

Rexroth IndraLogic XLC IndraMotion MLC 13VRS Automation Interface

R911336355
Ausgabe 01

Anwendungsbeschreibung



Titel Rexroth IndraLogic XLC
IndraMotion MLC 13VRS
Automation Interface

Art der Dokumentation Anwendungsbeschreibung

Dokumentations-Type DOK-XLCMLC-AUT*INT*V13-AP01-DE-P

Interner Ablagevermerk RS-f4cbe24321ce9b5a0a6846a000fd6245-1-de-DE-13

Zweck der Dokumentation Diese Dokumentation beschreibt den scriptbasierten Zugriff auf IndraWorks-Projektdaten über die Schnittstelle des Automation Interfaces.

Änderungsverlauf

Ausgabe	Stand	Bemerkung
Ausgabe 01	04.2012	Erstausgabe für 13VRS

Schutzvermerk © Bosch Rexroth AG 2012

Alle Rechte bei Bosch Rexroth AG, auch für den Fall von Schutzrechtsanmeldungen. Jede Verfügungsbefugnis, wie Kopier- und Weitergaberecht, bei uns.

Verbindlichkeit Die angegebenen Daten dienen allein der Produktbeschreibung und sind nicht als zugesicherte Eigenschaften im Rechtssinne zu verstehen. Änderungen im Inhalt der Dokumentation und Liefermöglichkeiten der Produkte sind vorbehalten.

Herausgeber Bosch Rexroth AG

Bgm.-Dr.-Nebel-Str. 2 ■ D-97816 Lohr a. Main
Telefon +49 9352 18 0 ■ Fax +49 9352 18 8400
<http://www.boschrexroth.com/>

Systementwicklung Automation Motion Logic Control (WoP) SyMu

Hinweis Diese Dokumentation ist auf chlorfrei gebleichtem Papier gedruckt.

Inhaltsverzeichnis

	Seite
1 Zu dieser Dokumentation.....	3
1.1 Gültigkeit der Dokumentation.....	3
1.2 Erforderliche und ergänzende Dokumentationen.....	3
1.3 Darstellung von Informationen.....	8
1.3.1 Verwendete Symbole.....	8
1.3.2 Bezeichnungen und Abkürzungen.....	8
2 Wichtige Gebrauchshinweise.....	11
2.1 Bestimmungsgemäßer Gebrauch.....	11
2.1.1 Einführung.....	11
2.1.2 Einsatz- und Anwendungsbereiche.....	11
2.2 Nicht-bestimmungsgemäßer Gebrauch.....	12
3 Einleitung.....	13
3.1 Allgemein.....	13
3.2 Scripting.....	14
3.3 Verwendung von Automation-Scripts	14
4 Basis Objekte.....	17
4.1 Objekt Modell.....	17
4.2 Objekt - Application.....	17
4.3 Objekt - ProjectManager.....	19
4.4 Objekt - Project.....	21
4.5 Objekt - ProjectElement.....	24
4.6 Objekt - Device (abgeleitet von ProjectElement).....	27
4.7 Objekt - File (abgeleitet von ProjectElement).....	28
4.8 Objekt - Folder (abgeleitet von ProjectElement).....	28
4.9 Objekt - AiCollection.....	28
4.10 Objekt - AiList.....	29
5 Codebeispiele - Basis Objekte.....	31
5.1 Codebeispiel - Automation Interface initialisieren.....	31
5.2 Codebeispiel - IndraWorks Projekt anlegen.....	31
5.3 Codebeispiel - IndraWorks Projekt öffnen.....	32
5.4 Codebeispiel - Hinzufügen von Ordnern und Dateien.....	32
5.5 Codebeispiel - Durchsuchen von Projekten.....	33
5.6 Codebeispiel - Zugriff auf Geräte.....	35
5.7 Codebeispiel - Projekt archivieren.....	36
5.8 Codebeispiel - Projekt wiederherstellen.....	37
5.9 Codebeispiel - Projekt validieren.....	38

Inhaltsverzeichnis

	Seite
6 XLC/MLC-Objekte.....	41
6.1 Objekt Modell.....	41
6.2 Objekt - XLC (abgeleitet von Device).....	41
6.3 Hinzufügen einer XLC.....	48
6.4 Objekt - MLC (abgeleitet von Device).....	50
6.5 Hinzufügen einer MLC.....	57
6.6 Objekt - MultiDeviceTable.....	59
7 Codebeispiele - XLC/MLC-Objekte.....	61
7.1 Codebeispiel - Zugriff auf die XLC/MLC.....	61
7.2 Codebeispiel - Online- / Offline-Schalten.....	61
7.3 Codebeispiel - Erweitertes Online- / Offline-Schalten.....	62
7.4 Codebeispiel - Phase lesen / schreiben.....	63
7.5 Codebeispiel - Parameter importieren.....	64
7.6 Codebeispiel - Erweitertes Parameter importieren.....	65
7.7 Codebeispiel - Parameter lesen / schreiben.....	66
7.8 Codebeispiel - Firmware-Download.....	67
7.9 Codebeispiel - XLC/MLC hinzufügen.....	68
7.10 Codebeispiel - Multi-Device verwenden / nicht verwenden.....	69
7.11 Codebeispiel - Aktivieren einer XLC/MLC aus der Multi-Device Tabelle.....	70
7.12 Codebeispiel - Zugriff auf das Objekt Logic.....	71
8 IL-Objekte.....	73
8.1 Objekt – Logic.....	73
8.2 Objekt – Application.....	73
9 Service und Support.....	77
Index.....	79

1 Zu dieser Dokumentation

1.1 Gültigkeit der Dokumentation

Zielgruppe Diese Dokumentation richtet sich an Anwender, die den script-basierten Zugriff auf IndraWorks-Projektdaten über die Schnittstelle des Automation Interfaces nutzen wollen.

