xs:string">
    <Value>1</Value>
  </Attribute>
  <Attribute Name="Failsafe_LowerBoundForFDestinationAddresses" AttributeDataType="xs:string">
    <Value>1</Value>
  </Attribute>
  <Attribute Name="Failsafe_UpperBoundForFDestinationAddresses" AttributeDataType="xs:string">
    <Value>99</Value>
  </Attribute>
  <Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
    <Value>V2.8</Value>
  </Attribute>
  <InternalElement ID="4f718e93-b541-4983-8f13-1f5b21c3e70c" Name="Default tag table">
    <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/TagTable"/>
  </InternalElement>
  <InternalElement ID="20e19f5b-8ace-4e0c-af0b-c710ae4817da" Name="CPU display_1">
    <Attribute Name="PositionNumber" AttributeDataType="xs:int">
      <Value>3</Value>
    </Attribute>
    <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
      <Value>true</Value>
    </Attribute>
    <SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/DeviceItem" />
  </InternalElement>
  <InternalElement ID="5a24516f-17d6-4b2a-a4ac-efc1b577875d" Name="Card reader/writer_1">
    <Attribute Name="PositionNumber" AttributeDataType="xs:int">
      <Value>4</Value>
    </Attribute>
    <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
      <Value>true</Value>
    </Attribute>

```

8.6 Hardware-Daten importieren/exportieren

```
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement> <InternalElement ID="0f746d71-035e-4e64-b0d7-51d0449cf88" Name="OPC
UA_1">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>254</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value> </Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<InternalElement ID="a0633104-a2ac-4680-bb99-81df50f5ec40" Name="PROFINET interface_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>X1</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>32768</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<InternalElement ID="e3497176-dbba-4fec-9d3a-772ae13987c4" Name="E1">
<Attribute Name="Type" AttributeDataType="xs:string">
<Value>Ethernet</Value> </Attribute>
<Attribute Name="NetworkAddress" AttributeDataType="xs:string">
<Value>192.168.0.1</Value>
</Attribute>
<Attribute Name="SubnetMask" AttributeDataType="xs:string">
<Value>255.255.255.0</Value>
</Attribute>
<Attribute Name="IpProtocolSelection" AttributeDataType="xs:string">
<Value>Project</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/Node" />
</InternalElement>
<InternalElement ID="3208384f-d5ba-4ccb-b8da-f08ec38ec681" Name="Port_1">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P1 R</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>32769</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<InternalElement ID="4a47c05e-9656-4e02-9b51-23b065b6fe47" Name="Port_2">
<Attribute Name="Label" AttributeDataType="xs:string">
<Value>P2 R</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>32770</Value>
</Attribute>
```

```
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationPort" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
CommunicationInterface" />
</InternalElement>
<SupportedRoleClass RefRoleClassPath="AutomationProjectConfigurationRoleClassLib/
DeviceItem" />
</InternalElement>
<InternalElement ID="e3fdb611-4b68-4682-b154-ae43c74a24d3" Name="F-DI 16x24V DC_1">
<Attribute Name="TypeName" AttributeDataType="xs:string">
<Value>F-DI 16x24V DC</Value>
</Attribute>
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>2</Value>
</Attribute>
<Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>false</Value> </Attribute>
<Attribute Name="TypeIdentifier" AttributeDataType="xs:string">
<Value>OrderNumber:6ES7 526-1BH00-0AB0</Value>
</Attribute>
<Attribute Name="FirmwareVersion" AttributeDataType="xs:string">
<Value>V1.0</Value>
</Attribute>
<InternalElement ID="77c4fea0-baba-44e6-80f2-72b7b830a88a" Name="F-DI 16x24V DC_1">
<Attribute Name="PositionNumber" AttributeDataType="xs:int">
<Value>1</Value>
</Attribute> <Attribute Name="BuiltIn" AttributeDataType="xs:boolean">
<Value>true</Value>
</Attribute>
<Attribute Name="Failsafe_FSourceAddress" AttributeDataType="xs:string">
<Value>1</Value>
</Attribute>
<Attribute Name="Failsafe_FDestinationAddress" AttributeDataType="xs:string">
<Value>655</Value>
</Attribute>
</InternalElement>
</InternalElement>
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

8.6.29.5 Import/Export fehlersicherer E/A

Voraussetzung

- Die Anwendung TIA Portal Openness ist mit dem TIA Portal verbunden.
Siehe Verbindung zum TIA Portal herstellen (Seite 79)
- Ein Projekt ist geöffnet.
Siehe Öffnen eines Projekts (Seite 111)
- Der PLC ist offline.

Anwendung

Sie können mit dem TIA Portal eine AML-Datei nach AR APC V1.1 mit fehlersicheren E/A mit fehlersicheren Attributen exportieren und importieren.

Attribute

Die folgende Tabelle zeigt die zugehörigen Attribute, die an fehlersicheren E/A-Modulen-/Submodulen für CAx-Importdateien und CAx-Exportdateien verfügbar sind:

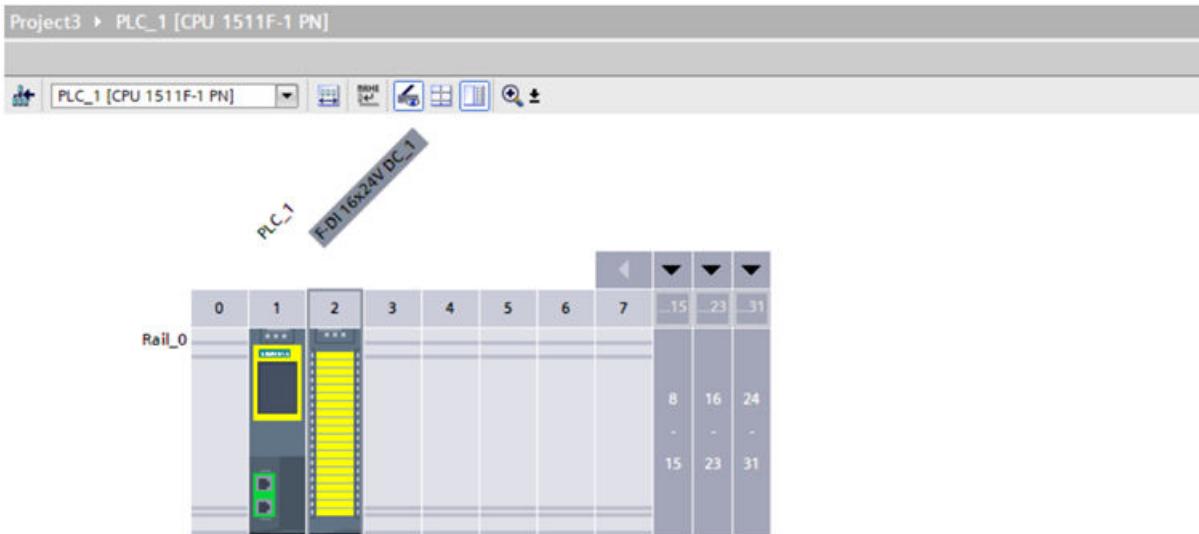
Attributname des Geräteelements	Handhabung	Kommentar
FSourceAddress	Obligatorisch	Export/Import nur, wenn das Geräteelement fehlersichere E/A ist und in TIA Portal unterstützt wird.
FDestinationAddresses	Obligatorisch	Export/Import nur, wenn das Geräteelement fehlersichere E/A ist und in TIA Portal unterstützt wird.

Hinweis

- Für die meisten fehlersicheren E/A ist FSourceAddress schreibgeschützt – also nicht beschreibbar und eine Reflexion des Attributs FCentralFSourceAddress des fehlersicheren PLC. Es wird der gleiche Wert exportiert und beim Import ignoriert.
- Wenn FSourceAddress und FDestinationAddress schreibgeschützt sind, wird der Wert exportiert. Jedoch muss während des Imports eine Warnmeldung im Info-Register des TIA Portals angezeigt werden.
- FSourceAddress und FDestinationAddress müssen auf E/A-Modul- und E/A-Submodulebene unterstützt werden.

Beispiel: Exportierte Konfiguration

Die folgende Konfiguration zeigt ein Geräteelement, bei dem die Attribute konfiguriert sind.



AML-Struktur der Exportdatei

Die folgende Abbildung zeigt die Struktur der exportierten AML-Datei.

```
<Attribute Name="FSourceAddress" AttributeDataType="xs:string">
<Value>1</Value>
</Attribute>
<Attribute Name="FDestinationAddress" AttributeDataType="xs:string">
<Value>65534</Value>
</Attribute>
```

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

8.6.29.6 Herstellerspezifisches Attribut exportieren/importieren

Voraussetzung

- Die Anwendung TIA Portal Openess ist mit dem TIA Portal verbunden.
[Siehe Verbindung zum TIA Portal herstellen \(Seite 79\)](#)
- Ein Projekt ist geöffnet.
[Siehe Projekt öffnen \(Seite 111\)](#)

8.6 Hardware-Daten importieren/exportieren

Anwendung

Im TIA Portal können Sie eine AML-Datei für Geräte und Geräteelemente mit dem Attribut "Manufacturer" exportieren und importieren, so dass Sie herstellerspezifische Informationen in Round-Trip-Szenarien austauschen können.

Das Attribut "Manufacturer" wird von Openness nicht unterstützt. Es wird beim Export/Import vom "Transformation-Plugin" aufgenommen.

Hinweis

AML-Dateien müssen von TIA Portal V16 nach AR APC V1.1 generiert/exportiert werden.

Exportieren

Sie können das Attribut "Manufacturer" in eine AML-Datei exportieren, wenn es im AmiModel verfügbar ist, das von Nutzern des "Transformation-Plugin" ausgefüllt wird.

Importieren

Sie können das Attribut "Manufacturer" ins TIA Portal importieren, in dem das herstellerspezifische Attribut im AmiModel verfügbar ist, das von Nutzern des "Transformation-Plugin" aufgenommen wird.

Derzeit wird dieses Attribut nur für Geräte und Geräteelemente vom Typ "Antrieb" unterstützt, da nur diese das offizielle, für CAx geschriebene "Transformation-Plugin" haben.

Siehe auch

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

[Projekt öffnen \(Seite 111\)](#)

8.6.30 AML-Attribute im Vergleich zu TIA Portal Openness-Attributen

Auf Attribute und Export-/Importattribute zugreifen

Über TIA Portal Openness können Sie auf Hardware-Attribute von Hardwareobjekten zugreifen. Einzelne Namen, die Sie für den Zugriff auf diese Attribute verwenden, z. B. von einem Geräteelement, unterscheiden sich von den Attributnamen in der Export-/Import-AML-Datei.

Liste der Attribute

Die folgende Tabelle bietet einen Überblick über beide Arten von Attributen:

Tabelle 8-6 Attributnamen von Geräten und GSD/GSDML-Geräten

AML-Datei	TIA Portal Openness
Name	Name
TypeIdentifier	TypeIdentifier
Comment	Comment

Tabelle 8-7 Attributnamen von Geräteelementen

AML-Datei	TIA Portal Openness
Name	Name
TypeIdentifier	Zugeordnet zum Substring von <TypeIdentifier> (d.h. Wert vor ersten Operator "/"), wobei der Bestandteil mit der Firmwareversion ignoriert wird. Die Zuordnung des Substrings ist nur zutreffend, wenn die Typkennung mit dem Präfix <OrderNumber :> beginnt und einen Bestandteil mit der Firmwareversion hat, ansonsten erfolgt die Zuordnung zur Vervollständigung von <TypeIdentifier>.
FirmwareVersion	<FirmwareVersion> ist dem Substring von <TypeIdentifier> zugeordnet (d. h. Wert nach dem ersten Operator "/"). Die Zuordnung des Substrings ist nur zutreffend, wenn <TypeIdentifier> mit dem Präfix <OrderNumber :> beginnt und einen Bestandteil mit der Firmwareversion hat.
TypeName	TypeName
DeviceItem Type (für CPU und Kopfmodul)	Classification
PositionNumber	PositionNumber
BuiltIn	IsBuiltIn
PlantDesignation IEC	PlantDesignation
LocationIdentifier IEC	LocationIdentifier
Comment	Comment

Tabelle 8-8 Attributnamen von GSD/GSDML-Geräteelementen

AML-Datei	TIA Portal Openness
Name	Name
TypeIdentifier	TypeIdentifier
TypeName	TypeName
DeviceItem Type (für Kopfmodul)	Classification
PositionNumber	PositionNumber

8.6 Hardware-Daten importieren/exportieren

AML-Datei	TIA Portal Openness
BuiltIn	IsBuiltIn
Comment	Comment
Label	Label

Tabelle 8-9 Attributnamen von Variablen

AML-Datei	TIA Portal Openness
Name	Name
DataType	DataTypeName
LogicalAddress	LogicalAddress
Comment	Comment

Tabelle 8-10 Attributnamen von Variablenklassen

AML-Datei	TIA Portal Openness
Name	Name
AssignToDefault	IsDefault

Tabelle 8-11 Attributnamen von Adressen

AML-Datei	TIA Portal Openness
StartAddress	StartAddress
Length	Length
IoType	IoType

Tabelle 8-12 Attributnamen von Ports

AML-Datei	TIA Portal Openness
Name	Name
TypeIdentifier	TypeIdentifier
FirmwareVersion	FirmwareVersion
TypeName	TypeName
PositionNumber	PositionNumber
BuiltIn	IsBuiltIn
Comment	Comment
Label	Label

Tabelle 8-13 Attributnamen von Geräten mit IO-Schnittstelle

AML-Datei	TIA Portal Openness
Name	Name
TypeIdentifier	TypeIdentifier
FirmwareVersion	FirmwareVersion
TypeName	TypeName
DeviceItemType (für CPU und Kopfmodul)	Classification
PositionNumber	PositionNumber
BuiltIn	IsBuiltIn
Label	Label
Comment	Comment

Tabelle 8-14 Attributnamen von Kanälen

AML-Datei	TIA Portal Openness
Type	Type
IoType	IoType
Number	Keinem Attribut in TIA Portal Openness zugeordnet.
Length	ChannelWidth

Wesentliche Änderungen

9.1 Größere Änderungen in TIA Portal Openness V15.1

Änderungen

Wenn Sie die Hinweise zur versionsübergreifenden Programmierung beachtet haben und Ihr Projekt nicht auf V15.1 aktualisieren, laufen Ihre Anwendungen ohne jede Einschränkung auf jedem Rechner, selbst wenn nur ein TIA Portal V15.1 installiert ist.

Wenn Sie Ihr Projekt auf V15.1 aktualisieren, ist es notwendig, Ihre Anwendung mit der SiemensEngineering.dll von V15.1 neu zu übersetzen. In manchen Fällen kann es erforderlich sein, den Code Ihrer Anwendung anzupassen.

Typkennungen

Die Typkennung für Baugruppenträger und Geräte der Art "PC mit Ports" sowie "Ethernet-Gerät mit Ports" wurde geändert.

PC mit Ports	Typkennung vor V15.1	Typkennung ab V15.1
Gerät (device)	System:DesktopPC.Device	System:Device/DesktopPC
Baugruppenträger (rack)	System:DesktopPC.Rack	System:Rack/DesktopPC
Geräteelement	System:DesktopPC.Port<X> <X> denotes the number of ports	System:DeviceItem/EthernetDevice.Port<X> <X> denotes the number of ports

Ethernet-Geräte mit Ports		
Gerät (device)	System:DummyPC.Device	System:Device/EthernetDevice
Baugruppenträger (rack)	System:Rack/DummyPC	System:Rack/EthernetDevice
Geräteelement	System:DummyPC.Port<X> <X> denotes the number of ports	

Ausfälle beim Versuch, mit dem TIA Portal zu verbinden

Führt der Versuch, eine Verbindung zum TIA Portal herzustellen, zu einem Ausfall, ist die daraufhin angezeigte Meldung nun spezifischer.

Thread-übergreifender Betrieb

Der Zugriff auf Openness-Objekte ist nicht inhärent threadsicher.

Wenn Sie Multithreading zur Verbesserung der Performance Ihrer Openness-Anwendung einsetzen, wird die Erstellung der Instanz des TIA Portals mit MTA empfohlen.

Wenn TIA Portal innerhalb eines STA-Threads erstellt oder angehängt wird, muss auf alle Openness-Objekte in Zusammenhang mit dieser Instanz des TIA Portals aus demselben STA-Thread zugegriffen werden; andernfalls wird eine Ausnahme ausgelöst.