Diese Dokumentation unterstützt den Anwender in den Phasen

- Engineering und
- Inbetriebnahme

1.2 Erforderliche und ergänzende Dokumentationen

Dokumentationstitel mit Typenschlüssel und Materialnummer

IndraWorks		MLC	XLC
/36/	Rexroth IndraWorks 13VRS Software-Installation DOK-IWORKS-SOFTINS*V13-COxx-DE-P, R911336879 Die Dokumentation beschreibt den Installationsprozess von IndraWorks.	X	X
/5/	Rexroth IndraWorks 13VRS Engineering DOK-IWORKS-ENGINEE*V13-APxx-DE-P, R911336869 Diese Dokumentation beschreibt die Verwendung von IndraWorks, in das sich die Rexroth Engineering-Tools integrieren. Beschrieben werden das Arbeiten mit IndraWorks und die Bedienung der Oszilloskop-Funktion.	X	X
/20/	Rexroth IndraMotion MLC 13VRS Funktionsbeschreibung DOK-MLC***-FUNC****V13-APxx-DE-P, R911336294 Die Dokumentation dient der Beschreibung der Wizards, Kontextmenüs, Dialoge, Steuerungsinbetriebnahme, Gerätekonfiguration, Funktionalitäten der IndraMotion MLC.	X	
/20/	Rexroth IndraLogic XLC 13VRS Funktionsbeschreibung DOK-XLC***-FUNC****V13-APxx-DE-P, R911336351 Die Dokumentation dient der Beschreibung der Wizards, Kontextmenüs, Dialoge, Steuerungsinbetriebnahme, Gerätekonfiguration, Funktionalitäten der IndraLogic XLC.		X
/7/	Rexroth IndraWorks 13VRS CamBuilder DOK-IWORKS-CAMBUIL*V13-APxx-DE-P, R911336290 Diese Dokumentation dient zur Beschreibung der Grundlagen und der Bedienung des CamBuilder, des Werkzeugs zur Kurvenscheibenbearbeitung.	X	X
/37/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Automation Interface DOK-XLCMLC-AUT*INT*V13-APxx-DE-P, R911336355 Diese Dokumentation beschreibt den skriptbasierten Zugriff auf IndraWorks-Projektdaten über die Schnittstelle des Automation Interfaces.	X	X
/38/	Rexroth IndraWorks 12VRS FDT-Container DOK-IWORKS-FDT*CON*V12-APxx-DE-P, R911334397 Diese Dokumentation beschreibt die FDT-Container-Funktionalität von IndraWorks. Beschrieben wird das Aktivieren der Funktionalität im Projekt und das Arbeiten mit DTMs.	X	X

Zu dieser Dokumentation

IndraWorks		MLC	XLC
/29/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Projektkonvertierung DOK-XLCMLC-PROCONV*V13-APxx-DE-P, R911336365 Diese Dokumentation beschreibt die Projektkonvertierung von IndraLogic 04VRS und IndraMotion MLC04VRS auf IndraWorks Version 12 mit IndraLogic 2G. Es wird insbesondere auf die Änderungen im Bereich Motion und SPS eingegangen.	X	X
/28/	Rexroth IndraMotion MLC 13VRS Inbetriebnahme DOK-MLC***-STARTUP*V13-COxx-DE-P, R911336307 Das vorliegende Handbuch beschreibt die Inbetriebnahme- und Servicearbeitsschritte für das System IndraMotion MLC. Es beinhaltet Checklisten für häufig durchzuführende Arbeiten und eine ausführliche Beschreibung der Arbeitsschritte.	X	

xx entsprechender Ausgabestand
 Abb. 1-1: XLC/MLC-Dokumentationsübersicht

Motion		MLC	XLC
/23/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS PLCopen-Bibliotheken DOK-XLCMLC-FUNLIB**V13-LIxx-DE-P, R911336359 Diese Dokumentation beschreibt die Funktionsbausteine, Funktionen und Datentypen der Bibliotheken RIL_CommonTypes, ML_Base und ML_PLCopen für die IndraLogic XLC/IndraMotion MLC. Die Dokumentation enthält ferner die Fehlerreaktion der Bausteine.	X	X
/27/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Generic Application Template DOK-XLCMLC-TF*GAT**V13-APxx-DE-P, R911336369 Dieses Handbuch stellt dem IndraLogic SPS-Programmierer ein strukturiertes Template, welches zum Hinzufügen und Bearbeiten des SPS-Programmiercodes verwendet werden kann, zur Verfügung. Beschrieben werden das Template selbst, der Template Wizard und Beispielapplikationen.	X	X
/31/	Rexroth IndraMotion MLC 13VRS RCL-Programmieranleitung DOK-MLC***-RCL*PRO*V13-APxx-DE-P, R911336296 Das vorliegende Handbuch informiert über die Robot Control. Die Programmiersprache RCL (Robot Control Language) steht dabei im Vordergrund. Beschrieben werden die Programmstruktur, Variablen, Funktionen, Bewegungsanweisungen sowie die notwendigen Parameter des Systems.	X	
/21/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Parameter DOK-XLCMLC-PARAM***V13-RExx-DE-P, R911336363 Diese Dokumentation beschreibt die Parameter der Systeme XLC/MLC, sowie das Wechselspiel zwischen Parametrieren und Programmieren. Beschrieben werden die Achs-Parameter, die Steuerungs-Parameter, die Kinematik-Parameter, die Messtaster- und die Nockenschaltwerks-Parameter.	X	X
/10/	Rexroth IndraDrive Firmware für Antriebsregelgeräte MPH-, MPB-, MPD-, MPC-07 DOK-INDRV*-MP*-07VRS**-FKxx-DE-P, R911328669 Diese Dokumentation beschreibt alle funktionellen Eigenschaften der IndraDrive-Firmware in den Varianten MPH-07, MPB-07, MPD-07 und MPC-07.		
/11/	Rexroth IndraDrive MPx-17 Funktionen DOK-INDRV*-MP*-17VRS**-APxx-DE-P, R911331235 Diese Dokumentation beschreibt alle funktionellen Eigenschaften der IndraDrive-Firmware in den Varianten MPB-17, MPM-17, MPC-17 und MPE-17.		

xx entsprechender Ausgabestand
 Abb. 1-2: XLC/MLC-Dokumentationsübersicht

Zu dieser Dokumentation

Feldbusse		MLC	XLC
/39/	Rexroth IndraMotion MLC 11VRS PLCopen Feldbus DOK-IM*ML*-PLCFBUS*V11-APxx-DE-P, R911333895 Die Dokumentation beschreibt die Anlage von Feldbusantrieben innerhalb eines IndraWorks-Projekts, Funktionsbausteine, Funktionen und Datentypen der Bibliotheken RIL_CommonTypes.library (Auszug für Feldbusantriebe), RMB_PLCopenFieldBus.library, RIL_Utility.library (Auszug für Feldbusantriebe). Die Dokumentation enthält ferner die Fehlerreaktion der Bausteine.	X	X
/4/	Rexroth IndraWorks 13VRS Feldbusse DOK-IWORKS-FB*****V13-APxx-DE-P, R911336871 Diese Dokumentation dient der Beschreibung der unterstützten Feldbusse und ihrer Diagnose-Funktionsbausteine.	X	X
/63/	sercos Systemhandbuch für I/O-Geräte DOK-CONTRL-ILS3*****-APxx-DE-P, R911333511 Diese Dokumentation beschreibt die Projektierung, Parametrierung, Inbetriebnahme und Diagnose von I/O-Geräten mit sercos Schnittstelle.	X	X

xx entsprechender Ausgabestand
Abb. 1-3: XLC/MLC-Dokumentationsübersicht

HMI		MLC	XLC
/8/	Rexroth IndraWorks 13VRS HMI DOK-IWORKS-HMI*****V13-APxx-DE-P, R911336873 Diese Dokumentation beschreibt die Funktionen, die Konfiguration und die Bedienung der Bedienoberflächen IndraWorks HMI Engineering und IndraWorks HMI Operation.	X	X
/6/	Rexroth IndraWorks 13VRS WinStudio DOK-IWORKS-WINSTUD*V13-APxx-DE-P, R911336881 Diese Dokumentation beschreibt die Installation der Software, das Arbeiten mit WinStudio und das Erstellen und den Betrieb von Applikationen.	X	X
/50/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS HMI-Anbindung DOK-XLCMLC-HMI*****V13-APxx-DE-P, R911336361 Dieses Dokument beschreibt die von der IndraLogic XLC und IndraMotion MLC unterstützten Visualisierungssysteme sowie deren Anbindung.	X	X

xx entsprechender Ausgabestand
Abb. 1-4: XLC/MLC-Dokumentationsübersicht

SPS		MLC	XLC
/3/	Rexroth IndraWorks 13VRS IndraLogic 2G Programmieranleitung DOK-IWORKS-IL2GPRO*V13-APxx-DE-P, R911336875 Diese Dokumentation beschreibt das SPS-Programmierungswerkzeug IndraLogic 2G und seine Verwendung. Beschrieben werden die Basisverwendung, Erste Schritte, die Visualisierung, Menüpunkte und die Editoren.	X	X
/33/	Rexroth IndraWorks 13VRS Basis-Bibliotheken IndraLogic 2G DOK-IL*2G*-BASLIB**V13-LIxx-DE-P, R911336284 Das vorliegende Handbuch beschreibt die systemübergreifenden SPS-Bibliotheken.	X	X

xx entsprechender Ausgabestand
Abb. 1-5: XLC/MLC-Dokumentationsübersicht

Zu dieser Dokumentation

Technologie		MLC	XLC
/30/	Rexroth IndraMotion MLC 13VRS Technologie-Bibliotheken DOK-MLC***-TF*LIB**V13-LIxx-DE-P, R911336323 Diese Dokumentation beschreibt die Funktionsbausteine, Funktionen und Datentypen der Bibliotheken "ML_TechInterface.library", "ML_TechMotion.library", "RMB_TechCam.library" und "ML_TechBase.library" sowie Bibliotheken zur Wicklerfunktionalität, Registerreglerfunktionalität und Querschneiderfunktionalität.	X	
/60/	Rexroth IndraMotion MLC 13VRS RegisterControl (Bibliothek) DOK-MLC***-REGI*CO*V13-LIxx-DE-P, R911336305 Diese Dokumentation beschreibt die Ein- und Ausgänge der einzelnen Funktionsbausteine und gibt Hinweise für deren Verwendung.	X	
/62/	Rexroth IndraMotion MLC 13VRS RegisterControl (Anwendungsbeschreibung) DOK-MLC***-REGI*CO*V13-APxx-DE-P, R911336303 Dieses Dokument beschreibt die Anwendung der integrierten Registerregelung für eine Tiefdruckmaschine. Beschrieben werden die Komponenten des Markenstromsensors, die HMI-Anwendung und die Fehlerbehebung. Diese Anleitung dient dazu, die Registerregelung zu bedienen und auf Fehler zu reagieren sowie Diagnosen abzufragen. Die Zielgruppe für diese Dokumentation sind Maschineneinrichter und Maschinenbediener.	X	
/49/	Rexroth IndraMotion MLC 13VRS Anwendung Wicklerfunktionen DOK-MLC***-TF*WIND*V13-APxx-DE-P, R911336325 Diese anwendungsbezogene Systemdokumentation beschreibt die Anwendung der Wickler-Technologiefunktionen.	X	

xx entsprechender Ausgabestand
 Abb. 1-6: XLC/MLC-Dokumentationsübersicht

Hardware		MLC	XLC
/1/	Rexroth IndraControl L45/L65/L85 Steuerung DOK-CONTRL-ICL45L65L85-PRxx-DE-P, R911332115 Diese Dokumentation dient der Beschreibung der Steuerungen IndraControl L45/L65/L85.	X	X
/2/	Rexroth IndraControl L25 DOK-CONTRL-IC*L25*****-PRxx-DE-P, R911328473 Diese Dokumentation dient der Beschreibung der Steuerungen IndraControl L25.	X	X
/24/	Rexroth IndraControl Lxx 13VRS Funktionsmodule DOK-CONTRL-FM*LXX**V13-APxx-DE-P, R911336282 Diese Dokumentation dient der Beschreibung aller Funktionsmodule der Lxx Steuerungen inkl. Engineering und Diagnose.	X	X
/12/	Rexroth IndraDrive Antriebsregelgeräte MPx-02 bis MPx-07 DOK-INDRV*-GEN-**VRS**-PAxx-DE-P, R911297316 Diese Dokumentation enthält die Beschreibung aller, in der Firmware für Antriebsregelgeräte der IndraDrive-Familie implementierten Parameter. Sie unterstützt die Parametrierung der Antriebsregelgeräte.		

Zu dieser Dokumentation

Hardware		MLC	XLC
/13/	Rexroth IndraDrive MPx-02 bis MPx-07 und HMV DOK-INDRV*-GEN-**VRS**-WAxx-DE-P, R911297318 Diese Dokumentation enthält die Beschreibung aller in den folgenden Firmwares implementierten Diagnosen: - Antriebsregelgerät-Firmwares MPx-02 bis MPx-08 und - Firmwares der Versorgungsgeräte des Typs "HMV". Sie unterstützt den Maschinenbediener und den Anlagenprogrammierer bei der Fehlerbehebung.		
/35/	Rexroth IndraDrive Antriebsregelgeräte Steuerteile CSB01, CSH01, CDB01 DOK-INDRV*-CSH*****-PRxx-DE-P, R911295011 Diese Dokumentation ist eine Projektierungsbeschreibung zur Auswahl und Verwendung der Steuerteile CSB01, CDB01 und CSH01 für Antriebsregelgeräte der Produktfamilien Rexroth IndraDrive M und Rexroth IndraDrive C.		