Submodule haben nicht die Attribute Author und TypeName.

Die Attribute Author und TypeName wurden von Submodulen, die nicht gesteckt werden können, entfernt.

Öffnen einer globalen Bibliothek

Ab TIA Portal Openness V15.1 kann eine über Openness, unabhängig von einem bestehenden Vorschaumodus der Bibliothek, eine globale Bibliothek geöffnet werden.

Code bei Anwendungsabbruch

Im Fall eines Codes bei Anwendungsabbruch geschieht Folgendes:

- Bis zu TIA Portal Openness V15 erhalten Sie einen Ausnahmefehler ohne Wiederherstellbarkeit
- Ab TIA Portal Openness V15.1 erhalten Sie eine EngineeringRuntimeException oder eine EngineeringTargetException, wenn der Fehlercode bekannt ist, oder einen Ausnahmefehler ohne Wiederherstellbarkeit, wenn der Fehlercode unbekannt ist.

Schemaerweiterung für namenlose Parameter

SCL-Bausteine können importiert werden, selbst wenn ENO bei einem nicht formellen Aufruf genutzt wird.

Header indizierter Parameter

Ab TIA Portal Openness V15.1 kann der Header indizierter Parameter nicht über Openness geändert werden.

Einige Antriebsparameter werden als indizierte Parameter modelliert, da sie mehrere Datenstücke zu einem Thema bieten. Die Modellierung der indizierten Antriebsparameter in Openness folgt der antriebsspezifischen Definition der jeweiligen Parameter wie im entsprechenden Listenhandbuch festgelegt.

Indizierte Parameter werden wie folgt modelliert:

Header-Element: Der jeweilige Antriebsparameter ohne Index. Das Header-Element enthält einen beschreibenden Text, der Sie über die Semantik des referenzierten indizierten Antriebsparameters informiert. Deshalb ist das Header-Element schreibgeschützt, weil es keinen Istwert enthält.

Indizierte Elemente: Die indizierten Elemente des Antriebsparameters unter dem Header-Element. Sie liefern beschreibenden Text, der die Semantik des spezifischen indizierten

Elements definiert (schreibgeschützt). Zusätzlich geben sie auch den Wert an, der über die Openness API abgerufen werden kann. Wenn der jeweilige indizierte Parameter auch beschreibbar ist, kann der Wert auch über die Openness API festgelegt werden.

Attribut TransmissionRateAndDuplex

Einige fehlerhafte Enumerationswerte für das Attribut TransmissionRateAndDuplex wurden korrigiert, beispielsweise wurde der Enumerationswert "POFPCF100MbpsFullDuplexLD" entfernt und "POFPCF100MbpsFullDuplex" wurde hinzugefügt. Ausführliche Informationen finden Sie unter Konfigurierbare Attribute einer Port-zu-Port-Verbindung (Seite 139).

Attribut AutoNumber für Bausteine mit Know-how-Schutz

Ab V15.1 kann das Attribut AutoNumber nicht über TIA Portal Openness geändert werden, wenn ein Baustein über Know-how-Schutz verfügt.

Anzahl der von Schnittstelle ChannelInfo gelisteten Kanäle

Bis TIA Portal Openness V15 zeigt die Schnittstelle ChannelInfo die Anzahl der verfügbaren Kanälen bei manchen Modulen nicht richtig an.

Zugriff auf ProDiag-FB-Attribute

Auf die folgenden Attribute eines ProDiag-Funktionsbausteins kann über TIA Portal Openness zugegriffen werden:

- Version
- Erstwerterfassung
- Nutzung zentraler Zeitstempel

Import/Export fehlersicherer Bausteine

Der Import fehlersicherer Bausteine aus früheren Versionen ist nicht möglich.

Der Export von systemgenerierten fehlersicheren Bausteinen wird ab TIA Portal Openness V15.1 verhindert.

R/H-Systeme

Der Download bzw. Zugriff für R/H-Geräte ist auf dem Gerät verfügbar.

Online- und Download-Provider sind für einzelne R/H PLCs (DeviceItems) nicht verfügbar.

Für PLC2 eines R/H-Systems ist SoftwareContainer nicht verfügbar.

9.2 Größere Änderungen in TIA Portal Openness V15

Änderungen

Wenn Sie die Hinweise zur versionsübergreifenden Programmierung beachtet haben und Ihr Projekt nicht auf V15 aktualisieren, laufen Ihre Anwendungen ohne jede Einschränkung auf jedem Rechner, selbst wenn nur ein TIA Portal V15 installiert ist.

Wenn Sie Ihr Projekt auf V15 aktualisieren, ist es notwendig, Ihre Anwendung mit der SiemensEngineering.dll von V15 neu zu übersetzen.

In manchen Fällen ist es notwendig, den Code Ihrer Anwendung anzupassen

- Verhaltensänderungen für Zusammensetzungen in DeviceItemComposition
- BitOffset von ASi-Adressen
- Klasse Exception
- Systemordner von System-UDTs
- Submodule haben nicht die Attribute Author und TypeName
- Zeitstempel für letzte Änderung
- Export-XML für GRAPH-Bausteine
- Variablenlisten importieren
- Nicht fehlersichere relevante Attribute eines PLC ändern
- F-Parameter bei eingestelltem Sicherheitspasswort ändern
- Zugriff auf Objekte in einer S7-1200 CPU

Verhaltensänderungen für Zusammensetzungen in DeviceItemComposition

Die folgenden Zusammensetzungen in DeviceItemCompositon wurden geändert, um ein dynamisches Verhalten zu erreichen: Die Zusammensetzung wird jetzt aktualisiert, wenn ein Element über die Benutzeroberfläche des TIA Portals hinzugefügt oder gelöscht wird.

- IoSystem – ConnectedIoDevices
- Subnet – IoSystems
- Subnet – Nodes
- NetworkInterface – Nodes
- NetworkInterface – Ports
- NetworkPort – ConnectedPorts
- SubnetOwner – Subnets

BitOffset von ASi-Adressen

Wenn ein Modul eine Eingangs- und eine Ausgangsadresse für beide Adressobjekte hat, wird das korrekte Attribut BitOffset bereitgestellt.

Wenn ein Modul Kanäle hat, wird das Attribut BitOffset für den Kanal nicht bereitgestellt.

Klasse Exception

ServicelD und MessageID wurden aus der Klasse exception entfernt.

Submodule haben nicht die Attribute Author und TypeName.

Die Attribute Author und TypeName wurden von Submodulen, die nicht gesteckt werden können, entfernt.

Systemordner von System-UDTs

Bei Systemordnern von System-UDTs werden der entsprechende Ordner und die entsprechende Zusammensetzung bereitgestellt. Das führt auch zu einer Änderung in der Hierarchie von Vergleichsergebnissen.

Zeitstempel für letzte Änderung

Wenn während eines Upgrades ein Objekt geändert wird, ändert sich auch der Zeitstempel für die letzte Änderung.

Export-XML für GRAPH-Bausteine

Die Export-XML für GRAPH-Bausteine enthält eine zusätzliche leere Aktion: <Actions />.

VariablenTabellen importieren

Die Einstellung von Variablenattributen ist nicht mehr von Datentypen abhängig.

Nicht fehlersichere relevante Attribute eines PLC ändern

Alle nicht fehlersicheren relevanten Attribute eines PLC können über TIA Portal Openness geändert werden, selbst wenn ein Sicherheitspasswort eingestellt ist.

F-Parameter bei eingestelltem Sicherheitspasswort ändern

F-Parameter von F-IO können nur geändert werden, wenn das Sicherheitspasswort nicht eingestellt ist.

Zugriff auf Objekte in einer S7-1200 CPU

Der Zugriff auf Array-Variablen für die Technologieobjekte TO_PositioningAxis und TO_CommandTable wurde geändert. Einzelheiten hierzu finden Sie im Kapitel über S7-1200 Motion Control.

9.3 Die wichtigsten Änderungen in V14 SP1

9.3.1 Die wichtigsten Änderungen in V14 SP1

Einleitung

Die folgenden Änderungen wurden bei dem TIA Portal Openness API-Objektmodell V14 SP1 vorgenommen, was möglicherweise Auswirkungen auf Ihre vorhandenen Anwendungen hat:

Änderung	Erforderliche Anpassung von Programmcode
Verbesserte Handhabung von Masterkopien	<p>Die Aktion CreateFrom erstellt ein neues Objekt basierend auf einer Masterkopie in einer Bibliothek und platziert es in der Zusammensetzung, wo die Aktion aufgerufen wurde. Die Aktion CreateFrom unterstützt nur Masterkopien, die ausschließlich einzelne Objekte enthalten. Der Ausgabotyp entspricht dem jeweils zusammengesetzten Typ.</p> <p>Die folgenden Zusammensetzungen unterstützen CreateFrom:</p> <ul style="list-style-type: none">• Siemens.Engineering.HW.DeviceComposition• Siemens.Engineering.HW.DeviceItemComposition• Siemens.Engineering.SW.Blocks.PlcBlockComposition• Siemens.Engineering.SW.Tags.PlcTagTableComposition• Siemens.Engineering.SW.Tags.PlcTagComposition• Siemens.Engineering.SW.Types.PlcTypeComposition• Siemens.Engineering.SW.TecnologicalObjects.TecnologicalInstanceDBComposition• Siemens.Engineering.SW.Tags.PlcUserConstantComposition• Siemens.Engineering.Hmi.Tag.TagTableComposition• Siemens.Engineering.Hmi.Tag.TagComposition• Siemens.Engineering.Hmi.Screen.ScreenComposition• Siemens.Engineering.Hmi.Screen.ScreenTemplateComposition• Siemens.Engineering.Hmi.RuntimeScripting.VBScriptComposition• Siemens.Engineering.HW.SubnetComposition• Siemens.Engineering.HW.DeviceUserGroupComposition• Siemens.Engineering.SW.Blocks.PlcBlockUserGroupComposition• Siemens.Engineering.SW.ExternalSources.PlcExternalSourceUserGroupComposition• Siemens.Engineering.SW.Tags.PlcTagTableUserGroupComposition• Siemens.Engineering.SW.Types.PlcTypeUserGroupComposition
Verbesserte Handhabung globaler Bibliotheken	<p>Bestehende Aktionen mit globalen Bibliotheken können jetzt Änderungsaktionen umfassen, z. B. Löschen einer Masterkopie aus einer globalen Bibliothek.</p> <p>UpdateProject und UpdateLibrary verwenden nicht länger die Parameter UpdatePathsMode und DeleteUnusedVersionsMode. Nicht verwendete Versionen werden nach einem Update nicht gelöscht</p>

Änderung	Erforderliche Anpassung von Programmcode
Änderung von System.String in System.IO.FileInfo	Alle Ereignisse, bei denen ein String-Pfad spezifiziert werden muss, verwenden den Pfad FileInfo oder den Pfad DirectoryInfo. Beispiel:
Änderung von System.String in System.IO.DirectoryInfo	<ul style="list-style-type: none"> • Projekt öffnen • Bibliothek öffnen • Projekt erstellen • Globale Bibliothek erstellen • ...

Neue Elemente im Objektmodell

Name	Typ	Namensraum	Kommentar
PlcUserConstant	Klasse	Siemens.Engineering.SW.Tags	Teilen von PlcConstant
PlcUserConstantComposition	Klasse	Siemens.Engineering.SW.Tags	Teilen von PlcConstantComposition
PlcSystemConstant	Klasse	Siemens.Engineering.SW.Tags	Teilen von PlcConstant
PlcSystemConstantComposition	Klasse	Siemens.Engineering.SW.Tags	Teilen von PlcConstantComposition
MultilingualTextItem	Klasse	Siemens.Engineering	Auf mehrsprachige Texte zugreifen.
MultilingualTextItemComposition	Klasse	Siemens.Engineering	Auf mehrsprachige Texte zugreifen.
TiaPortalTrustAuthority.Feature-Tokens	Enumerationswert	Siemens.Engineering	Auf Einstellungen des TIA Portals zugreifen.
TiaPortalSetting	Klasse	Siemens.Engineering.Settings	Auf Einstellungen des TIA Portals zugreifen.
TiaPortalSettingComposition	Klasse	Siemens.Engineering.Settings	Auf Einstellungen des TIA Portals zugreifen.
TiaPortalSettingsFolder	Klasse	Siemens.Engineering.Settings	Auf Einstellungen des TIA Portals zugreifen.
TiaPortalSettingsFolderComposition	Klasse	Siemens.Engineering.Settings	Auf Einstellungen des TIA Portals zugreifen.
LanguageAssociation	Klasse	Siemens.Engineering	Auf aktive Sprachen zugreifen.
LanguageComposition.Find	Methode	Siemens.Engineering	Auf aktive Sprachen zugreifen.

Geänderte Elemente im Objektmodell

Name	Typ	Namensraum	Kommentar
PlcConstant	Klasse	Siemens.Engineering.SW.Tags	Herausgegebene Basisklasse von PlcUserConstant und PlcSystemConstant.
PlcTag	Klasse	Siemens.Engineering.SW.Tags	Teilen von PlcConstantComposition
ITargetComparable	Schnittstelle	Siemens.Engineering.Compare	String-Attribut DataTypeName anstelle eines offenen Links DataType.
MultilingualText	Klasse	Siemens.Engineering	Auf mehrsprachige Texte zugreifen.

Wesentliche Änderungen

9.3 Die wichtigsten Änderungen in V14 SP1

Name	Typ	Namensraum	Kommentar
ProjectComposition.Create	Methode	Siemens.Engineering	Parameter geändert zur Verwendung von DirectoryInfo und eines Strings.
Project.Subnets	Attribut	Siemens.Engineering	Auf Subnetze zugreifen
Project.Languages	Attribut	Siemens.Engineering	Verschoben, um Attribut von Siemens.Engineering.LanguageSettings darzustellen und unterstützte Sprachen bereitzustellen

Gelöschte Elemente im Objektmodell

Name	Typ	Namensraum	Kommentar
PlcConstantComposition	Klasse	Siemens.Engineering.SW.Tags	Teilen in PlcSystemConstantComposition und PlcUserConstantComposition.
CompareResultElement.PathInformation	Attribut	Siemens.Engineering.SW.Tags	Nicht mehr verwendet.
MultilingualText.GetText(CultureInfo cultureInfo)	Methode	Siemens.Engineering.Compare	Geändertes Konzept für den Zugriff auf Textelemente von MultilingualText.
TiaPortalTrustAuthority.CustomerIdentification	Enumerationswert	Siemens.Engineering	Nicht mehr verwendet.
TiaPortalTrustAuthority.ElevatedAccessExtensions	Enumerationswert	Siemens.Engineering	Nicht mehr verwendet.

Verhaltensänderungen

Name	Typ	Namensraum	Kommentar
PlcTag.Export(FileInfo path, ExportOptions options)	Methode	Siemens.Engineering.SW.Tags	Der Wert des Attributs LogicalAddress wird jetzt immer in internationaler Mnemonik exportiert. Deutsche Mnemonik wird für den Import noch akzeptiert.
PlcTag.LogicalAddress	Attribut	Siemens.Engineering.SW.Tags	Der Wert des Attributs LogicalAddress wird jetzt immer in internationaler Mnemonik zurückgegeben. Deutsche Mnemonik wird für Schreiben noch akzeptiert.