xx entsprechender Ausgabestand
Abb. 1-7: XLC/MLC-Dokumentationsübersicht

Diagnose und Service		MLC	XLC
/26/	Rexroth IndraMotion MLC/MLP/XLC 11VRS IndraMotion Service Tool DOK-IM*ML*-IMST***V11-RExx-DE-P, R911331939 Die Dokumentation beschreibt das IndraMotion Service Tool (IMST). Es ist ein webbasiertes Diagnosewerkzeug, das zum Zugriff auf die Steuerungssysteme IndraMotion MLC L25, L45 oder L65 sowie MLP über eine Ethernet-Hochgeschwindigkeitsverbindung dient. Das IMST ermöglicht OEMs, Endanwendern und Kundenbetreuern von überall auf ein System zuzugreifen und Ferndiagnosen zu stellen. Voraussetzung ist ein PC mit der Internet Explorer Version 6 oder 7.	X	X
/22/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Diagnosen DOK-XLCMLC-DIAG***V13-RExx-DE-P, R911336357 Diese Dokumentation enthält alle in den Steuerungssystemen IndraLogic XLC und IndraMotion MLC implementierten Steuerungsparameter.	X	X

xx entsprechender Ausgabestand
Abb. 1-8: XLC/MLC-Dokumentationsübersicht

Systemübersicht		MLC	XLC
/48/	Rexroth IndraMotion for Printing 13VRS Systemübersicht DOK-IM*PR*-SYSTEM**V13-PRxx-DE-P, R911336288 Das vorliegende Handbuch beschreibt das Produkt IndraMotion for Printing. Es stellt die Steuerungs-, Antriebs- und IO-Systeme vor und gibt eine Einführung in die Inbetriebnahme und Programmierung.	X	
/48/	Rexroth IndraMotion for Packaging 13VRS Systemübersicht DOK-IM*PA*-SYSTEM**V13-PRxx-DE-P, R911336286 Das vorliegende Handbuch beschreibt das Produkt IndraMotion for Packaging. Es stellt die Steuerungs-, Antriebs- und IO-Systeme vor und gibt eine Einführung in die Inbetriebnahme und Programmierung.	X	

Zu dieser Dokumentation

Systemübersicht		MLC	XLC
/9/	Rexroth IndraMotion MLC 13VRS Systemübersicht DOK-MLC***-SYSTEM**V13-PRxx-DE-P, R911336309 Diese Dokumentation stellt eine Übersicht der möglichen Hardware-/Softwarekomponenten des Automationssystems IndraMotion MLC in der genannten Version dar. Die Dokumentation dient als Auswahlhilfe zur Zusammenstellung eines Systems.	X	
/9/	Rexroth IndraLogic XLC 13VRS Systemübersicht DOK-XLC***-SYSTEM**V13-PRxx-DE-P, R911336353 Diese Dokumentation stellt eine Übersicht der möglichen Hardware-/Softwarekomponenten des Automationssystems IndraLogic XLC in der genannten Version dar. Die Dokumentation dient als Auswahlhilfe zur Zusammenstellung eines Systems.		X

xx entsprechender Ausgabestand
Abb. 1-9: XLC/MLC-Dokumentationsübersicht

Erste Schritte		MLC	XLC
/25/	Rexroth IndraMotion MLC 13VRS Die ersten Schritte DOK-MLC***-F*STEP**V13-COxx-DE-P, R911336292 Diese Dokumentation beschreibt die "ersten Schritte" mit der IndraMotion MLC und RobotControl. Beschrieben werden die Hard- und Softwarevoraussetzungen und das Erstellen eines Projekts.	X	
/25/	Rexroth IndraLogic XLC 13VRS Die ersten Schritte DOK-XLC***-F*STEP**V13-COxx-DE-P, R911336349 Diese Dokumentation beschreibt die "ersten Schritte" mit der IndraLogic XLC. Beschrieben werden die Hard- und Softwarevoraussetzungen und das Erstellen eines Projekts.		X

xx entsprechender Ausgabestand
Abb. 1-10: XLC/MLC-Dokumentationsübersicht

1.3 Darstellung von Informationen

1.3.1 Verwendete Symbole

Hinweis Hinweise werden wie folgt dargestellt:



Das ist ein Hinweis für den Anwender.

Tipp Tipps werden wie folgt dargestellt:



Das ist ein Tipp für den Anwender.

1.3.2 Bezeichnungen und Abkürzungen

Begriff	Bedeutung
AI	Automation Interface
HMI	Mensch Maschine Interface
HTML	Hypertext-Auszeichnungssprache
IDE	Integrierte Entwicklungsumgebung
MLC	Motion logic control

Zu dieser Dokumentation

OLE	Objekt-Verknüpfung und -Einbettung
VB	Visual Basic

Abb. 1-11: Verwendete Bezeichnungen und Abkürzungen

2 Wichtige Gebrauchshinweise

2.1 Bestimmungsgemäßer Gebrauch

2.1.1 Einführung

Die Produkte von Bosch Rexroth werden nach dem jeweiligen Stand der Technik entwickelt und gefertigt. Vor ihrer Auslieferung werden sie auf ihren betriebssicheren Zustand hin überprüft.

Die Produkte dürfen nur bestimmungsgemäß eingesetzt werden. Wenn sie nicht bestimmungsgemäß eingesetzt werden, dann können Situationen entstehen, die Sach- und Personenbeschädigung nach sich ziehen.



Für Schäden bei nicht-bestimmungsgemäßigem Gebrauch der Produkte leistet Bosch Rexroth als Hersteller keinerlei Gewährleistung, Haftung oder Schadensersatz; die Risiken bei nicht-bestimmungsgemäßigem Gebrauch der Produkte liegen allein beim Anwender.

Bevor Sie die Produkte der Firma Bosch Rexroth einsetzen, müssen die folgenden Voraussetzungen erfüllt sein, um einen bestimmungsgemäßen Gebrauch der Produkte zu gewährleisten:

- Jeder, der in irgendeiner Weise mit einem unserer Produkte umgeht, muss die entsprechenden Sicherheitsvorschriften und den bestimmungsgemäßen Gebrauch lesen und verstehen.
- Sofern es sich bei den Produkten um Hardware handelt, müssen sie in ihrem Originalzustand belassen werden; d. h. es dürfen keine baulichen Veränderungen an ihnen vorgenommen werden. Softwareprodukte dürfen nicht dekompiert werden und ihre Quellcodes dürfen nicht verändert werden.
- Beschädigte oder fehlerhafte Produkte dürfen nicht eingebaut oder in Betrieb genommen werden.
- Es muss gewährleistet sein, dass die Produkte entsprechend den in der Dokumentation genannten Vorschriften installiert sind.

2.1.2 Einsatz- und Anwendungsbereiche

Rexroth IndraMotion MLC

Die Rexroth IndraMotion MLC und ihre Funktionsmodule ist für Motion-/Logic-Anwendungen geeignet.



Die Rexroth IndraMotion MLC und ihre Funktionsmodule dürfen nur mit den in dieser Dokumentation angegebenen Zubehör- und Anbauteilen benutzt werden. Nicht ausdrücklich genannte Komponenten dürfen weder angebaut noch angeschlossen werden. Gleiches gilt für Kabel und Leitungen.

Der Betrieb darf nur in den ausdrücklich angegebenen Konfigurationen und Kombinationen der Hardware- Komponenten und mit der in den jeweiligen Dokumentationen und den Funktionsbeschreibungen angegebenen und spezifizierten Soft- und Firmware erfolgen.

Die Rexroth IndraMotion MLC und ihre Funktionsmodule sind für den Einsatz in ein- und mehrachsigen Antriebs- und Steuerungsaufgaben entwickelt worden.

Wichtige Gebrauchshinweise

Für den applikationsspezifischen Einsatz des Maschinenbedien- und Visualisierungsterminals stehen Gerätetypen mit unterschiedlicher Ausstattung und unterschiedlichen Schnittstellen zur Verfügung.