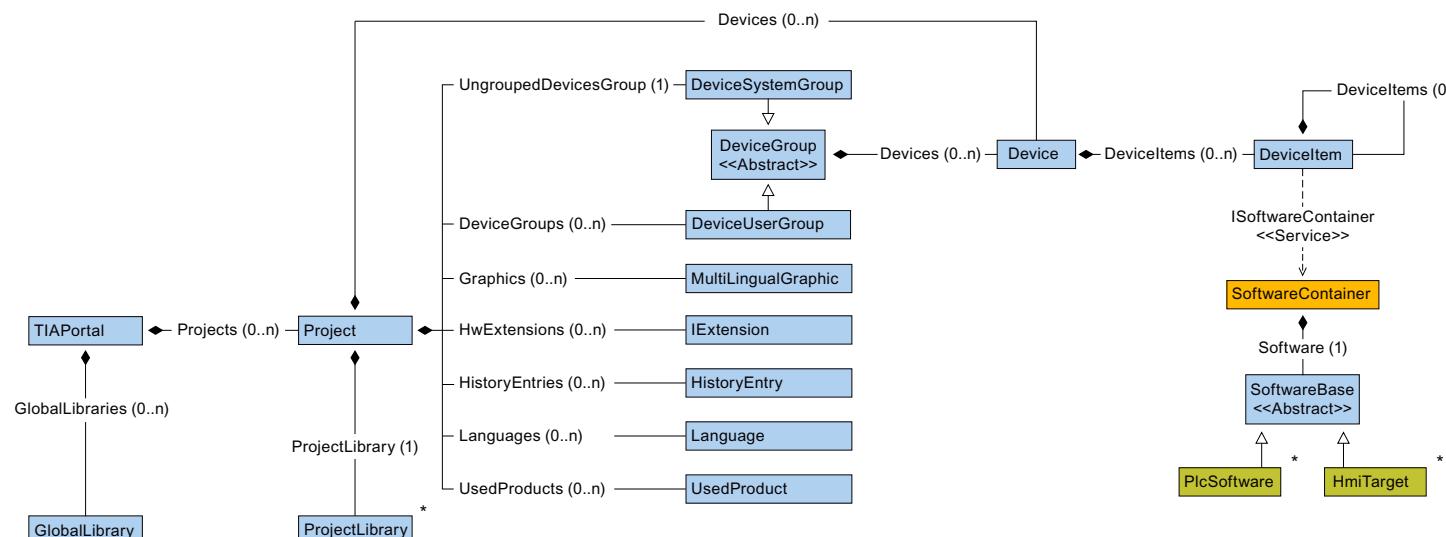
9.3.2 Wesentliche Änderungen im Objektmodell

Objektmodell von TIA Portal Openness V14

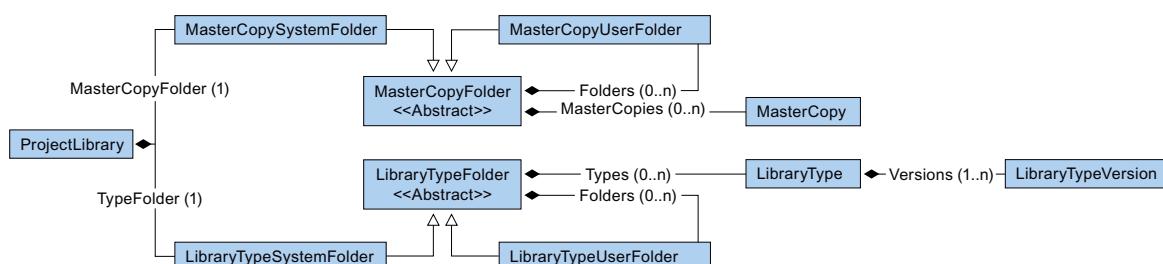
Um Ihnen einen Vergleich zwischen dem alten und dem neuen Objektmodell von TIA Portal Openness zu ermöglichen, beschreibt das nachstehende Diagramm das Objektmodell von TIA Portal V14.

Hinweis

Das in dem Diagramm beschriebene Objektmodell ist veraltet. Sie finden Informationen über das Objektmodell von TIA Portal Openness V14 SP1 unter AUTOHOTSPOT



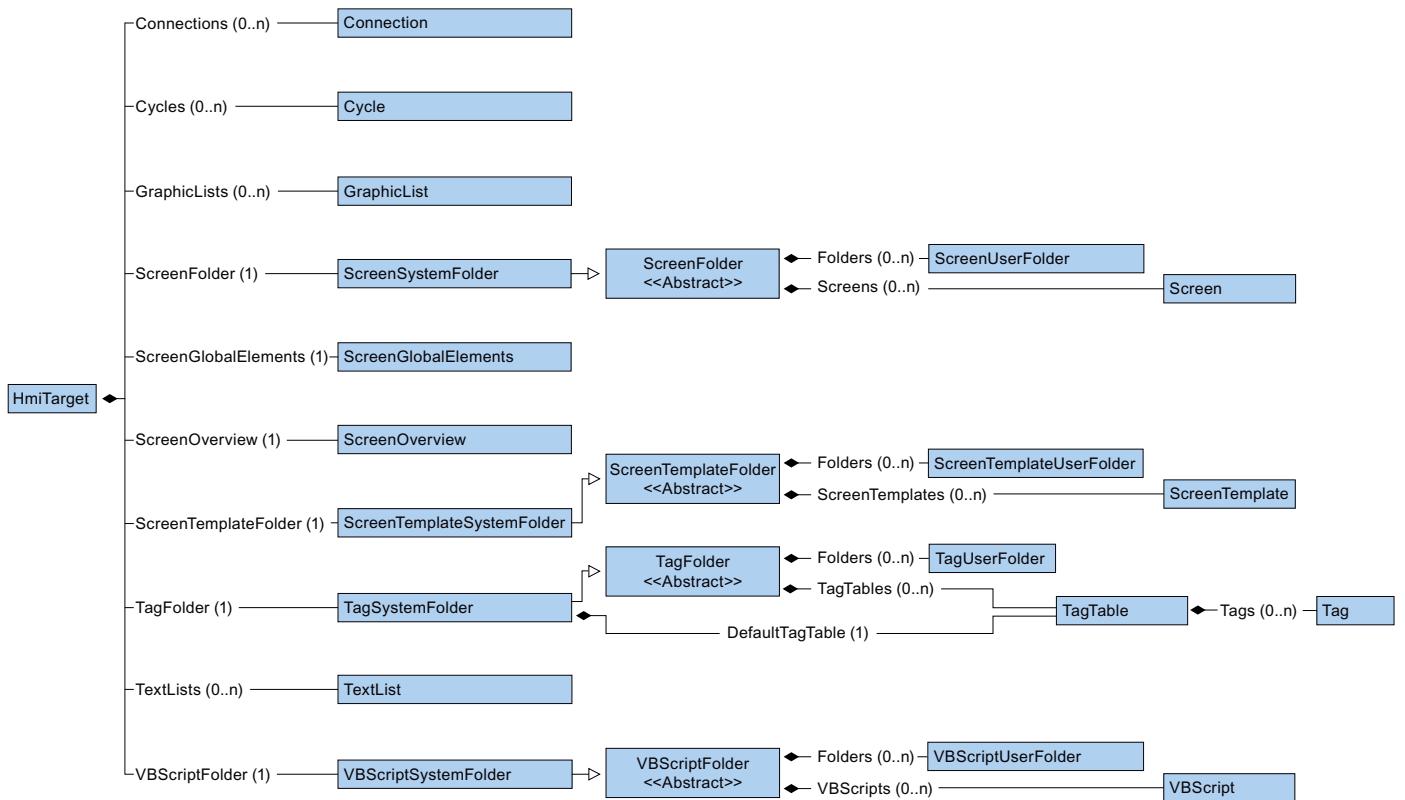
Das folgende Diagramm beschreibt die Objekte, die sich unter ProjectLibrary befinden.



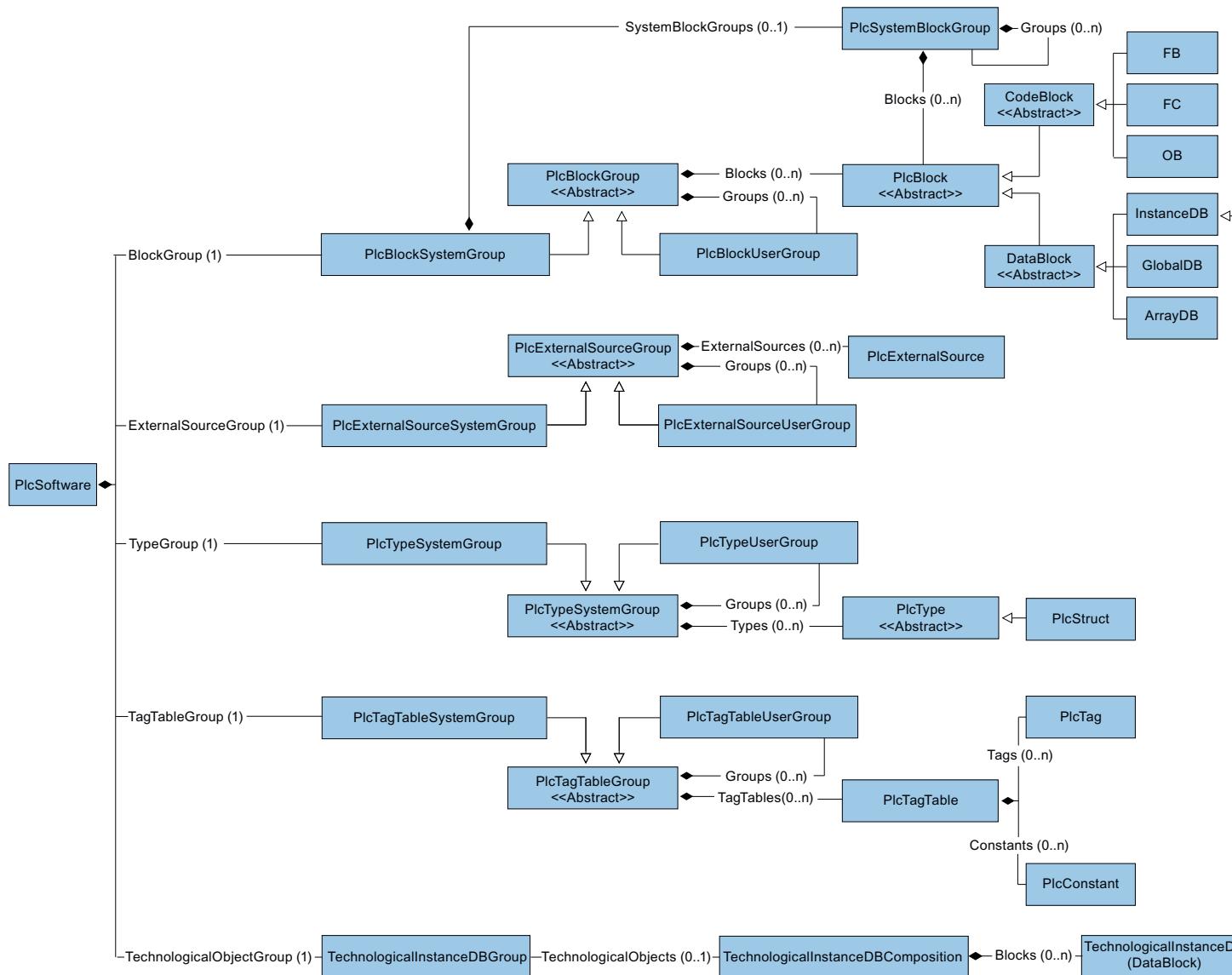
Das folgende Diagramm beschreibt die Objekte, die sich unter HmiTarget befinden.

Wesentliche Änderungen

9.3 Die wichtigsten Änderungen in V14 SP1

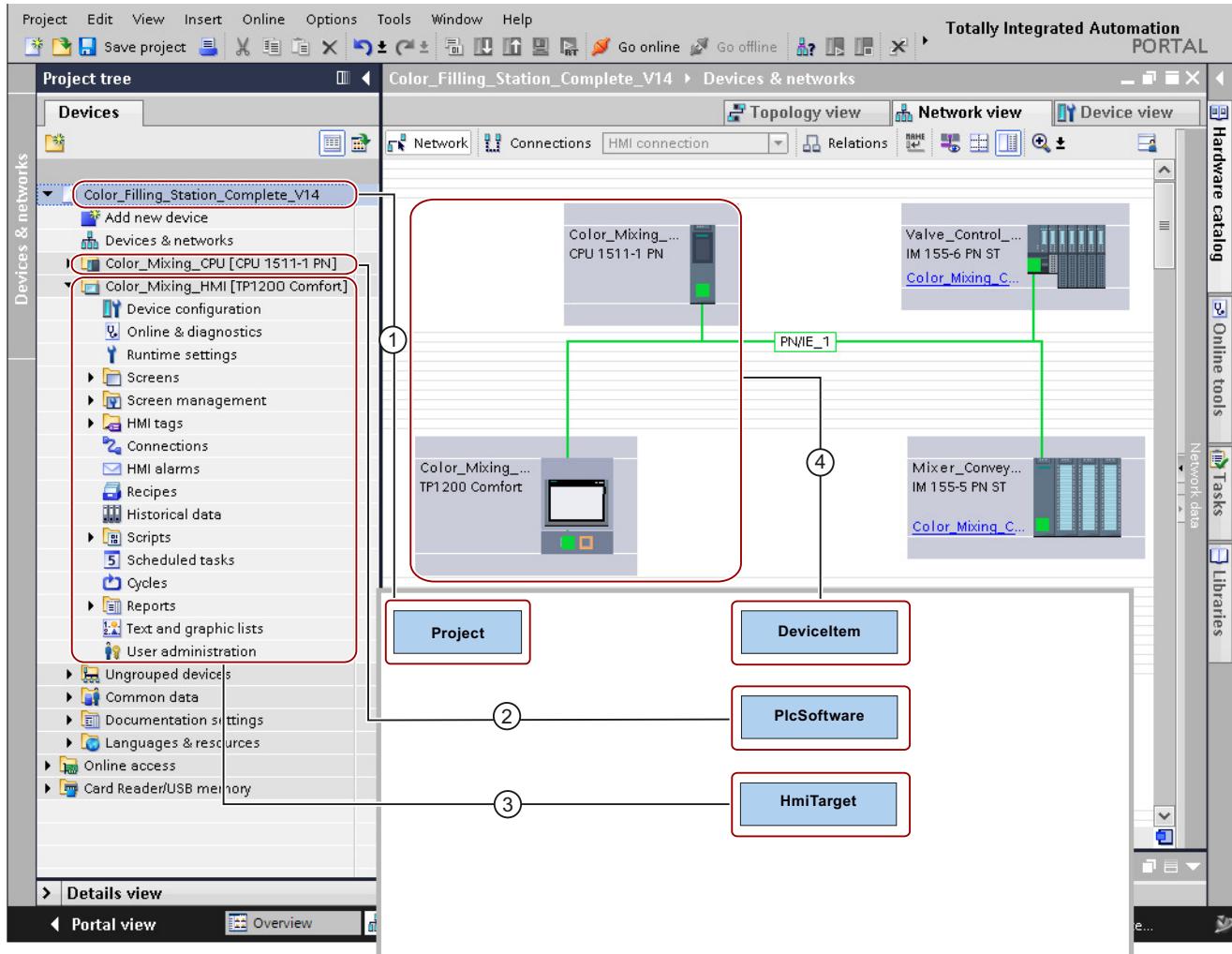


Das folgende Diagramm beschreibt die Objekte, die sich unter PlcSoftware befinden.



Beziehung zwischen TIA Portal und TIA Portal Openness-Objektmodell

Das Bild unten zeigt die Beziehung zwischen dem Objektmodell und einem Projekt im TIA Portal:



- ① Das Objekt "Project" entspricht einem offenen Projekt im TIA Portal.
- ② Das Objekt "PlcSoftware" ist vom Typ "SoftwareBase" ④ und entspricht einem PLC. Der Inhalt des Objekts entspricht einem PLC in der Projektnavigation mit Zugriff auf Objekte wie Bausteine oder PLC-Variablen.
- ③ Das Objekt "HmiTarget" ist vom Typ "SoftwareBase" ④ und entspricht einem Bediengerät. Der Inhalt des Objekts entspricht einem HMI-Gerät in der Projektnavigation mit Zugriff auf Objekte wie Bilder oder HMI-Variablen.
- ④ Das Objekt "DeviceItem" entspricht einem Objekt im Editor "Geräte & Netze". Ein Objekt vom Typ "DeviceItem" kann ein Baugruppenträger oder ein Einschubmodul sein.