Typische Anwendungsbereiche der Rexroth IndraMotion MLC und ihre Funktionsmodule sind:

- Handhabungs- und Montagesysteme
- Verpackungs- und Lebensmittelmaschinen
- Druck- und Papierverarbeitungsanlagen
- Werkzeugmaschinen

Rexroth IndraMotion MLC und ihre Funktionsmodule dürfen nur unter den in den weiterführenden Dokumentationen angegebenen Montage- und Installationsbedingungen, in der angegebenen Gebrauchslage und unter den angegebenen Umweltbedingungen (Temperatur, Schutzart, Feuchte, EMV u. a.) betrieben werden.

2.2 Nicht-bestimmungsgemäßer Gebrauch

Die Verwendung der Rexroth IndraMotion MLC und ihre Funktionsmodule außerhalb der vorgenannten Anwendungsgebiete oder unter anderen als den in der Dokumentation beschriebenen Betriebsbedingungen und angegebenen technischen Daten gilt als "nicht-bestimmungsgemäß".

Die Rexroth IndraMotion MLC und ihre Funktionsmodule darf nicht eingesetzt werden, wenn sie

- Betriebsbedingungen ausgesetzt wird, die die vorgeschriebenen Umgebungsbedingungen nicht erfüllen. Untersagt sind z. B. der Betrieb unter Wasser, unter extremen Temperaturschwankungen oder extremen Maximaltemperaturen.
- Außerdem darf die IndraMotion MLC und ihre Funktionsmodule nicht bei Anwendungen eingesetzt werden, die von Bosch Rexroth nicht ausdrücklich freigegeben sind. Beachten Sie hierzu bitte unbedingt die Aussagen in den allgemeinen Sicherheitshinweisen!

3 Einleitung

3.1 Allgemein

Mit dem Automation Interface wird eine einfache, klar strukturierte Script-Schnittstelle für IndraWorks zur Verfügung gestellt, die den Zugriff auf IndraWorks Projektdaten ermöglicht.

IndraWorks Automatisierungs-Skripte (im folgenden als AI-Skripte bezeichnet) können in Script-Sprachen, wie beispielsweise VBScript oder JavaScript, erstellt und anschließend im IndraWorks Engineering Desktop oder in der Windows Konsole ausgeführt werden. Bei der Ausführung in IndraWorks ist es möglich, AI-Skripte in das Hauptmenü zu integrieren bzw. diese als separaten Knoten in ein IndraWorks Projekt einzufügen.

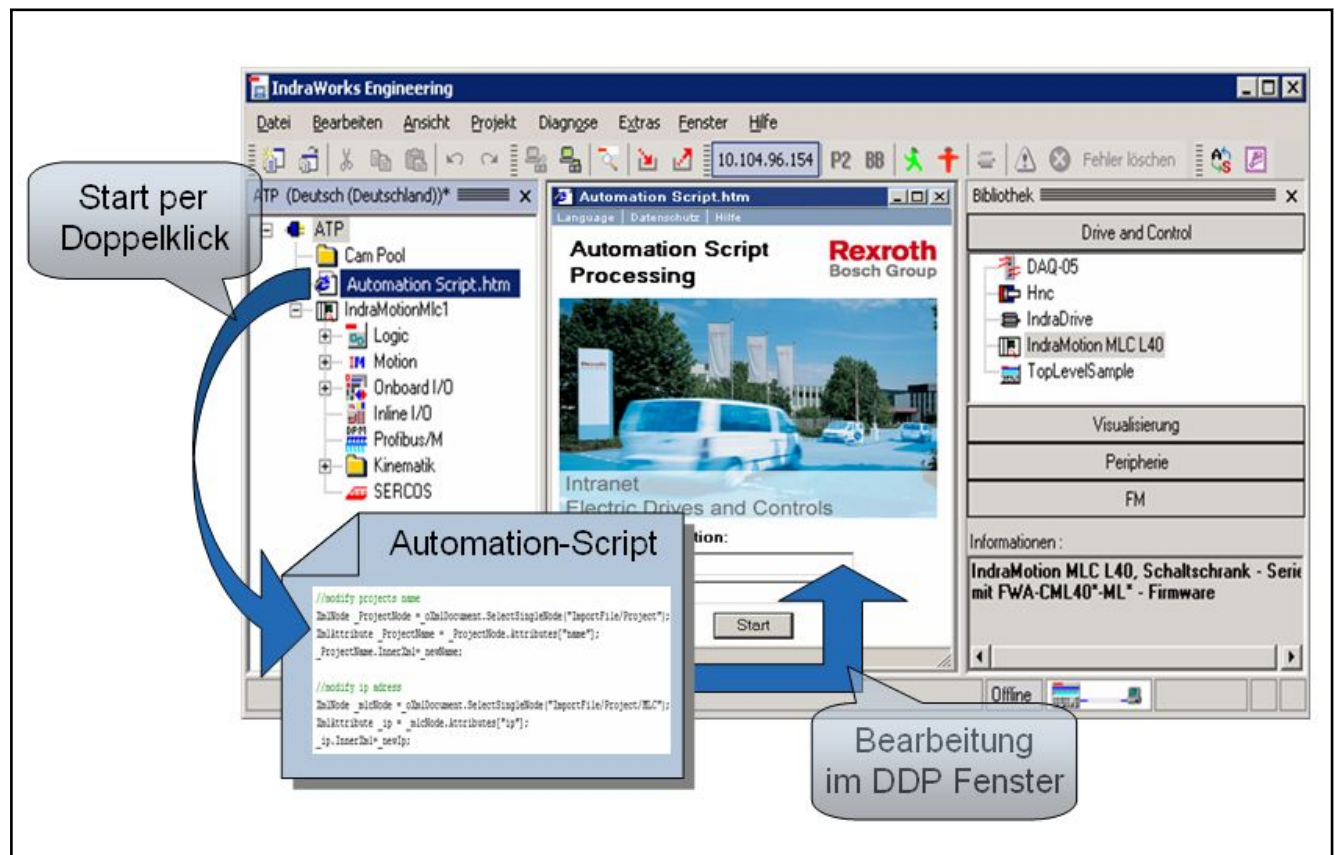


Abb.3-1: Automatisierungs-Skripte in IndraWorks

Im folgenden findet sich eine Aufzählung typischer Anwendungsfälle, für die sich eine Abwicklung über das Automation Interface anbietet:

- Erstellen von kundenspezifischen Reports.
- Automatisierung von regelmäßig wiederkehrenden Modifikationen an IndraWorks Projekten.
- Automatisierte Erzeugung von IndraWorks Projekten über die script-basierte Abarbeitung von Stücklisten.
- Erstellen von einfachen kundenspezifischen GUIs im IndraWorks Engineering-Desktop.