9.3.3 Änderungen an der Pilotfunktionalität

Einleitung

Die folgenden Änderungen wurden bei dem API-Objektmodell V14 SP1 vorgenommen und sind nur relevant für Anwender, die die Pilotfunktionalität von HW Konfig in V14 genutzt haben:

Änderungen an TIA Portal Openness API-Typen

TIA Portal Openness API-Typ	Neuer TIA Portal Openness API-Typ
Siemens.Engineering.HW.IAddress	Siemens.Engineering.HW.Address
Siemens.Engineering.HW.IAddressController	Siemens.Engineering.HW.Features.AddressController
Siemens.Engineering.HW.IChannel	Siemens.Engineering.HW.Channel
Siemens.Engineering.HW.IDevice	Siemens.Engineering.HW.Device
Siemens.Engineering.HW.IDeviceItem	Siemens.Engineering.HW.DeviceItem
Siemens.Engineering.HW.IExtension	Siemens.Engineering.HW.Extensions
Siemens.Engineering.HW.IGsd	Siemens.Engineering.HW.Features.GsdObject
Siemens.Engineering.HW.IGsdDevice	Siemens.Engineering.HW.Features.GsdDevice
Siemens.Engineering.HW.IGsdDeviceItem	Siemens.Engineering.HW.Features.GsdDeviceItem
Siemens.Engineering.HW.IHardwareObject	Siemens.Engineering.HW.HardwareObject
Siemens.Engineering.HW.IHwIdentifier	Siemens.Engineering.HW.HwIdentifier
Siemens.Engineering.HW.IHwIdentifierController	Siemens.Engineering.HW.Features.HwIdentifierController
Siemens.Engineering.HW.IIoConnector	Siemens.Engineering.HW.IoConnector
Siemens.Engineering.HW.IIoController	Siemens.Engineering.HW.IoController
Siemens.Engineering.HW.IIoSystem	Siemens.Engineering.HW.IoSystem
Siemens.Engineering.HW.IInterface	Siemens.Engineering.HW.Features.NetworkInterface
Siemens.Engineering.HW.Extensions.ModuleInformationProvider	Siemens.Engineering.HW.Utilities.ModuleInformationProvider
Siemens.Engineering.HW.INode	Siemens.Engineering.HW.Node
Siemens.Engineering.HW.OPCUAExportProvider	Siemens.Engineering.HW.Utilities.OpcUaExportProvider
Siemens.Engineering.HW.IPort	Siemens.Engineering.HW.Features.NetworkPort
Siemens.Engineering.HW.IRole	Siemens.Engineering.HW.Features.HardwareFeature Siemens.Engineering.HW.Features.DeviceFeature Siemens.Engineering.HW.Utilities.ModuleInformationProvider
Siemens.Engineering.HW.SoftwareBase	Siemens.Engineering.HW.Software
Siemens.Engineering.HW.ISubnet	Siemens.Engineering.HW.Subnet
Siemens.Engineering.HW.ISoftwareContainer	Siemens.Engineering.HW.Features.SoftwareContainer
Siemens.Engineering.HW.ISubnetOwner	Siemens.Engineering.HW.Features.SubnetOwner

Änderungen für Enums

TIA Portal Openness API-Typ	Daten-typ	Neuer TIA Portal Openness API-Typ	Daten-typ
Siemens.Engineering.HW.Enums.AddressContext		Siemens.Engineering.HW.AddressContext	
Siemens.Engineering.HW.Enums.AddressIoType		Siemens.Engineering.HW.AddressIoType	
Siemens.Engineering.HW.Enums.Attachment-Type		Siemens.Engineering.HW.MediumAttachment-Type	
Siemens.Engineering.HW.Enums.BaudRate		Siemens.Engineering.HW.BaudRate	
Siemens.Engineering.HW.Enums.BusLoad		Siemens.Engineering.HW.CommunicationLoad	
Siemens.Engineering.HW.Enums.BusProfile		Siemens.Engineering.HW.BusProfile	
Siemens.Engineering.HW.Enums.CableLength		Siemens.Engineering.HW.CableLength	
Siemens.Engineering.HW.Enums.CableName	Ulong	Siemens.Engineering.HW.CableName	Long
Siemens.Engineering.HW.Enums.ChannelIoType	Byte	Siemens.Engineering.HW.ChannelIoType	Int
Siemens.Engineering.HW.Enums.ChannelType	Byte	Siemens.Engineering.HW.ChannelType	Int
Siemens.Engineering.HW.Enums.DeviceItemC-lassifications		Siemens.Engineering.HW.DeviceItemClassificati-ons	
Siemens.Engineering.HW.Enums.InterfaceOpera-tingModes		Siemens.Engineering.HW.InterfaceOperatingMo-des	
Siemens.Engineering.HW.Enums.IpProtocolSe-lection		Siemens.Engineering.HW.IpProtocolSelection	
Siemens.Engineering.HW.Enums.MediaRedun-dancyRole		Siemens.Engineering.HW.MediaRedundancyRole	
Siemens.Engineering.HW.Enums.NetType		Siemens.Engineering.HW.NetType	
Siemens.Engineering.HW.Enums.ProfinetUpdate-TimeMode		entfernt	
Siemens.Engineering.HW.Enums.RtClass	Byte	Siemens.Engineering.HW.RtClass	Int
Siemens.Engineering.HW.Enums.SignalDelaySe-lection	Byte	Siemens.Engineering.HW.SignalDelaySelection	Int
Siemens.Engineering.HW.Enums.SyncRole	Byte	Siemens.Engineering.HW.SyncRole	Int
Siemens.Engineering.HW.Enums.Transmission-RateAndDuplex	Uint	Siemens.Engineering.HW.TransmissionRateAnd-Duplex	Int

Änderungen für Attributwerte von Siemens.Engineering.HW.IoConnector

Attribut	Datentyp	Neuer Name	Datentyp
ProfinetUpdateTimeMode	ProfinetUpdateTimeMo-de	PnUpdateTimeAutoCalculation	Bool
ProfinetUpdateTime		PnUpdateTime	
AdaptUpdateTime		PnUpdateTimeAdaption	
WatchdogFactor		PNWatchdogFactor	
		DeviceNumber	String

Änderungen für Attributwerte von Siemens.Engineering.HW.IoController

Attribut	Datentyp	Neuer Name	Datentyp
		DeviceNumber	String

Änderungen für Attributwerte von Siemens.Engineering.HW.Node

Attribut	Datentyp	Neuer Name	Datentyp
HighestAddress		entfernt, nur am Subnetz verfügbar	
TransmissionSpeed		entfernt, nur am Subnetz verfügbar	
IsoProtocolUsed		UselsoProtocol	
IpProtocolUsed		UseIpProtocol	
RouterAddressUsed		UseRouter	
PnDeviceNameAutoGenerated		PnDeviceNameAutoGeneration	
DeviceNumber		entfernt, zu IO-Connector / IO-Controller verschoben	

Änderungen für Attributwerte von Siemens.Engineering.HW.Subnet

Attribut	Datentyp	Neuer Name	Datentyp
HighestAddress	Byte	HighestAddress	Int
CableConfiguration		PbCableConfiguration	
RepeaterCount		PbRepeaterCount	
CopperCableLength		PbCopperCableLength	
OpticalComponentCount		PbOpticalComponentCount	
OpticalCableLength		PbOpticalCableLength	
OpticalRingEnabled		PbOpticalRing	
OlmP12		PbOlmP12	
OlmG12		PbOlmP12	
OlmG12Eec		PbOlmG12Eec	
OlmG121300		PbOlmG121300	
AdditionalNetworkDevices		PbAdditionalNetworkDevices	
AdditionalDpMaster	Byte	PbAdditionalDpMaster	Int
TotalDpMaster	Byte	PbTotalDpMaster	Int
AdditionalPassiveDevice	Byte	PbAdditionalPassiveDevice	Int
TotalPassiveDevice	Byte	PbTotalPassiveDevice	Int
AdditionalActiveDevice	Byte	PbAdditionalActiveDevice	Int
TotalActiveDevice	Byte	PbTotalActiveDevice	Int
PbCommunicationLoad	BusLoad	PbAdditionalCommunicationLoad	CommunicationLoad
OptimizeDde		PbDirectDateExchange	
MinimizeTslot		PbMinimizeTslotForSlaveFailure	
OptimizeCableConfig		PbOptimizeCableConfiguration	
CyclicDistribution		PbCyclicDistribution	
TslotInit		PbTslotInit	

Wesentliche Änderungen

9.3 Die wichtigsten Änderungen in V14 SP1

Attribut	Datentyp	Neuer Name	Datentyp
Tslot		PbTslot	
MinTsdr		PbMinTsdr	
MaxTsdr		PbMaxTsdr	
Tid1		PbTid1	
Tid2		PbTid2	
Trdy		PbTrdy	
Tset		PbTset	
Tqui		PbTqui	
Ttr		PbTtr	
TtrMs		entfernt	
TtrTypical		PbTtrTypical	
TtrTypicalMs		entfernt	
Watchdog		PbWatchdog	
WatchdogMs		entfernt	
Gap	Byte	PbGapFactor	Int
RetryLimit	Byte	PbRetryLimit	Int
IsochronMode		IsochronousMode	
AdditionalDevice		PbAdditionalPassivDeviceForIsochronousMode	
TotalDevice		PbTotalPassivDeviceForIsochronousMode	
DpCycleTimeAutoCalc		DpCycleMinTimeAutoCalculation	
TiToAutoCalc		IsochronousTiToAutoCalculation	
Ti		IsochronousTi	
To		IsochronousTo	

Änderungen für Attributwerte von Siemens.Engineering.Project

Attribut	Datentyp	Neuer Name	Datentyp
.HwExtensions		.HwUtilities	

Änderungen für Attributwerte von Siemens.Engineering.HW.Baudrate

Attribut	Datentyp	Neuer Name	Datentyp
BaudRate.BAUD_9600		BaudRate.BAUD9600	
BaudRate.BAUD_19200		BaudRate.BAUD19200	
BaudRate.BAUD_45450		BaudRate.BAUD45450	
BaudRate.BAUD_93750		BaudRate.BAUD93750	
BaudRate.BAUD_187500		BaudRate.BAUD187500	
BaudRate.BAUD_500000		BaudRate.BAUD500000	
BaudRate.BAUD_1500000		BaudRate.BAUD1500000	
BaudRate.BAUD_3000000		BaudRate.BAUD3000000	

Attribut	Datentyp	Neuer Name	Datentyp
BaudRate.BAUD_6000000		BaudRate.BAUD6000000	
BaudRate.BAUD_12000000		BaudRate.BAUD12000000	

Änderungen für Attributwerte von Siemens.Engineering.HW.CableLength

Attribut	Datentyp	Neuer Name	Datentyp
CableLength.Unknown		CableLength.None	
CableLength.Length_20m		CableLength.Length20m	
CableLength.Length_50m		CableLength.Length50m	
CableLength.Length_100m		CableLength.Length100m	
CableLength.Length_1000m		CableLength.Length1000m	
CableLength.Length_3000m		CableLength.Length3000m	

Änderungen für Attributwerte von Siemens.Engineering.HW.ChannelloType

Attribut	Datentyp	Neuer Name	Datentyp
ChannelloType.Unknown		ChannelloType.Complex	

Änderungen für Attributwerte von Siemens.Engineering.HW.IpProtocolSelection

Attribut	Datentyp	Neuer Name	Datentyp
IpProtocolSelection.Address-Tailoring		IpProtocolSelection.VialoController	

Änderungen für Attributwerte von Siemens.Engineering.HW.TransmissionRateAndDuplex

Attribut	Daten-typ	Neuer Name	Datentyp
TransmissionRateAndDuplex.Unknown		TransmissionRateAndDuplex.None	
TransmissionRateAndDuplex.TP10Mbps_HalfDuplex		TransmissionRateAndDuplex.TP10MbpsHalfDuplex	
TransmissionRateAndDuplex.TP10Mbps_FullDuplex		TransmissionRateAndDuplex.TP10MbpsFullDuplex	
TransmissionRateAndDuplex.AsyncFiber10Mbps_HalfDuplex		TransmissionRateAndDuplex.AsyncFiber10MbpsHalfDuplex	
TransmissionRateAndDuplex.AsyncFiber10Mbps_FullDuplex		TransmissionRateAndDuplex.AsyncFiber10MbpsFullDuplex	
TransmissionRateAndDuplex.TP100Mbps_HalfDuplex		TransmissionRateAndDuplex.TP100MbpsHalfDuplex	
TransmissionRateAndDuplex.TP100Mbps_FullDuplex		TransmissionRateAndDuplex.TP100MbpsFullDuplex	
TransmissionRateAndDuplex.FO100Mbps_FullDuplex		TransmissionRateAndDuplex.FO100MbpsFullDuplex	

Attribut	Daten-typ	Neuer Name	Datentyp
TransmissionRateAndDuplex.X1000Mbps_FullDuplex		TransmissionRateAndDuplex.X1000MbpsFullDuplex	
TransmissionRateAndDuplex.FO1000Mbps_Full-Duplex_LD		TransmissionRateAndDuplex.FO1000MbpsFull-DuplexLD	
TransmissionRateAndDuplex.FO1000Mbps_Full-Duplex		TransmissionRateAndDuplex.FO1000MbpsFull-Duplex	
TransmissionRateAndDuplex.TP1000Mbps_Full-Duplex		TransmissionRateAndDuplex.TP1000MbpsFull-Duplex	
TransmissionRateAndDuplex.FO10000Mbps_Full-Duplex		TransmissionRateAndDuplex.FO10000MbpsFull-Duplex	
TransmissionRateAndDuplex.FO100Mbps_Full-Duplex_LD		TransmissionRateAndDuplex.FO100MbpsFull-DuplexLD	
TransmissionRateAndDuplex.POFCF100Mbps_FullDuplex_LD		TransmissionRateAndDuplex.POFCF100MbpsFullDuplexLD	

9.3.4 Änderungen bei Export und Import

9.3.4.1 Änderungen bei Export und Import

Einleitung

Der Export und Import mittels TIA Portal Openness API wurde in V14 SP1 erweitert, um Kommentare bei Array-Elementen verarbeiten zu können. Dadurch wurde ein neues Schema erforderlich. Für den Import und Export von Bausteinschnittstellen werden ab jetzt zwei Schemaversionen verwendet:

- Für den Import: Die Entscheidung über die verwendete Schemaversion wird anhand des Namensraums getroffen: <Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
- Für den Export: Die Entscheidung über die verwendete Schemaversion wird anhand der Projektversion getroffen. Projekte V14 SP1 führen zu Version 2, Projekte V14 führen zu Version 1.

9.3.4.2 Änderungen in der API

Quelle generieren

Die folgenden Methoden wurden aus ProgramBlocks entfernt:

- GenerateSourceFromBlocks
- GenerateSourceFromTypes

Die folgenden Methoden wurden ergänzt:

- GenerateSource bis PlcExternalSourceSystemGroup

Beispiel

```
// generate source for V14
var blocks = new List<PlcBlock>() {block1};
var types = new List<PlcBlock>() {udt1};
var fileInfoBlock = new FileInfo("D:\Export\Block.scl");
var fileInfoType = new FileInfo("D:\Export\Type.udt");

PlcBlocksSystemGroup blocksGroup = ...;
blocksGroup.GenerateSourceFromBlocks(blocks, fileInfo);
PlcTypesSystemGroup plcDataTypesGroup = ...;
plcDataTypesGroup.GenerateSourceFromTypes(types, fileInfo);

//generate source as of V14 SP1
var blocks = new List<PlcBlock>() {block1};
var types = new List<PlcBlock>() {udt1};
var fileInfoBlock = new FileInfo("D:\Export\Blocks.scl");
var fileInfoType = new FileInfo("D:\Export\Type.udt");

PlcExternalSourceSystemGroup externalSourceGroup = plc.ExternalSourceGroup;
externalSourceGroup.GenerateSource(blocks, fileInfoBlock);
externalSourceGroup.GenerateSource(types, fileInfoType);
```

9.3.4.3 Schemaerweiterung

Schemaerweiterung für Kommentare und Startwerte

Kommentare und Startwerte werden in dem neuen Element "Subelement" gespeichert, das sich auf das Array-Element mit dem Attribut "Path" bezieht.

Subelement enthält den Startwert und den Kommentar für das referenzierte Array-Element. Das Attribut "Path" an StartValue wird im neuen Schema entfernt.

Schemadefinition von "Subelement":

```
<xs:element name="Subelement" type="Subelement_T"/>
<xs:complexType name="Subelement_T">
  <xs:sequence>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="StartValue"/>
      <xs:element ref="Comment"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="Path" type="IndexPath_TP"/>
</xs:complexType>
```

Erweiterung des Mitgliedstyps:

```
<xs:complexType name="Member_T">
  <xs:sequence>
    <xs:element ref="AttributeList" minOccurs="0" maxOccurs="1"/>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="Member"/>
      <xs:element ref="Sections"/>
      <xs:element ref="StartValue"/>
      <xs:element ref="Comment"/>
      <xs:element ref="Subelement"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

Beispiele:

Speicherung von Kommentaren und Startwerten in einfachen Arrays:

```
<Member Name="Static_1" Datatype="Array[0..1] of Bool">
  <Comment>
    <MultiLanguageText Lang="de-DE">comment for array</MultiLanguageText>
  </Comment>
  <Subelement Path="0">
    <StartValue>true</StartValue>
    <Comment>
      <MultiLanguageText Lang="de-DE">comment for array element 0</MultiLanguageText>
    </Comment>
  </Subelement>
  <Subelement Path="1">
    <StartValue>true</StartValue>
    <Comment>
      <MultiLanguageText Lang="de-DE">comment for array element 1</MultiLanguageText>
    </Comment>
  </Subelement>
</Member>
```

Speicherung von Kommentaren und Startwerten in Arrays aus UDT:

```

<Member Name="Static_1" Datatype="Array[0..1] of &quot;User_data_type_1&quot;">
    <Comment>
        <MultiLanguageText Lang="de-DE">comment for array</MultiLanguageText>
    </Comment>
    <Subelement Path="0">
        <Comment>
            <MultiLanguageText Lang="de-DE">cmt array 0</MultiLanguageText>
        </Comment>
    </Subelement>
    <Sections>
        <Section Name="None">
            <Member Name="Element_1" Datatype="Bool">
                <Subelement Path="0">
                    <StartValue>true</StartValue>
                    <Comment>
                        <MultiLanguageText Lang="de-DE">comment for element 0</MultiLanguageText>
                    </Comment>
                </Subelement>
                <Subelement Path="1">
                    <StartValue>true</StartValue>
                    <Comment>
                        <MultiLanguageText Lang="de-DE">comment for element 1</MultiLanguageText>
                    </Comment>
                </Subelement>
            </Member>
            <Member Name="Element_2" Datatype="Struct">
                <Member Name="Element_1" Datatype="Int">
                    <Subelement Path="0">
                        <StartValue>11</StartValue>
                        <Comment>
                            <MultiLanguageText Lang="de-DE">comment for element 0</MultiLanguageText>
                        </Comment>
                    </Subelement>
                </Member>
                <Member Name="Element_2" Datatype="Bool">
                    <Subelement Path="1">
                        <StartValue>true</StartValue>
                        <Comment>
                            <MultiLanguageText Lang="de-DE">comment for element 1</MultiLanguageText>
                        </Comment>
                    </Subelement>
                </Member>
            </Section>
        </Sections>
    </Member>

```

Speicherung von Kommentaren und Startwerten in Arrays aus Struct:

```

<Member Name="Static_1" Datatype="Array[0..1] of Struct">
    <Member Name="Static_1" Datatype="Int">
        <Subelement Path="0">
            <StartValue>11</StartValue>
            <Comment>
                <MultiLanguageText Lang="de-DE">comment for int elem</MultiLanguageText>
            </Comment>
        </Subelement>
    </Member>
    <Member Name="Static_2" Datatype="Bool">
        <Subelement Path="1">
            <StartValue>true</StartValue>
            <Comment>
                <MultiLanguageText Lang="de-DE">comment for bool elem</MultiLanguageText>
            </Comment>
        </Subelement>
    </Member>
</Member>

```

9.3.4.4 Schemaänderungen

Knoten Access in SW.PlcBlocks.Access.xsd

Das Attribut Type des Knotens Access wurde zu den untergeordneten Knoten von Access verschoben bei

- AbsoluteOffset – erforderlich
- Address – optional

```
<StlStatement UID="22">
  <StlToken Text="L" />
  <Access Scope="Address">
    <Address Area="Local" Type="Word" BitOffset="80" />
  </Access>
</StlStatement>
```

Das Attribut Type von Constant wurde mit dem neuen Unterknoten ConstantType ersetzt.

```
<Access Scope="LocalConstant">
  <IntegerAttribute Name="NumBLs" Informative="true">5</IntegerAttribute>
  <Constant Name="LocalConstant_A">
    <ConstantType Informative="true">Int</ConstantType>
    <ConstantValue Informative="true">10</ConstantValue>
    <StringAttribute Name="Format" Informative="true">Dec_signed</StringAttribute>
  </Constant>
</Access>
```

Der Wert des Attributs Scope in Access wurde in TypedConstant umbenannt, wenn der ConstantValue einen typqualifizierten Wert enthält (z. B.: int#10).

Constant hat kein Attribut Type, wenn ConstantValue einen typqualifizierten Wert enthält (z. B.: int#10).

Lokale Variablen haben keinen Knoten Address, wenn Scope eine LocalVariable ist.

Wenn ein Access auf beliebiger Ebene in einem anderen Access verschachtelt ist, muss nur der äußere Access eine Uid haben.

Knoten Address in SW.PlcBlocks.Access.xsd

Das Attribut BitOffset des Knotens Address ist optional geworden.

Die Deklarationen zum Exportieren des absoluten Zugriffs wurde wie in der folgenden Tabelle gezeigt geändert:

Bereich ab V14 SP1	Typ	Bausteinnummer	Bit-Offset	Beispiel
DB	Block_DB	obligatorisch	verboten	OPN %DB12
DB	unsortiert	vorhanden	obligatorisch	%DB100.DBX10.3
DB	unsortiert	nicht vorhanden	obligatorisch	%DB100.DBX10.3
L	unsortiert	verboten	obligatorisch	%LW10.0

Bereich ab V14 SP1	Typ	Bausteinnummer	Bit-Offset	Beispiel
I Q M	unsortiert	verboten	obligatorisch	%I0.0 %Q0.0 %M0.0
T C	unsortiert	verboten	obligatorisch	%T0 %C1
Block_FC Block_FB	Block_FC Block_FB	obligatorisch	verboten	Aufruf %FB4, %DB5 Input_1 := %FC10 Aufruf %FB4, %DB5 Input_2 := %FB11
Peripherieeingang	unsortiert	verboten	obligatorisch	
Peripherieausgang	unsortiert	verboten	obligatorisch	

Knoten Area in SW.PlcBlocks.Access.xsd

Der Knoten Area hat eine vereinfachte Enum-Liste erhalten:

- LocalC und LocalN wurden zu Local
- DBc, DBv, DBr wurden entfernt.

Knoten CallInfo in SW.PlcBlocks.Access.xsd

Das Attribut Name des Knotens CallInfo ist optional geworden.

Das Attribut BlockType des Knotens CallInfo ist erforderlich geworden.

+2.2.5 Anwenderbausteinaufrufe

Knoten Constant in SW.PlcBlocks.Access.xsd

Der Knoten Constant referenziert den Knoten CostantType mit minOccurs=0.

Der Knoten Constant referenziert den Knoten IntegerAttribute nicht mehr.

Knoten ConstantValue in SW.PlcBlocks.Access.xsd

Der Knoten ConstantValue erhält ein informatives Attribut.

Knoten Instruction in SW.PlcBlocks.Access.xsd

Der Knoten Instruction referenziert den Knoten Acces mit minOccurs=0.

Die Parameterattribute Section, Type und TemplateReference wurden bei Instruction gelöscht.

Knoten Parameter in SW.PlcBlocks.Access.xsd

Das Attribut SectionName des Knotens Parameter ist optional geworden.

Werte für Scope in SW.PlcBlocks.Access.xsd

Die Enum-Liste von Scope wurde erweitert um:

- TypedConstant
- AddressConstant
- LiteralConstant
- AlarmConstant
- Address
- Statusword
- Expression
- Call
- CallWithType

Knoten Statusword in SW.PlcBlocks.Access.xsd

Die Enum-Liste von Statusword wurde erweitert um:

- STW

Knoten ConstantType in SW.PlcBlocks.Access.xsd

Der neue Knoten ConstantType wurde mit dem optional verwendeten Attribut Informativ eingeführt.

Knoten CallRef in SW.PlcBlocks.LADFBDB.xsd

Der Knoten CallRef wurde in Call umbenannt und hat keinen Unterknoten BooleanAttribute mehr.

Knoten InstructionRef in SW.PlcBlocks.LADFBDB.xsd

Der Knoten InstructionRef wurde durch den Knoten Part ersetzt.

Knoten Part in SW.PlcBlocks.LADFBDB.xsd

Der neue Knoten ConstantType wurde eingeführt und ersetzt den Knoten InstructionRef.

- Attribute: Name und Version
- Unterknoten: Der Unterknoten Instruction als neue Wahl neben dem vorhandenen Equation
- hat weder einen Unterknoten BooleanAttribute noch ein Attribut Gate.

Knoten Wire in SW.PlcBlocks.LADFBDB.xsd

Das Attribut Name des Knotens Wire wurde entfernt.

Knoten TemplateReference in SW.PlcBlocks.LADFBD.xsd

Der Knoten TemplateReference wurde gelöscht.

Knoten StatementList in SW.PlcBlocks.STL.xsd

Enum-Liste von StatementList (STL_TE):

- L_STW wurde entfernt
- T_STW wurde entfernt

9.3.4.5 Verhaltensänderungen

Absoluter Zugriff

In V14 wurde der Import des absoluten Zugriffs für die meisten Kombinationen abgebrochen. Ab V14 SP1 funktioniert der Import des absoluten Zugriffs für die folgenden Bereiche:

- Eingang
- Ausgang
- Merker
- Zeit, sofern auf dem PLC unterstützt
- Zähler, sofern auf dem PLC unterstützt
- DB
- DI

Wenn ein symbolischer Zugriff und ein absoluter Zugriff gleichzeitig verwendet und nicht von der Prüfung des Schemas oder der Knotenart abgelehnt werden, ist der Import nur dann erfolgreich, wenn die Boxzugriffsinformationen erfolgreich aufgelöst werden. Wenn der symbolische Zugriff zu anderen Informationen führt als der absolute Zugriff, wird der Import abgelehnt.

```

<Access Scope="Address">
    <Address Area="Memory" Type="Word" BitOffset="0" />
</Access>

<CallInfo Name="Block_1" BlockType="FC">
    <Parameter Name="Input_1" Section="Input" Type="Int" />
    <Parameter Name="Input_1" Section="Input" Type="Int" />
    <!-- Import will be aborted because parameter name 'Input_1' is used more than once -->
    <Parameter Name="Output_1" Section="NotExisting" Type="Int" />
    <!-- Import will be aborted because the section 'NotExisting' can not be used. -->
    <Parameter Name="ENO" Section="Output" Type="Int" />
    <!-- Import will be aborted because the parameter name 'ENO' is restricted and can not be used. -->
</CallInfo>
```

Direkter DB-Zugriff

Ab V14 SP1 kann der indirekte DB-Zugriff nur dann importiert werden, wenn 'Offset', 'Typ' und 'Symbol' angegeben werden.

```
...
<Access Scope="LocalVariable" UID="21">
  <Symbol>
    <Component Name="Output_3" />
    <AbsoluteOffset BitOffset="16" Type="Word" />
  </Symbol>
</Access>
...
...
```

Symbolische und absolute Informationen für lokalen Zugriff

Beim Importieren von "symbolischem Zugriff" werden alle möglichen angegebenen "Informationen für absoluten Zugriff" überprüft, sofern sie nicht als "informativ" gekennzeichnet sind. Ab V14 SP1 wird der Import abgebrochen, wenn die absoluten Informationen nicht übereinstimmen.

Einschränkungen der Bausteinschnittstelle

In V14SP1 werden verschiedene Einschränkungen geprüft. Diese Einschränkungen sind Benutzern des Bausteinschnittstellen-Editors wohlbekannt. Immer wenn der Bausteinschnittstellen-Editor einen Parameter durch Ergänzen oder Erhöhen von '_1' umbenennt, wird der OPNS-Import abgebrochen.

Beispielsweise werden die folgenden Einschränkungen überprüft:

- Doppelte Parameternamen
- Falsche Abschnittsnamen, einschließlich 'Return-Abschnitt' bei FB-Bausteinen
- Eingeschränkte Wörter

Sortierung der Abschnitte beim Import

Wenn der aufgerufene Baustein zum Zeitpunkt des Imports nicht vorhanden ist, wird die Schnittstellendefinition auf Aufrufseite dazu verwendet, den aufgerufenen Anwenderbaustein anzuzeigen. In V14 SP1 werden die Abschnitte in der Reihenfolge sortiert, in der sie in der Bausteinschnittstelle des aufgerufenen Bausteins angezeigt werden würden, wenn dieser mit den gleichen Parametern vorhanden wäre.

Die Abschnittsreihenfolge der importierten Parameter lautet:

- Eingang
- Ausgang

Das folgende Beispiel in AWL xml

```

<StlStatement UIId="21">
<StlToken Text="CALL" />
<Access Scope="Call">
  <CallInfo Name="Block_2" BlockType="FC">
    <Parameter Name="Output_1" Section="Output" Type="Int">
      <Access Scope="GlobalVariable">
        <Symbol>
          <Component Name="Tag_3" />
        </Symbol>
      </Access>
    </Parameter>
    <Parameter Name="Input_1" Section="Input" Type="Int">
      <Access Scope="GlobalVariable">
        <Symbol>
          <Component Name="Tag_1" />
        </Symbol>
      </Access>
    </Parameter>
    <Parameter Name="Output_2" Section="Output" Type="Int">
      <Access Scope="GlobalVariable">
        <Symbol>
          <Component Name="Tag_4" />
        </Symbol>
      </Access>
    </Parameter>
    <Parameter Name="Input_2" Section="Input" Type="Int">
      <Access Scope="GlobalVariable">
        <Symbol>
          <Component Name="Tag_2" />
        </Symbol>
      </Access>
    </Parameter>
  </CallInfo>
</Access>
</StlStatement>

```

führt zu:

CALL "Block_2" Input_1 := "Tag_1" Input_2 := "Tag_2" Output_1 := "Tag_3" Output_2 := "Tag_4"	
--	--

Eindeutige Aufruernamen von Anwenderbausteinen

Im TIA Portal müssen Namen eindeutig sein. Das bedeutet beispielsweise, dass eine Variable nicht den gleichen Namen wie ein Baustein haben kann. Für den XML-Import der TIA Portal Openness API bedeutet dies für den Fall, dass die XML einen Anwenderbaustinaufruf enthält, bei dem der aufgerufene Baustein zum Zeitpunkt des Imports nicht vorhanden ist, der Name des aufgerufenen Bausteins verglichen mit allen vorhandenen Namen im Projekt eindeutig sein muss. Ist der Name des aufgerufenen Bausteins nicht eindeutig, wird der Import abgebrochen.

Im folgenden Beispiel wird der Import abgebrochen, weil der Name des aufgerufenen Bausteins "Tag_1" bereits für eine Variabellabelle verwendet wird.

9.3 Die wichtigsten Änderungen in V14 SP1

```
...
<SW.Tags.PlcTag ID="1" CompositionName="Tags">
  <AttributeList>
    <DataTypeName>Int</DataTypeName>
    <LogicalAddress>%MW2</LogicalAddress>
    <Name>Tag_1</Name>
  </AttributeList>
</SW.Tags.PlcTag>
...
...
<StlStatement UID="21">
  <StlToken Text="CALL" />
  <Access Scope="Call">
    <CallInfo Name="Tag_1" BlockType="FC">
      <Parameter Name="Input_1" Section="Input" Type="Int">
        <Access Scope="GlobalVariable">
          <Symbol>
            <Component Name="Tag_1" />
          </Symbol>
        </Access>
      </Parameter>
    ...
  
```

Im folgenden Beispiel wird der Import abgebrochen, weil zwei Parameter den gleichen Namen "Input1" haben.