Einleitung



Für die in späteren Kapiteln aufgezeigten Code-Beispiele sind teilweise Voraussetzungen notwendig, die nicht immer automatisch bei der Installation erfüllt werden. Dadurch kann es sein, dass ein Script nicht in jedem Fall sofort lauffähig ist, wenn z.B. die Sicherheitseinstellungen des Browsers die Erstellung von ActiveX Objekten verhindern.

3.2 Scripting

IndraWorks Automation stellt eine Möglichkeit für andere Applikationen zur Verfügung, um IndraWorks Projektdaten über Windows OLE Automation (ActiveX) zu nutzen. Typischerweise wird das Automation Interface dabei über Script Clients, wie z.B. JavaScript oder VBScript genutzt.

Scripte können entweder über den Microsoft Scripting Host (Kommandozeilenversion: CScript.exe) oder integriert im IndraWorks Engineering-Desktop ausgeführt werden. Dazu muss eine HTML-Seite erzeugt werden, in die das Automation-Script eingebettet wird (ein Beispiel hierzu findet sich im folgenden Kapitel). Die HTML-Seite kann anschließend als eigener Menüpunkt im IndraWorks Hauptmenü (Hauptmenü **Extras ▶ Externe Anwendungen...**) bzw. als separater Knoten zum IndraWorks Projekt hinzugefügt und von dort aus gestartet werden.

Im Folgenden finden sich einige wichtige Anmerkungen zum Scripting:

- Scripte sind typenlos, d.h. im Script-Raum gibt es nur Objekte, aber keine speziellen Typen (Klassen).
- Typauflösung über LateBinding, d.h. es wird erst zur Laufzeit geprüft, ob der angesprochene Typ/Methode aufgelöst werden kann.
- Bei der Erstellung von AI-Scripten muss auf die vorliegende Dokumentation des Automation Interfaces zurückgegriffen werden; da Scripte typenlos sind, existiert keine IDE IntelliSense Hilfe.
- Beim Scripting sind in einem abgeleiteten Objekt alle Methoden und Attribute der Basisklassen verfügbar, d.h. es kann z.B. im Objekt **MLC** auf Methoden und Attribute von **Device** und **ProjectElement** zugegriffen werden.

3.3 Verwendung von Automation-Scripts

Ausführung von in HTML-Dateien eingebetteten Scripten

Scripte können auch eingebettet in einer HTML-Datei verwendet werden. Das Script wird automatisch ausgeführt, wenn die HTML-Seite in einem Browser geöffnet wird.

Beispiel:

Nachfolgendes Beispiel zeigt den Aufbau einer HTML-Seite mit eingebettetem JavaScript.

Programm:

```
<html>
  <body>
    <script language = "JavaScript">
      alert("Hello World...");
    </script>
  </body>
</html>
```

Ausführung von Script-Dateien in einer Konsole mit "CScript.exe"

Scripte können in einem Konsolenfenster unter Verwendung der in Microsoft Windows enthaltenen Applikation "CScript.exe" ausgeführt werden. Ausgaben des Scripts werden direkt in das Konsolenfenster geschrieben.

Beispiel:

Nach dem Öffnen einer Konsole (Eingabeaufforderung) kann nach Eingabe von "CScript.exe" die JavaScript-Datei "MyScript.js" wie folgt ausgeführt werden:

```
CScript.exe MyScript.js
```

Eine zum Entwickeln von Scripts hilfreiche Option ist "//D".

```
Mit CScript.exe MyScript.js //D
```

kann ein Script debuggt werden. Hierzu muss ein Debugger wie z.B. Microsoft "Visual Studio" auf dem das Script ausführenden Rechner installiert sein. Als Breakpoint innerhalb eines Scripts ist das Schlüsselwort "debugger" zu verwenden. Sobald der Program-Counter bei der Abarbeitung des Scripts auf das Schlüsselwort "debugger" stößt, wird die weitere Abarbeitung des Scripts unterbrochen und es kann der zu verwendende Debugger gestartet werden, in dem das Script anschließend weiter ausgeführt wird.

Weitere Optionen von "CScript.exe" lassen sich durch Aufruf von
CScript.exe

(ohne Übergabeargumente) auflisten.

4 Basis Objekte

4.1 Objekt Modell

Das folgende Diagramm zeigt die Objekthierarchie des Automation Interfaces. Das Diagramm entspricht hinsichtlich der Struktur einem IndraWorks Projekt.

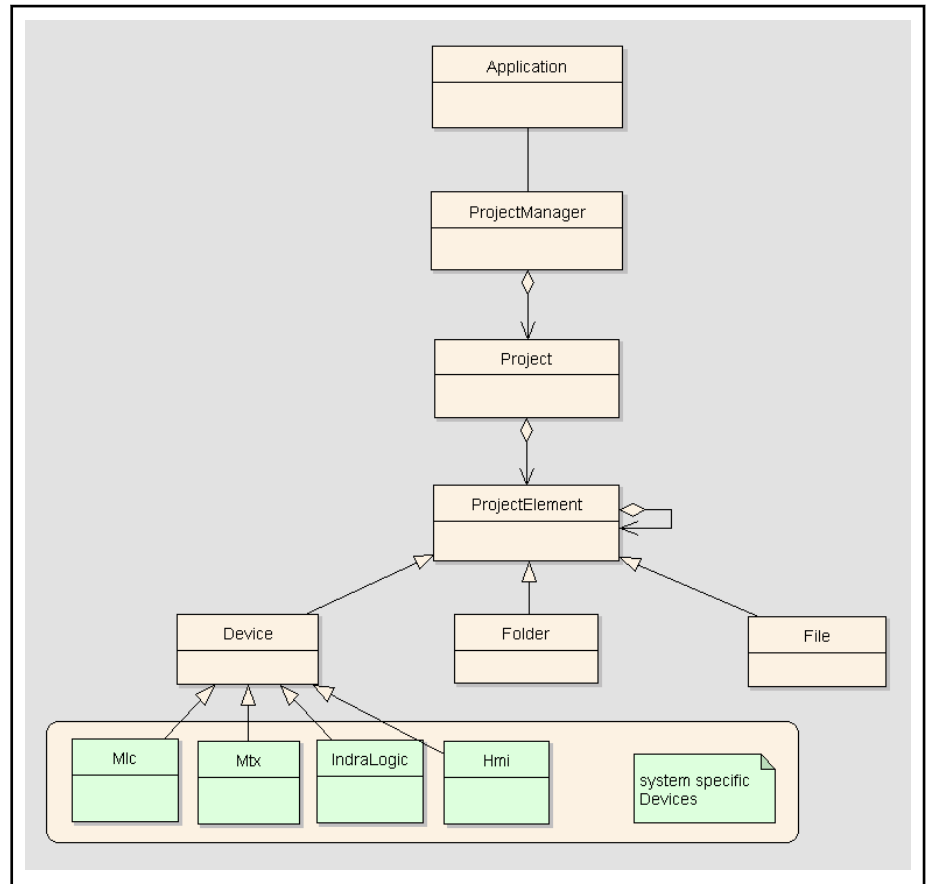


Abb.4-1: Objekt Modell des Automation Interface für Zugriff vom AI-Script

Bei den im Diagramm beige dargestellten Objekten handelt es sich um das **Basis Automation Interface**, das in dieser Dokumentation beschrieben wird. Die grün dargestellten Objekte sind systemspezifische Geräte wie beispielsweise HMI, MTX, MLC oder IndraLogic, die das Basis Automation Interface um systemspezifische Aspekte erweitern. Die Beschreibung dieser Objekte, sofern sie für ein System verfügbar sind, ist Bestandteil der entsprechenden Systemdokumentation.