```
<StlStatement UID="22">
  <StlToken Text="CALL" />
  <Access Scope="Call">
    <CallInfo Name="Block_1" BlockType="FB">
      <Instance Scope="GlobalVariable">
        <Component Name="Block_1_DB" />
      </Instance>
      <Parameter Name="Input1" Section="Input" Type="Int">
        <Access Scope="GlobalVariable">
          <Symbol>
            <Component Name="Tag_9" />
          </Symbol>
        </Access>
      </Parameter>
      <Parameter Name="Input1" Section="Input" Type="Time">
        <Access Scope="TypedConstant">
          <Constant>
            <ConstantValue>T#1s</ConstantValue>
          </Constant>
        </Access>
      </Parameter>
    </CallInfo>
  </Access>
</StlStatement>
```

Bibliotheksbausteinaufrufe

Die importierte XML kann Aufrufe von Anwenderbausteinen enthalten. Diese Anwenderbausteine werden anhand des Namens identifiziert.

Anwenderbausteine können auch Bibliothekselemente aufrufen. Diese Bibliothekselemente können als 'Bibliotheksbausteinaufrufe' generiert werden. Da Bibliotheksbausteinen denselben Namensraum wie Anwenderbausteine verwenden, kann bei einem anhand des

Namens durchgeführten Imports eines Anwenderbausteinaufrufs die Implementierung eines Anwenderbausteins aufgerufen werden.

Vor V14 SP1 hat der Importvorgang versucht, die Parameter zwischen dem Anwenderbausteinaufruf und dem Anweisungsbausteinaufruf zuzuordnen. Gelegentlich wurde der Import abgebrochen, gelegentlich wurden beim Import alle nicht übereinstimmenden Parameter gelöscht.

Ab V14 SP1 findet der Anwenderbausteinaufruf immer noch den Bibliotheksbaustein, doch der Aufruf wird nicht ungültig.

Nichtübereinstimmung des Bausteintyps

Wenn die XML einen Anwenderbausteinaufruf von 'Block_1' mit mehr Parametern als die entsprechende FC im Projekt enthält, definiert der Import ab V14 SP1 eine neue Schnittstelle für den aufgerufene Baustein, die dem Anwenderbausteinaufruf aus der XML entspricht. Bei der nächsten Übersetzung des Programmbausteins wird versucht, den Aufruf zu aktualisieren.

Neue Funktionsumfänge für Konstanten

Ab V14SP1 wurden verschiedene neue Funktionsumfänge für Konstanten definiert. Der Import ist nur dann erfolgreich, wenn die Werte in der XML-Datei dem Funktionsumfang der Konstante entsprechen. Der Import wird möglicherweise abgebrochen, wenn nicht alle für eine Konstante angegebenen Informationen der vorhandenen Konstante entsprechen.

```
...
<Access Scope="LiteralConstant">
  <Constant>
    <ConstantType>Int</ConstantType>
    <ConstantValue>16#0000_0001</ConstantValue>
  </Constant>
</Access>
...
<Access Scope="TypedConstant">
  <Constant>
    <ConstantValue>Int#10</ConstantValue>
  </Constant>
</Access>
...
<Access Scope="LiteralConstant">
  <Constant>
    <ConstantType>Int</ConstantType>
    <ConstantValue>10</ConstantValue>
  </Constant>
</Access>
...
<Access Scope="GlobalConstant">
  <Constant Name="Constant_1" />
</Access>
...
<Access Scope="LocalConstant">
  <Constant Name="Constant_1" />
</Access>
...
<Access Scope="AddressConstant">
  <Constant Name="Tag_1" />
</Access>
...
<Access Scope="AlarmConstant">
  <Constant>
    <ConstantType>C_Alarm</ConstantType>
    <ConstantValue>16#0000_0001</ConstantValue>
  </Constant>
</Access>
...
```

Anmerkungen von Anweisungsversionen

Ab V14 SP1 können nur Anweisungsversionen importiert werden, die auf dem PLC einsetzbar sind, in den importiert werden soll. Ist in der XML keine Anweisungsversion angemerkt, wird die im PLC ausgewählte Version verwendet. In KOP und FUP gibt es für einige als Anweisungen dargestellte Elemente keine Versionierung. Diese Elemente können nur ohne Version importiert werden.

```
...
<Part Name="MIN" Version="1.0" UIId="27" DisabledENO="false">
  <TemplateValue Name="card" Type="Cardinality">2</TemplateValue>
  <TemplateValue Name="value_type" Type="Type">Int</TemplateValue>
</Part>
<Part Name="MIN" UIId="28" DisabledENO="false">
  <TemplateValue Name="card" Type="Cardinality">2</TemplateValue>
  <TemplateValue Name="value_type" Type="Type">Int</TemplateValue>
</Part>
...
...
```

Deaktivierter ENO

Die Funktion zum Deaktivieren des ENO dient auf S7-1200 und S7-1500 PLCs dazu, Laufzeit verbrauchende Berechnungen des ENO-Verbindungszustand zu deaktivieren.

Ab V14 SP1 kann der Merker DisabledENO nur auf PLCs importiert werden, die diese Funktion unterstützen.

```
...
<Part Name="Add" UIId="24" DisabledENO="false">
  <TemplateValue Name="Card" Type="Cardinality">2</TemplateValue>
  <TemplateValue Name="SrcType" Type="Type">Int</TemplateValue>
</Part>
...
...
```

Typvalidierung für absoluten Zugriff auf den L-Stack

Ab V14 SP1 wird der Import abgebrochen, wenn der Typ nicht verwendet oder zugeordnet werden kann.

Validierung von Index-Ids

Indexzugriff ist verwendbar, wenn die Funktion "Symbolischer Zugriff auf den Speicher" definiert ist, beispielsweise lokaler Zugriff, globaler Zugriff, indirekter Zugriff.

Wenn eine literale Konstante als Index verwendet wird, werden die Typen Ganzzahl mit und ohne Vorzeichen in Dint geändert. Ab V14 SP1 wird der Import abgebrochen, wenn ein Typ außerhalb des angegebenen Bereichs angegeben wird.

Sämtlicher Indexzugriff wird daraufhin geprüft, ob die Art des Zugriffs überhaupt als 'Indexzugriff' verwendet werden kann. Ab V14 SP1 wird der Import abgebrochen, wenn der definierte Indexzugriff nicht verwendet werden kann.

Sortierung der Elementreihenfolge

Ab V14 SP1 werden die Elemente in KOP und FUP in der Reihenfolge der Codeerzeugung sortiert, wo beim Export automatisch möglich. In einigen seltenen Fällen kann die exportierte XML nicht wieder importiert werden. In diesen Fällen muss entweder die XML angepasst werden oder die entsprechenden Netzwerke müssen gelöscht und neu programmiert werden. Doch die Reihenfolge von Leitern und Referenzen ist immer noch nicht zuverlässig.

Alarmkonstanten

Mit V14 SP1 wurden die Übersetzungsprüfungen auf gültige Alarmkonstanten erweitert. Es kann vorkommen, dass aufgrund einer in V14 mit defekten Alarmkonstanten importierten XML ein Projekt in V14 SP1 übersetzbare ist. Öffnen Sie in diesem Fall das relevante Netzwerk im KOP/FUP-Editor und löschen Sie den tatsächlichen Alarmoperanden. Der Editor erstellt automatisch eine neue gültige Alarmkonstante.

```
<FlgNet>
  <Parts>
    <Access Scope="AlarmConstant" UID="21">
      <Constant>
        <ConstantType>C_Alarm</ConstantType>
        <ConstantValue>16#0000_0002</ConstantValue>
      </Constant>
    </Access>
    <Call UID="22">
      <CallInfo Name="Block_1" BlockType="FB">
        <Instance UID="23" Scope="GlobalVariable">
          <Component Name="Block_1_DB" />
        </Instance>
        <Parameter Name="Input_1" Section="Input" Type="C_Alarm" />
      </CallInfo>
    </Call>
  </Parts>
  <Wires>
    <Wire UID="24">
      <PowerRail />
      <NameCon UID="22" Name="en" />
    </Wire>
    <Wire UID="25">
      <IdentCon UID="21" />
      <NameCon UID="22" Name="Input_1" />
    </Wire>
  </Wires>
</FlgNet>
```

Einschränkungen für Instanzen von Anwenderbausteinen und Anweisungen

In V14 war es möglich, FC-Anwenderbausteinaufrufe mit einer Instanz zu importieren und diese Aufrufe gelegentlich sogar zu übersetzen.

Ab V14 SP1 ist der Import von Instanzen nur möglich, wo Instanzen unterstützt werden. Bestehende Projekte mit Instanzen an FC-Anwenderbausteinaufrufen und Anweisungen lassen sich möglicherweise nicht mehr übersetzen. In diesem Fall muss der Aufruf gelöscht und neu programmiert werden. Jeder Versuch, eine Aufrufaktualisierung oder eine andere Art der automatischen Reparatur durchzuführen, schlägt fehl.

EnENO sichtbar

In V14 waren die EN- und ENO-Anschlüsse von 'InstructionRef' abhängig vom Merker ENENO verwendbar oder nicht.

Ab V14SP1 werden vom OPNS während des Imports basierend auf dem Element und der Verdrahtung entweder die EN- oder die ENO-Anschlüsse verwendet. Aufgrund dieser automatischen Erkennung ist eine unterschiedliche Verwendung von EN- und ENO-Anschläßen festzustellen. Höchstwahrscheinlich zeigen nur die Boxen von IEC-Zeiten und IEC-Zählern gewisse Probleme.

UId-Zuweisung

Die Zuweisung von UIds zu Teilen, Zugriffen und Leitern ändert sich mit V14 SP1. Die UIds für Ausdrücke, CallInfo und Operanden müssen in einer Übersetzungseinheit eindeutig sein. Aus der TIA Portal-Perspektive sind die UIds in der XML Schlüssel ohne zusätzliche Bedeutung neben der Identifikation eines Elements.

Prüfen von Zeichenfolgen

Strengere Prüfungen bezüglich Anführungszeichen, Stellvertreterzeichen und Steuerzeichen werden für das Attribut Name durchgeführt beim Import von

- IntegerAttribute
- StringAttribute
- DateAttribute
- AutomaticTyped
- Component
- Invisible
- Label
- NameCon
- Negated
- TemplateValue
- CallInfo
- Instruction
- Parameter
- Part
- Step

Strengere Prüfungen bezüglich Stellvertreterzeichen und Steuerzeichen werden durchgeführt beim Import von

- Bausteintiteln und Netzwerken
- LineComment-Text
- Konstanten Zeichenfolgen (Typen String, WString, Char, Wchar)

9.3 Die wichtigsten Änderungen in V14 SP1

Strengere Prüfungen bezüglich Stellvertreterzeichen und Steuerzeichen (Tab und neue Zeile zulässig) werden durchgeführt beim Import von

- Kommentaren und Netzwerken
- String-Attributen
- Knoten, die mehrsprachige Texte definieren wie Alarmtext, Comments
- Token-Texte

Nichtbeachtung von Groß-/Kleinschreibung bei Vorlagenoperationen und Parametern

Ab V14 SP1 wird die Nichtbeachtung von Vorlagenoperationen für Anweisungen und Aufruf- oder Anweisungsparameter importiert und automatisch korrigiert.

Der folgende Code wird importiert und der fehlerhafte Wert "Eq" wird in "EQ" korrigiert und der fehlerhafte Parameter "iN1" wird in "IN1" korrigiert:

```
<StlStatement UIId="22">
  <StlToken Text="CALL" />
  <Access Scope="Call">
    <Instruction Name="CompType">
      <TemplateValue Name="src_type" Type="Type">Variant</TemplateValue>
      <TemplateValue Name="relation" Type="Operation">Eq</TemplateValue>
      <Parameter Name="iN1">
        <Access Scope="GlobalVariable">
          <Symbol>
            <Component Name="Tag_12" />
          </Symbol>
        </Access>
      </Parameter>
      ...
    </Instruction>
  </Access>
</StlStatement>
```

In Aufrufen verwendete Multiinstanzen

Ab V14 SP1 wird der Import abgebrochen, wenn die in einem Aufruf verwendete Multiinstanz nicht vorhanden ist.

Der folgende Code zeigt ein XML-Beispiel, in dem die Multiinstanz im Schnittstellenabschnitt korrekt definiert ist:

```

<SW.Blocks.FB ID="0">
  <AttributeList>
    <Interface>
      <Sections xmlns="http://www.siemens.com/automation/Openness/SW/Interface/v2">
        <Section Name="Input" />
        <Section Name="Output" />
        <Section Name="InOut" />
        <Section Name="Static">
          <!-- The next line must be present if multiinstance is used in code-->
          <Member Name="Static_1" Datatype=""Block_2"" />
        </Section>
      ....
    <StlStatement UID="22">
      <StlToken Text="CALL" />
      <Access Scope="Call">
        <CallInfo BlockType="FB">
          <!-- Multiinstance usage-->
          <Instance Scope="LocalVariable">
            <Component Name="Static_1" />
          </Instance>
          <Parameter Name="Input_1" Section="Input" Type="Int">
            <Access Scope="GlobalVariable">
              <Symbol>
                <Component Name="Tag_9" />
              </Symbol>
            </Access>
            </Parameter>
          </CallInfo>
        </Access>
      </StlStatement>
    
```

Vorlagenkardinalitäten in AWL

In AWL haben die Vorlagenkardinalitäten für jede Anweisung einen festen Standardwert, der der einzige gültige Wert ist. Ab V14 SP1 wird der Import abgebrochen, wenn ein anderer Wert für die Kardinalität verwendet wird.

Indirekten Zugriff importieren

Ab V14 SP1 kann der indirekte Zugriff nur dort importiert werden, wo er übersetzt werden kann.

```

<StlStatement UID="22">
  <StlToken Text="L" />
  <Access Scope="Address">
    <Indirect Width="Word" Area="Memory">
      <Access Scope="LocalVariable">
        <Symbol>
          <Component Name="Temp_1" />
        </Symbol>
      </Access>
    </Indirect>
  </Access>
</StlStatement>

```

Statuswörter Importieren

Ab V14 SP1 kann das Statuswort nur bei Anweisungen importiert werden, bei denen es unterstützt wird.

- L - Unterstütztes Statuswort: STW
- T - Unterstütztes Statuswort: STW
- A - Unterstütztes Statuswort: BR, OV, OS, EQ, NE, GT, Lt, GE, LE, U0, NU
- AN - Unterstütztes Statuswort: BR, OV, OS, EQ, NE, GT, Lt, GE, LE, U0, NU
- O - Unterstütztes Statuswort: BR, OV, OS, EQ, NE, GT, Lt, GE, LE, U0, NU
- ON - Unterstütztes Statuswort: BR, OV, OS, EQ, NE, GT, Lt, GE, LE, U0, NU
- X - Unterstütztes Statuswort: BR, OV, OS, EQ, NE, GT, Lt, GE, LE, U0, NU
- XN - Unterstütztes Statuswort: BR, OV, OS, EQ, NE, GT, Lt, GE, LE, U0, NU

Hinweis

Die meisten Statuswörter sind nur bei S7-300 und S7-400 PLCs nützlich.

Leere Anweisungen

Der Import wird abgebrochen, wenn eine Anweisung keinen Knoten <StlStatement/> hat. Bei einer leeren Anweisung ergänzen Sie den Knoten <StlToken Text="Empty_Line" />.

Der Import wird abgebrochen, wenn eine leere Anweisung über Kommentare verfügt. Bei einer Anweisung nur mit Kommentaren verwenden Sie <StlToken Text="COMMENT" />.

```
<!-- Declaration of an empty statement -->
<StlStatement UID="23">
  <StlToken Text="EMPTY_LINE" />
</StlStatement>

<!-- Declaration of a statement with only comments-->
<StlStatement UID="22">
  <LineComment>
    <Text>Comment number 1</Text>
  </LineComment>
  <StlToken Text="COMMENT" />
</StlStatement>
```

9.3.4.6 Änderungen an Bausteinattributen

Änderungen an allgemeinen Attributen

AutoNumber hat einen neuen Standardwert (false) bei klassischen OBs.

HeaderVersion hat einen neuen Typ System.Version (anstelle von String).

IsKnowHowProtected wird auch auf benutzerdefinierte Datentypen angewendet.

ILibraryTypeInstance.ConnectedVersion, ILibraryTypeInstance.Dependencies, ILibraryTypeInstance.Dependents wurden aus der Tabelle der allgemeinen Attribute entfernt, weil sie weder in XML exportiert werden noch über API zugänglich sind.

MemoryLayout hat einen neuen Standardwert: Standard bei klassischen PLCs und Optimized bei Plus-PLCs.

Number wird auch auf benutzerdefinierte Datentypen angewendet, und es wird in XML dargestellt und ist auch über API zugänglich.

Änderungen an spezifischen Attributen

IsOnlyStoredInLoadMemory und IsWriteProtectedInAS sind jetzt für IDBofUDT schreibgeschützt, sofern der UDT zu einem Systembibliothekselement gehört.

OfSystemLibElement und OfSystemLibVersion gehören jetzt nicht mehr zu den allgemeinen, sondern zu den spezifischen Attributen.

OfSystemLibVersion hat einen neuen Typ System.Version (anstelle von String).

ParameterPassing bleibt Lesen/Schreiben bei FCs und FBs nur, wenn

- ProgrammingLanguage AWL ist und
- MemoryLayout Standard ist und
- die Schnittstelle leer ist

GraphVersion hat einen neuen Typ System.Version (anstelle von String).

Ein neues Attribut namens ExtensionBlockName wird für in Graph geschriebene FBs (ab Graph Version V4) eingeführt.

Ein neues Attribut namens InvalidValuesAcquisition wird für in Graph geschriebene FBs (ab Graph Version V4) eingeführt.

Ein neues Attribut namens IsWriteProtected wird für Codebausteine eingeführt.

DownloadWithoutReinit ist jetzt schreibgeschützt und gilt auch für IDBofFBs.

Supervisions ist jetzt bei IDBofFBs schreibgeschützt.

Änderungen in Enums

Die Enum-Werte für ProgrammingLanguage wurden wie folgt geändert:

- Ein neuer Enum-Wert F_CALL wurde eingeführt.
- Ein neuer Enum-Wert Motion_DB wurde für das Technologieobjekt Motion eingeführt.
- GRAPH_SEQUENCE, GRAPH_ACTIONS, GRAPH_ADDINFOS wurden aus den Enums gelöscht. Sie wurden durch GRAPH ersetzt.

Die Enum-Werte für BlockType wurden wie folgt geändert:

- Die Werte OB, FC, DB, SFC wurden gelöscht, weil diese Enums lediglich beim Attribut InstanceOfType verwendet werden.

9.4 Die wichtigsten Änderungen in V14

9.4.1 Wesentliche Änderungen des Objektmodells

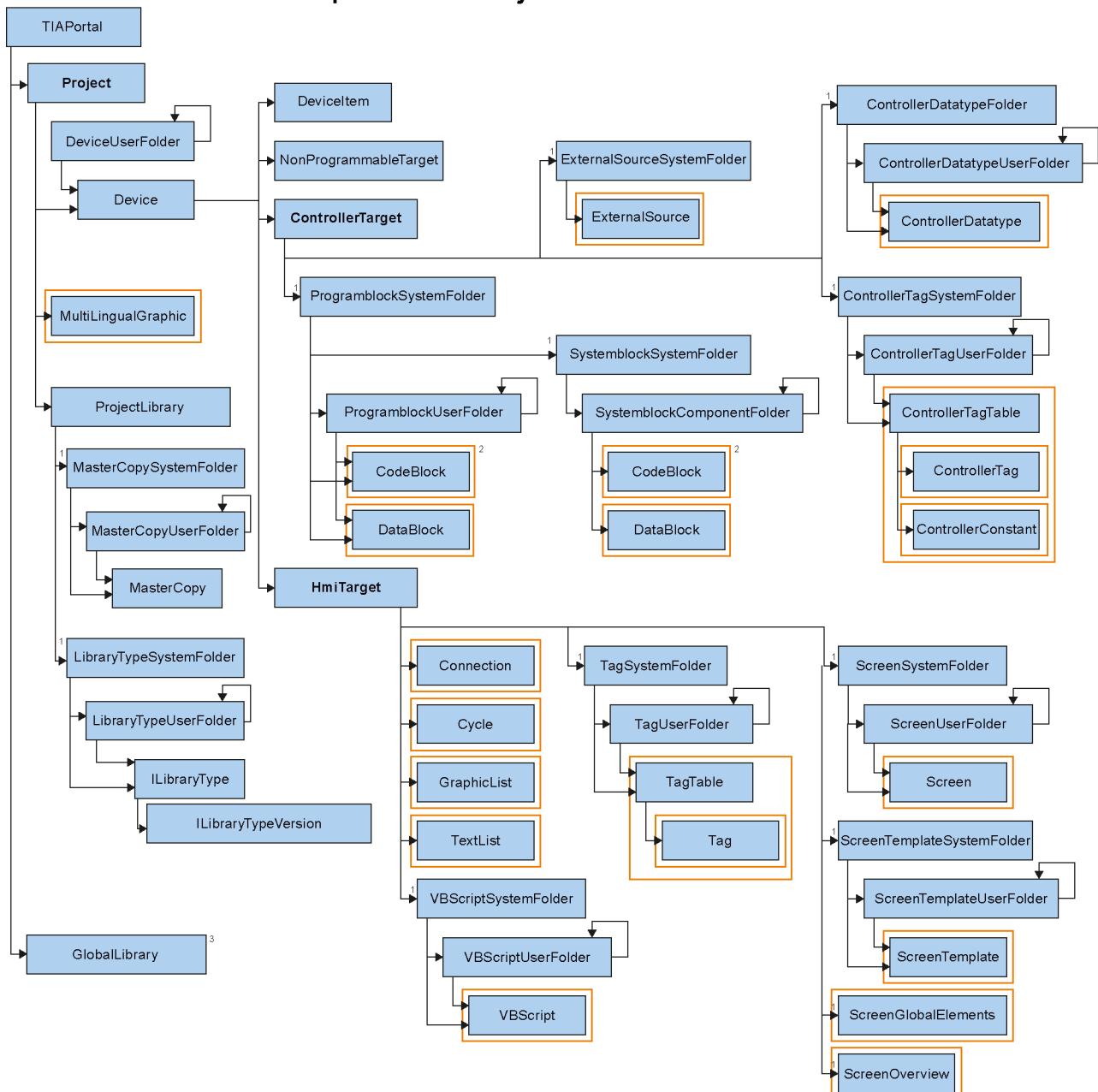
Objektmodell von TIA Portal Openness V13 SP1 und älter

Um Ihnen einen Vergleich zwischen dem alten und dem neuen Objektmodell von TIA Portal Openness zu ermöglichen, beschreibt das nachstehende Diagramm das Objektmodell von TIA Portal V13 SP1.

Hinweis

Das in dem Diagramm beschriebene Objektmodell ist veraltet. Sie finden Informationen über das Objektmodell von TIA Portal Openness V14 SP1 unter AUTOHOTSPOT

Openness object model V13 SP1



9.4.2 Vor dem Hochrüsten einer Anwendung auf TIA Portal Openness V14

Anwendung

Vor dem Hochrüsten einer Anwendung auf TIA Portal Openness V14 müssen die folgenden Einstellungen geändert werden:

1. Die Verweise auf die V14 API sind durch Ergänzen der folgenden TIA Portal Openness APIs anzupassen:
 - Siemens.Engineering
 - Siemens.Engfineering.Hmi
2. Das .Net-Framework von Visual Studio auf Version 4.6.1 ändern
3. Aktualisieren Sie die AssemblyResolve-Methode durch Anpassen des neuen Installationspfads des TIA Portal.
 - Arbeiten Sie aus der Registrierungsdatei, passen Sie den neuen Schlüssel wie im folgenden Beispiel an:
`"HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation_InstalledSW\\TIAP14\TIA_Opns\..."`
 - Arbeiten Sie mit der Anwendungs-Konfigurationsdatei, passen Sie die Pfade an den neuen Installationspfad an.

9.4.3 Wesentliche Stringänderungen

Einleitung

An TIA Portal Openess V14 wurden die folgenden Änderungen vorgenommen, was möglicherweise Auswirkungen auf Ihre vorhandenen Anwendungen hat:

Änderung	Erforderliche Anpassung von Programmcode
Die Übersetzungsmethoden wurden geändert.	<p>Ändern Sie die Übersetzungsmethoden wie im folgenden Beispiel:</p> <ul style="list-style-type: none"> • TIA Portal Openess V13 SP1(veraltet): <code>controllerTarget.Compile(CompilerOptions. Software, BuildOptions.Rebuild);</code> • TIA Portal Openess V14: <code>plcSoftware.GetService<ICompilable>().Com pile();</code>
Es wurden neue Namensräume hinzugefügt.	<ol style="list-style-type: none"> 1. Fügen Sie die folgenden Namensraumanweisungen hinzu: <code>Siemens.Engineering.SW.Blocks;</code> <code>Siemens.Engineering.SW.ExternalSources;</code> <code>Siemens.Engineering.SW.Tags;</code> <code>Siemens.Engineering.SW.Types;</code> 2. Entfernen Sie die Namensraumanweisung <code>using ControllerTarget = Siemens.Engineering.HW.ControllerTarget.</code> 3. Übersetzen Sie die Anwendung.
ControllerTarget wurde durch PlcSoftware ersetzt und in einigen Fällen hat sich Funktionalität geändert.	<ol style="list-style-type: none"> 1. Prüfen Sie die Codebeispiele in der Dokumentation, die zu Ihrer Anwendungsfunktionalität gehört. 2. Aktualisieren Sie den Programmcode Ihrer TIA Portal Openess-Anwendung nach dem folgenden Beispiel: <ul style="list-style-type: none"> – TIA Portal Openess V13 SP1(veraltet): <code>ControllerTarget controllerTarget = deviceItem as ControllerTarget</code> – TIA Portal Openess V14: <code>PlcSoftware plcSoftware = deviceItem.GetService<SoftwareContainer >().Software as PlcSoftware</code> 3. Übersetzen Sie die Anwendung.

Wesentliche Änderungen

9.4 Die wichtigsten Änderungen in V14

Änderung	Erforderliche Anpassung von Programmcode
Es wurden Objekte ersetzt.	<p>1. Suchen und ersetzen Sie die folgenden Objekte:</p> <pre>DeviceUserFolderAggregation = DeviceUserGroupComposition DeviceFolders = DeviceGroups DeviceUserFolder = DeviceUserGroup ProgramblockSystemFolder = PlcBlockSystemGroup ProgramblockUserFolder = PlcBlockUserGroup IBlock = PlcBlock ControllerDatatypeSystemFolder = PlcTypeSystemGroup ControllerDatatypeUserFolder = PlcTypeUserGroup ControllerDatatype = PlcType ControllerTagSystemFolder = PlcTagTableSystemGroup ControllerTagUserFolder = PlcTagTableUserGroup ControllerTagTable = PlcTagTable ControllerTag = PlcTag ControllerConstant = PlcConstant ExternalSourceSystemFolder = PlcExternalSourceSystemGroup ExternalSource = PlcExternalSource IOnline = OnlineProvider ILibraryType = LibraryType</pre> <p>2. Übersetzen Sie die Anwendung.</p>
Aggregationen wurden durch Zusammensetzungen ersetzt.	<p>1. Ersetzen Sie jede Aggregation in Ihrem Code durch Composition wie in den folgenden Beispielen:</p> <pre>ProjectAggregation = ProjectComposition IDeviceAggregation = IDeviceComposition TagTableAggregation = TagTableComposition CycleAggregation = CycleComposition GraphicListAggregation = GraphicListComposition TextListAggregation = TextListComposition ConnectionAggregation = ConnectionComposition MultiLingualGraphicAggregation = MultiLingualGraphicComposition UpdateCheckResultMessageAggregation = UpdateCheckResultMessageComposition</pre> <p>2. Übersetzen Sie die Anwendung.</p>
Ordner wurden in jeder Beziehung außer zu Bediengeräten durch Gruppen ersetzt.	<p>1. Ersetzen Sie mit Ausnahme von Codeteilen, die Bediengeräte betreffen, jeden Folder in Ihrem Programmcode durch Group.</p> <p>2. Übersetzen Sie die Anwendung.</p>

Änderung	Erforderliche Anpassung von Programmcode
Die Methode <code>GetAttributeNames</code> wurde durch die Methode <code>GetAttributeInfos</code> ersetzt.	<ol style="list-style-type: none"> 1. Verwenden Sie <code>IList<EngineeringAttributeInfo></code> <code>IEngineeringObject.GetAttributeInfos(AttributeAccessMode attributeAccessMode);</code>, um Attribute zu ermitteln. 2. Übersetzen Sie die Anwendung. Ausführlichere Informationen finden Sie unter Objektstruktur und -Attribute ermitteln (Seite 127).
Die Methode <code>Close</code> zum Schließen eines Objekts hat sich geändert.	<ol style="list-style-type: none"> 1. Ersetzen Sie <code>project.Close(CloseMode.PromptIfModified);</code> durch <code>project.Close();</code>. 2. Übersetzen Sie die Anwendung. Ausführlichere Informationen finden Sie unter Projekt schließen (Seite 139).
Gleichzeitiger Zugriff wurde durch exklusiven Zugriff und exklusive Transaktionen ersetzt.	<ol style="list-style-type: none"> 1. Ersetzen Sie den gleichzeitigen Zugriff durch exklusiven Zugriff und exklusive Transaktionen wie in den folgenden Beispielen: <ul style="list-style-type: none"> - TIA Portal Openness V13 SP1(veraltet): <code>tiaProject.StartTransaction("Resetting project to default");</code> <code>...</code> <code>tiaProject.CommitTransaction();</code> - TIA Portal Openness V14: <code>//Use exclusive access to avoid user changes</code> <code>ExclusiveAccess exclusiveAccess =</code> <code>tiaPortal.ExclusiveAccess();</code> <code>...</code> <code>exclusiveAccess.Dispose();</code> <code>//Use transaction to be able to rollbank changes:</code> <code>Transaction transaction =</code> <code>exclusiveAccess.Transaction(tiaProject,</code> <code>"Compiling device");</code> <code>transaction.CommitOnDispose();</code> 2. Übersetzen Sie die Anwendung. Weitere Informationen hierzu siehe Ausschließlicher Zugriff (Seite 95) und Transaktionsbehandlung (Seite 104).

Wesentliche Änderungen

9.4 Die wichtigsten Änderungen in V14

Änderung	Erforderliche Anpassung von Programmcode
Der Online-Zugriff auf die CPU wurde geändert	<ol style="list-style-type: none">1. Ändern Sie den Online-Zugriff auf die CPU wie in den folgenden Beispielen:<ul style="list-style-type: none">– TIA Portal Openess V13 SP1(veraltet): <code>((IOnline) controllerTarget).GoOffline() ;</code>– TIA Portal Openess V14: <code>((DeviceItem) plcSoftware.Parent.Parent).GetService<O nlin eProvider>().GoOffline();</code>2. Übersetzen Sie die Anwendung.
Die Hardware-Konfiguration wurde geändert	<ol style="list-style-type: none">1. Ändern Sie die Hardware-Konfiguration: <code>Device.Elements = Device.Items</code>2. Entfernen Sie die folgenden Hardware-Attribute:<ul style="list-style-type: none">– Device.InternalDeviceItem– Device.SubType3. Übersetzen Sie die Anwendung.

Siehe auch

[Umgang mit Exceptions \(Seite 766\)](#)

[Verbindung zum TIA Portal aufbauen \(Seite 79\)](#)

9.4.4 Importieren von Dateien, die mit TIA Portal Openess V13 SP1 und älter erzeugt wurden

Anwendung

Wenn Sie versuchen, Dateien zu importieren, die mit TIA Portal Openess V13 SP1 oder älter erzeugt wurden, wird eine Ausnahme wegen Inkompatibilität ausgelöst. Grund dafür sind Änderungen bei den HMI-Variablen und HMI-Bildschirmen. Die folgenden Tabellen zeigen die wichtigsten Attributänderungen. Ausführlichere Informationen finden Sie im Kapitel "Bilder erstellen Arbeiten mit Objekten und Objektgruppen > Arbeiten mit Objekten > Konfigurieren von Bereichen" in der Online-Hilfe des TIA Portal:

Änderungen von HMI-Variablen

Die folgende Tabelle zeigt die wichtigsten Änderungen der HMI-Variablenattribute:

Entfernte Attribute	Hinzugefügte Attribute
RangeMaximumType	LimitUpper2Type.
RangeMaximum	LimitUpper2.
RangeMinimumType	LimitLower2Type.
RangeMinimum	LimitLower2.
	LimitUpper1Type
	LimitUpper1
	LimitLower1Type
	LimitLower1

Änderungen von HMI-Bildelementen

Die folgende Tabelle zeigt die wichtigsten Änderungen der Schieberegler-Attribute:

Entfernte Attribute	Hinzugefügte Attribute
	RangeLower1Color RangeLower1Enabled RangeLower2Color RangeLower2Enabled RangeNormalColor RangeNormalEnabled RangeUpper1Color RangeUpper1Enabled RangeUpper2Color RangeUpper2Enabled ScalePosition ShowLimitLines ShowLimitMarkers ShowLimitRanges

Die folgende Tabelle zeigt die wichtigsten Änderungen der Messbereichsattribute:

Entfernte Attribute	Hinzugefügte Attribute
DangerRangeColor	RangeLower1Color
DangerRangeStart	RangeLower1Enabled
DangerRangeVisible	RangeLower2Color
WarningRangeColor	RangeLower2Enabled
WarningRangeStart	RangeNormalColor
WarningRangeVisible	RangeNormalEnabled
	RangeUpper1Color
	RangeUpper1Enabled
	RangeUpper1Start
	RangeUpper2Color
	RangeUpper2Enabled
	RangeUpper2Start

Die folgende Tabelle zeigt die wichtigsten Änderungen der Balkenattribute:

Entfernte Attribute	Hinzugefügte Attribute
AlarmLowerLimitColor	RangeLower1Color
AlarmUpperLimitColor	RangeLower1Enabled
	RangeLower2Color
	RangeLower2Enabled