4.2 Objekt - Application

Das Objekt **Application** ist das Startobjekt, über das die Initialisierung des Automation Interfaces durchgeführt wird. Es repräsentiert eine laufende Instanz des IndraWorks Engineering-Desktops. Beim Ausführen eines AI-Skriptes in IndraWorks wird das Objekt bereits von IndraWorks erzeugt und kann im Script von dem Property **window.external** gelesen werden. Nach der Initialisierung kann auf Daten von IndraWorks Projekten zugegriffen werden. Neben dem Zugriff über **window.external** kann das Objekt **Application** auch über den Namensraum **Rexroth.Indraworks.Automation.Application** initialisiert werden.

Basis Objekte

den. In diesem Fall wird beim Ausführen des Scriptes eine Sicherheitsabfrage des Internet Explorers aufgeschaltet.

Attribute von Application

Attribut	Typ	Beschreibung
ProjectManager	Rexroth.Indraworks.Automation.ProjectManager	Nur Lesen. Referenz auf den ProjectManager.

Methoden von Application

Methode	Typ	Beschreibung
Connect (string dest-ComputerName, string mode)	void	Initialisiert das Automation Interface und erstellt eine Verbindung zum Automation Server. Parameter: destComputerName [string in], Computernamen des Zielrechners, auf dem sich der Automation Server befindet, zu dem eine Verbindung aufgenommen werden soll. Aktuell werden nur lokale Verbindungen unterstützt (destComputerName = "localhost"). Parameter: mode [string in], dtm / automation / tci. Aktuell wird nur der mode = "automation" unterstützt.
GetService(string serviceName)	object	Über diese Methode wird der Zugriff auf, am AI registrierte Dienste bereitgestellt. Bei erfolgreicher Ausführung liefert die Methode eine Referenz auf die Instanz des angefragten Dienstes zurück. Im Fehlerfall wird 'null' zurückgeliefert. Die Dienstbeschreibung, d.h. die Schnittstellenbeschreibung des jeweiligen Dienstes wird vom Dienstanbieter (z.B. MTX, MLC, IndraLogic, Drive...) zur Verfügung gestellt. Parameter: serviceName [string in], Eindeutiger Name des am AI registrierten Services für den ein Zugriffsobjekt zurückgeliefert werden soll.
Disconnect()	void	Schließt die Verbindung zum Automation Server. Der Engineering- Desktop wird nicht beendet.

Basis Objekte

Methode	Typ	Beschreibung
Shutdown()	void	Beendet den IndraWorks Engineering-Desktop, zu dem eine Verbindung besteht. Dabei werden alle aktiven Vorgänge im Engineering-Desktop abgebrochen.
CreateComInstance (string progId)	object	Über diese Methode können COM Objekte erzeugt werden, die im Scripting selbst per new ActiveXObject(..), auf Grund der Security Einstellungen des Internet Explorers, nicht erzeugt werden können. Wenn z.B. im Automation Script ein Dateizugriff über das FileSystemObject benötigt wird, dann wird das FileSystemObject über den folgenden Aufruf im Script angelegt: <pre>var fso = new ActiveXObject("Scripting.FileSystemObject")</pre> Das ist jedoch nur möglich, wenn in den Browser Sicherheitseinstellungen die entsprechenden ActiveX-Berechtigungen gesetzt sind. Erlauben die Sicherheitseinstellungen es nicht, neue Objekte per new ActiveXObject(..) anzulegen, dann kann das benötigte FileSystemObject mit dem Aufruf von: <pre>var fso = Application.CreateComInstance("Scripting.FileSystemObject");</pre> über das Automation Interface erzeugt werden. Dadurch ist ein Dateizugriff möglich, auch wenn dieser normalerweise auf Grund der Sicherheitseinstellungen nicht erlaubt ist. Bei erfolgreicher Ausführung liefert die Methode eine Referenz auf die neue Instanz zurück. Im Fehlerfall wird eine COMException mit dem HRESULT Code: 0x800A01AD ("Can't create ActiveX object!") ausgelöst. Parameter: progId [string in], ProgId derjenigen registrierten COM Klasse, die über diesen Aufruf instanziiert werden soll. Beispiel: Die ProgId für das FileSystemObject ist "Scripting.FileSystemObject".

Beispielcode siehe: Kapitel [5.2 Codebeispiel - IndraWorks Projekt anlegen](#), Seite 31.

4.3 Objekt - ProjectManager

Über das Objekt **ProjectManager** können die Projekte verwaltet, d.h. neu angelegt, geöffnet, gelöscht, archiviert bzw. wiederhergestellt werden. Beim Neu Anlegen und Öffnen wird eine Referenz auf ein **Projekt**-Objekt geliefert.

Über das Attribut **Projects** kann eine Collection aller geöffneten Projekte gelesen werden.

Attribute von ProjectManager

Attribut	Typ	Beschreibung
Projects	Rexroth.IndraWorks.Automation.AiCollection	Nur Lesen. Liste aller geöffneten Projekte. Die Liste vom Typ AiCollection, enthält die Referenzen auf Project Objekte. In der aktuellen IndraWorks Implementierung kann nur ein einziges Projekt geöffnet werden.