	RangeNormalColor
	RangeNormalEnabled
	RangeUpper1Color
	RangeUpper1Enabled
	RangeUpper2Color
	RangeUpper2Enabled

Index

A

Abfragen

- Attribut „Consistency“ eines Bausteins, 520
- Bausteinautor, 520
- Bausteinfamilie, 520
- Bausteinname, 520
- Bausteinnummer, 520
- Bausteinstitel, 520
- BausteinTyp, 520
- Bausteinversion, 520
- Finden, 515
- Informationen aus einer PLC-VariablenTabelle, 588
- Informationen des Anwenderdatentyps, 520
- Informationen des Bausteins, 520
- Ordner „Programmbausteine“, 514
- Systemordner für PLC-Variablen, 584
- Technologieobjekt, 546
- Zeitstempel eines Bausteins, 520

Allgemeine Einstellungen des TIA Portals, 116

Anwenderdatentyp

- Exportieren, 898
- Importieren, 917
- Informationen abfragen, 520
- Löschen, 536
- Quelle generieren, 531

Anwendungsbeispiel Public API, 72

Ausnahmen

- Beim Zugriff auf das TIA Portal über öffentliche APIs, 766

B

Baustein

- Exportieren, 898
- Gruppe erzeugen, 524
- Gruppe löschen, 524
- Importieren, 897
- Informationen abfragen, 520
- Löschen, 523
- Quelle generieren, 531

Baustineditor

- Starten, 537

Bearbeitungssituation

- Ihre Openness-Anwendung und das TIA Portal befinden sich auf demselben Computer, 50

Beenden der Verbindung zum TIA Portal, 88

Beispielprogramm, 53

Besondere Überlegungen für HMI-Variablen des Datentyps "UDT", 803

Bibliothek

- Auf Ordner zugreifen, 155
- Funktionen, 142
- Typversionen von Instanzen ermitteln, 176

D

Datentypen

- Technologieobjekt, 545

E

Editor „Geräte & Netze“

- Öffnen, 185

Enumrieren

- Alle Variablen einer VariablenTabelle, 276
- Bausteine, 518
- Benutzerdefinierte Bausteinordner, 517
- Benutzerdefinierte Ordner für PLC-Variablen, 585
- Geräte, 236, 239
- Geräteelemente, 252
- Mehrsprachige Texte, 125, 130
- Parameter eines Technologieobjekts, 551
- PLC-Variablen, 591
- PLC-VariablenTabellen, 587
- Systemunterordner, 515
- Technologieobjekt, 550

Erstellen

- Benutzerdefinierte Bilderordner erzeugen, 270
- Benutzerdefinierte Ordner für HMI-Variablen, 276
- Benutzerdefinierte Unterordner für Skripte, 278
- Benutzerdefinierter Ordner für PLC-VariablenTabellen, 586
- Gruppe für Baustein, 524
- Messtaster, 562
- Nocken, 562
- Nockenspur, 562
- Technologieobjekt, 547

Export/Import

- Verwendung, 39

Exportdatei

- Grundstruktur, 790, 921, 925
- Inhalt, 779
- Struktur der XML-Datei, 790, 925

Exportierbare Bildobjekte, 813

Exportieren

Anwenderdatentyp, 898

Baustein, 898

Einzelne Variable oder Konstante aus einer PLC-Variablenliste, 914

F

Finden

Messtaster, 562

Nocken, 562

Nockenspur, 562

Parameter eines Technologieobjekts, 552

Technologieobjekt, 550

Funktionen, 53

Alle Bilder löschen, 272

Allgemein, 79, 87, 88

Allgemeine Einstellungen des TIA Portals, 116

Anwendungsbeispiel Public API, 72

Attribut „Consistency“ eines Bausteins

abfragen, 520

Bausteinautor abfragen, 520

Bausteine enumerieren, 518

Bausteinfamilie abfragen, 520

Bausteinname abfragen, 520

Bausteinnummer abfragen, 520

Bausteinstitel abfragen, 520

Bausteintyp abfragen, 520

Bausteinversion abfragen, 520

Begrenzung auf Projekte von TIA Portal V13, 111

Benutzerdefinierte Bausteinordner

enumerieren, 517

Benutzerdefinierte Bilderordner erzeugen, 270

Benutzerdefinierte Ordner für HMI-Variablen erzeugen, 276

Benutzerdefinierte Ordner für PLC-Variablen enumerieren, 585

Benutzerdefinierte Unterordner für Skripte erzeugen, 278

Benutzerdefinierten Ordner für PLC-Variablenlisten erzeugen, 586

Benutzerdefinierten Ordner für PLC-Variablenlisten löschen, 587

Bild löschen, 271

Bildvorlage löschen, 272

Einlesen der Uhrzeit der letzten Änderungen in eine PLC-Variablenliste, 590

Geräte enumerieren, 236, 239

Geräteelemente enumerieren, 252

Grafikliste löschen, 275

Grafiksammlungen löschen, 135

HMI, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279

Informationen aus einer PLC-Variablenliste abfragen, 588

Mehrsprachige Texte enumerieren, 125, 130

Ordner „Programmbausteine“ abfragen, 514

PLC, 514, 515, 517, 518, 520, 586, 587, 590, 592, 593, 914, 916

PLC-Konstanten, 593

PLC-Target und HMI-Target abfragen, 185

PLC-Variablen enumerieren, 591

PLC-Variablenliste importieren, 914

PLC-Variablenliste löschen, 590

PLC-Variablenlisten in Ordnern enumerieren, 587

Projekt öffnen, 111

Projekt schließen, 139

Projekt speichern, 137

Projekte, 111, 116, 125, 130, 135, 137, 139, 185, 236, 239, 252, 584, 585, 587, 588, 591

Systemordner ermitteln, 515

Systemordner für PLC-Variablen abfragen, 584

Systemunterordner enumerieren, 515

Textliste löschen, 274

Variablen aus einer PLC-Variablenliste

löschen, 592

Variablen aus einer Variablenliste löschen, 277

Variablen in eine PLC-Variablenliste importieren, 916

Variablen oder Konstante aus einer PLC-Variablenliste exportieren, 914

Variablen einer HMI-Variablenliste enumerieren, 276

Variablenliste löschen, 278

VB-Skript aus einem Ordner löschen, 279

Verbindung löschen, 275

Zeitstempel eines Bausteins abfragen, 520

Zyklus löschen, 273

G

Generieren

Quelle aus Anwenderdatentyp, 531

Quelle aus Baustein, 531

Geräte enumerieren, 236, 239

Geräteelemente enumerieren, 252

Globale Bibliothek

Zugreifen, 143, 147

Zugriff auf Spracheinstellungen, 145

Grundstruktur einer AML-Exportdatei, 921

Grundstruktur einer Exportdatei, 790, 925

H**Hardware**

Übersetzen, 135

Hierarchie von Hardware-Objekten des Objektmodells, 69

HMI-Variablen des Datentyps "UDT", 803

I**Import/Export**

XML-Datei bearbeiten, 778

Importieren

Anwenderdatentyp, 917

Baustein, 897

Einzelne Variable in eine PLC-Variablenliste, 916

PLC-Variablenlisten, 914

Importieren/Exportieren

Alle Bildvorlagen exportieren, 824

Alle Grafiken eines Projekts exportieren, 783

Anwendungsbereich, 776

Auch Standardwerte exportieren, 779

Ausgewählte Variable exportieren, 800

Bausteine mit Knowhow-Schutz exportieren, 855

Bausteine ohne Knowhow-Schutz exportieren, 898

Besondere Überlegungen für integrierte HMI-Variablen, 802

Bild aus einem Bilderordner exportieren, 818

Bild mit einer Bildbausteininstanz exportieren, 834

Bild mit einer Bildbausteininstanz importieren, 836

Bilder eines HMI-Geräts exportieren, 817

Bilder in ein HMI-Gerät importieren, 820

Bildvorlagen exportieren, 825

Bildvorlagen importieren, 827

Datenstruktur, 790, 925

Eine HMI-Variable in eine Variablenliste importieren, 801

Einschränkungen, 776

Erweiterte XML-Formate für den Export/Import von Textlisten, 808

Exporte auf geänderte Werte beschränken, 779

Exporteinstellungen, 778

Exportformat, 776

Exportierbare Bildobjekte, 813

Exportierbare Objekte, 773

Exportieren eines Slide-in-Bilds, 831

Exportieren mehrsprachiger Kommentare, 994, 1003, 1011, 1013

Exportieren von Pop-up-Bildern, 829

Exportumfang, 778

Grafiken, 782

Grafiken in ein Projekt importieren, 784

Grafikliste importieren, 810

Grafiklisten exportieren, 810

Grundlagen, 773

HMI, 794, 795, 796, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 810, 812, 813, 817, 818, 820, 823, 824, 825, 827, 829, 830, 831, 833, 834, 836, 912

HMI-Variablenlisten exportieren, 796

Importierbare Objekte, 773

Importieren eines Slide-in-Bilds, 833

Importieren mehrsprachiger Kommentare, 994, 1003, 1011, 1013

Importieren von Pop-up-Bildern, 830

Importverhalten mit Programmcodes einstellen, 780

Konfigurationsdaten exportieren, 778

Konfigurationsdaten importieren, 780

Nur geänderte Werte exportieren, 779

Objekte von AML, 921

Permanentfenster exportieren, 823

Permanentfenster importieren, 823

PLC, 855, 894, 898

PLC-Variablenliste exportieren, 912

Projektdaten, 783, 784

Round-Trip-Geräte und -Module, 963

Stabile AML-GUIDs, 963

Systembausteine exportieren, 894

Textlisten exportieren, 806

Textlisten importieren, 807

Variable aus einer Variablenliste exportieren, 800

Variablen exportieren, 1014

Variablen importieren, 1014

Variablenliste in einen Variablenordner importieren, 799

VB-Skripte exportieren, 803, 804

VB-Skripte importieren, 805

Verbindungen exportieren, 811

Verbindungen importieren, 812

Vorgehensweise zum Importieren, 781

Zyklen exportieren, 794

Zyklen importieren, 795

Installation

Authentifizierung für Zugriff prüfen, 31

Benutzer der Benutzergruppe hinzufügen, 31

- Standardschritte für den Zugriff auf das TIA Portal, 37
TIA Openness V13 Add-on-Paket, 30
Installation des Add-on-Pakets, 30
Instanzen
 Typversion ermitteln, 176
Integrierte HMI-Variablen, 802
- K**
- Konfiguration
 Ihre Openness-Anwendung und das TIA Portal befinden sich auf unterschiedlichen Computern, 49
Kopieren
 Inhalt einer Masterkopie in Projektordner, 173
 Masterkopie, 176
- L**
- Lesen
 Parameter eines Technologieobjekts, 553
 Uhrzeit der letzten Änderungen an einer PLC-Variablenliste, 590
Löschen
 Alle Bilder, 272
 Anwenderdatentyp, 536
 Baustein, 523
 Benutzerdefinierter Ordner für PLC-Variablenlisten, 587
 Bild, 271
 Bildvorlage, 272
 Einzelne Variable in einer PLC-Variablenliste, 592
 Einzelne Variablen einer Variablenliste, 277
 Grafikliste, 275
 Grafiksammlung, 135
 Gruppe für Baustein, 524
 PLC-Konstanten, 593
 PLC-Variablenliste aus einem Ordner löschen, 590
 Programmbaustein, 523
 Technologieobjekt, 548
 Textliste, 274
 Variablenliste, 278
 VB-Skript aus einem Ordner, 279
 Verbindung, 275
 Zyklus, 273
- M**
- Masterkopie
 Inhalt in Projektordner kopieren, 173
 Kopieren, 176
Masterkopien
 Löschen, 182
Mehrsprachige Texte enumerieren, 125, 130
- O**
- Objekte
 Exportierbare Objekte, 773
 Importierbare Objekte, 773
Objektmodell, 54
Öffnen
 Editor „Geräte & Netze“, 185
Öffnen eines Projekts, 111
Ordner
 Löschen, 182
- P**
- Parameter
 Easy Motion Control, 582
 PID-Regelung, 581
 S7-1500 Motion Control, 564
 Zählung, 582
Parameter eines Technologieobjekts
 Enumерieren, 551
 Finden, 552
 Lesen, 553
 Schreiben, 554
PLC
 Online-Verbindung herstellen, 498
 Online-Verbindung trennen, 498
 Status ermitteln, 437
 Vergleich mit tatsächlichem Status, 493
 Vergleichen, 493
Programmbaustein
 Löschen, 523
Programmgesteuertes Quittieren von Systemereignissen, 87
Programmierübersicht, 53
Projekt
 Gerätetyp abfragen, 185
 HMI-Targets abfragen, 185
 Öffnen, 111
 PLC-Targets abfragen, 185

- Schließen, 139
 - Speichern, 137
 - Projekt speichern, 137
 - Projektbibliothek
 - Auf Masterkopien zugreifen, 169
 - Zugreifen, 143, 147
- S**
- Schreiben
 - Parameter eines Technologieobjekts, 554
 - Siemens.Engineering, 45
 - Siemens.Engineering.Hmi, 45
 - Siemens.Engineering.Hmi.Communication, 45
 - Siemens.Engineering.Hmi.Cycle, 45
 - Siemens.Engineering.Hmi.Globalization, 45
 - Siemens.Engineering.Hmi.RuntimeScripting, 45
 - Siemens.Engineering.Hmi.Screen, 45
 - Siemens.Engineering.Hmi.Tag, 45
 - Siemens.Engineering.Hmi.TextGraphicList, 45
 - Siemens.Engineering.HW, 45
 - Siemens.Engineering.SW, 45
 - Software
 - Übersetzen, 135
 - Starten
 - Baustineditor, 537
 - Variableneditor, 583
 - Status (PLC)
 - Ermitteln, 437
 - Struktur der Exportdaten, 790, 921, 925
- T**
- Technologieobjekt, 543
 - Abfragen, 546
 - Datentypen, 545
 - Enumerieren, 550
 - Erstellen, 547
 - Finden, 550
 - Löschen, 548
 - Übersetzen, 548
 - Technologieobjektgruppe
 - Übersetzen, 549
 - TIA Portal Openness, 47
 - Benutzer der Benutzergruppe hinzufügen, 31
 - Einleitung, 47
 - Export/Import, 39
 - Funktionen, 53
 - Funktionsumfang, 47
 - Grundbegriffe von Aggregationen, 100
- Grundlegende Konzepte bei der Handhabung von Ausnahmen, 766
 - Grundlegende Konzepte der Überprüfung auf Objektgleichheit, 102
 - Grundlegende Konzepte von Zuordnungen, 100
 - Konfiguration, 49
 - Notwendige Kenntnisse des Benutzers, 29
 - Programmierübersicht, 53
 - Public API, 53
 - Standardschritte für den Zugriff auf das TIA Portal, 37
 - Typische Aufgaben, 38
 - Voraussetzungen, 29
 - Zugriff, 38
 - Zugriffsrechte, 31
 - Typen
 - Löschen, 182
- U**
- Übersetzen
 - Hardware, 135
 - Software, 135
 - Technologieobjekt, 548
 - Technologieobjektgruppe, 549
- V**
- Variableneditor
 - Starten, 583
 - Verbindung zum TIA Portal
 - Einrichtung, 79
 - Schließen, 88
 - Verbindung zum TIA Portal herstellen, 79
 - Verschalten
 - Analogantriebe mit der Hardware-Adresse, 558
 - Analogantriebe mit einem Datenbaustein, 560
 - Antriebe, 568
 - Geber, 573
 - Geber für Analogantriebe mit der Hardware-Adresse, 559
 - Geber für PROFIdrives mit der Hardware-Adresse, 557
 - Geber mit einem Datenbaustein, 561
 - Gleichlaufachse mit Leitwerten, 578
 - Messtaster, 576
 - Nocken, 575
 - Nockenspur, 575
 - PROFIdrives mit der Hardware-Adresse, 556
 - PROFIdrives mit einem Datenbaustein, 560
 - Telegramm 750, 571

X

XML-Datei
Bearbeiten, 778
Export, 779

Z

Zugreifen
Masterkopie in einer Projektbibliothek, 169