Basis Objekte

Methoden von ProjectManager - Teil 1

Methode	Typ	Beschreibung
CreateProject(string fileName)	Rexroth.IndraWorks.Automation.Project	Erzeugt ein neues Projekt mit dem angegebenen Namen und liefert eine Referenz darauf zurück. Parameter: fileName [string in], Pfad und Name der Projektdatei (*.iwp Datei), beinhaltet Laufwerk und Directory-Bezeichnung.
OpenProject(string fileName)	Rexroth.IndraWorks.Automation.Project	Öffnet ein vorhandenes Projekt mit dem angegebenen Namen und liefert eine Referenz darauf zurück. Parameter: fileName [string in], Pfad und Name der Projektdatei (*.iwp Datei), beinhaltet Laufwerk und Directory-Bezeichnung.
DeleteProject(Rexroth.IndraWorks.Automation.Project iProject)	void	Löscht das angegebene Projekt.
ArchiveProject(Scripting.Dictionary archiveArgs)	void Exception im Fehlerfall.	Erstellt ein ZIP Archiv eines IndraWorks Projektes. Es werden nur diejenigen Daten in das Archiv aufgenommen, die sich im Projektverzeichnis befinden. Eventuell notwendige systemspezifische Archivierungsvorgänge müssen vorher durchgeführt worden sein. Es können nur geschlossene Projekte archiviert werden. Beim Archivieren eines geöffneten Projektes muss dieses geschlossen werden, dies kann über den Key 'CloseProject' (True/False) eingestellt werden. Parameter: archiveArgs [System.Dictionary, in], In einem Scripting.Dictionary Objekt werden alle Parameter für die Archivierung als Key/Value Paare übergeben. Für die Archivierung stehen die folgenden Keys (Parameter) zur Verfügung. Key: 'ProjectFile' [string in], Value: [string in] Projekt Datei (*.iwp) des zu archivierenden Projektes (inklusive Pfadname). Key: 'ArchiveFile' [string in], Value: [string in] Name der Archivdatei (*.zip), die bei der Archivierung erzeugt werden soll (inklusive Pfadinformation). Key: 'ArchiveComment' [string in], Value: [string in] [Optional Parameter] Kommentar zum Archiv. Key: 'Password' [string in], Value: [string in] [Optional Parameter] Passwort mit dem das Archiv geschützt werden soll. Wenn dieser Key nicht angegeben wird bzw. als Passwort ein Leerstring übergeben wird, dann wird das Archiv nicht geschützt. Key: 'CloseProject' [string in], Value: [string in, True or False] Über diesen Parameter können Sie festlegen, ob das Projekt, das Sie archivieren wollen, vor der Archivierung automatisch geschlossen werden soll. Der Parameter ist besonders dann sinnvoll, wenn eine zeitgesteuerte automatische Archivierung per AI-Script ausgeführt wird, während ein Benutzer das Projekt mit IndraWorks bearbeitet. Wenn Sie für diesen Fall CloseProject auf 'False' setzen, kann das Projekt vom AI nicht automatisch geschlossen werden, während es in der Oberfläche unter Bearbeitung ist. True: Projekt wird beim Archivieren geschlossen und danach wieder geöffnet. False: Projekt wird nicht geschlossen. Achtung: Ein geöffnetes Projekt kann nur archiviert werden, wenn dieser Parameter auf True gesetzt ist. Beispielcode siehe: Kapitel 5.7 Codebeispiel - Projekt archivieren , Seite 36 .

Methoden von ProjectManager - Teil 2

Methode	Typ	Beschreibung
RestoreProject(Scripting.Dictionary restoreArgs)	void Exception im Fehlerfall.	Stellt ein zuvor erstelltes IndraWorks Projektarchiv wieder her. Dabei wird das Archiv entpackt und das IndraWorks Projekt wiederhergestellt. Parameter: restoreArgs [System.Dictionary, in], In einem Scripting.Dictionary Objekt werden alle Parameter für die Wiederherstellung als Key/Value Paare übergeben. Für die Wiederherstellung stehen die folgenden Keys (Parameter) zur Verfügung. Key: 'ArchiveFile' [string in], Value: [string in] Datei des IndraWorks Projectarchives (*.zip) das wiederhergestellt werden soll (inklusive Pfadinformation). Key: 'RestoreDir' [string in], Value: [string in] Zielverzeichnis für die Wiederherstellung. Gibt an, in welchem Verzeichnis das IW Projektarchiv wiederhergestellt werden soll. Unterhalb dieses Verzeichnisses wird bei der Wiederherstellung das Projektverzeichnis angelegt. Key: 'NewProjectName' [string in], Value: [string in] [Optional Parameter] Wenn das wiederherzustellende Projekt einen neuen Namen bekommen soll, können Sie diesen hier angeben. In diesem Fall werden bei der Wiederherstellung, Projektverzeichnis, Projektdatei und Projektname entsprechend geändert. Key: 'Password' [string in], Value: [string in] [Optional Parameter] Passwort mit dem das Archiv geschützt ist. Wenn dieser Key nicht angegeben wird bzw. als Passwort ein Leerstring übergeben wird, dann wird versucht, das Archiv ohne Passwort zu restaurieren. Key: 'OpenProject' [string in], Value: [string in, True or False] Boolesche Variable, die festlegt, ob das wiederhergestellte Projekt nach der Wiederherstellung geöffnet werden soll. True: Projekt wird nach der Wiederherstellung automatisch geöffnet (das aktuell geöffnete Projekt wird zuvor geschlossen). False: Das Projekt wird nach der Wiederherstellung nicht geöffnet. Beispielcode siehe: Kapitel 5.8 Codebeispiel - Projekt wiederherstellen, Seite 37.

Beispielcode siehe: Kapitel 5.2 [Codebeispiel - IndraWorks Projekt anlegen](#), Seite 31.

4.4 Objekt - Project

Das Objekt **Project** repräsentiert ein IndraWorks-Projekt. Über die Methoden des Projektes können neue unterlagerte Projektelemente angelegt, Projektelemente gesucht oder gelöscht werden. Über **Export/Import** können die Projektdaten exportiert- bzw. importiert werden.

Sofern in dem Projekt systemspezifische Geräte wie beispielsweise MTX, MLC oder IndraLogic vorhanden sind, kann mit Hilfe der Methode **GetChildrenOfType** und des entsprechenden Typstrings eine Referenz auf diese Objekte ermittelt werden. Über diese Referenz kann auf die erweiterte Funktionalität der systemspezifischen Geräte zugegriffen werden.

Das Attribut **Children** liefert eine Collection aller Projektelemente des Projektes (alle Knoten der ersten Ebene unterhalb des Projektknotens).

Basis Objekte

Attribute von Project

Attribut	Typ	Beschreibung
Name	string	Lesen/Schreiben. Name des Projektes.
Id	string	Nur Lesen. Eindeutige ID (Guid) des Projektes.
Location	string	Nur Lesen. Verzeichnispfad, unter dem das Projekt (Projektdatei) gespeichert ist.
Children	Rexroth.IndraWorks.Automation.AiCollection	Nur Lesen. Liefert eine Collection aller Children (alle Knoten der ersten Ebene unterhalb des Projektknotens). Die Collection enthält Referenzen auf Objekte vom Typ Rexroth.IndraWorks.Automation.Device, Rexroth.IndraWorks.Automation.File, Typ Rexroth.IndraWorks.Automation.Folder oder aber Typen systemspezifischer Geräte.

Methoden von Project

Methode	Typ	Beschreibung
Save()	void	Speichert das Projekt.
SaveAs(string fileName)	Rexroth.IndraWorks.Automation.Project	Speichert das aktuelle Projekt unter neuem Namen und liefert eine Referenz auf das neue Projekt zurück. Achtung: Das alte Objekt Project (mit altem Namen) ist nach dem Aufruf ungültig! Parameter: fileName [string in], Name der neuen Projektdatei (*.iwp), beinhaltet Laufwerk und Directory-Bezeichnung.
Close()	void	Schließt das Projekt.
Export(string fileName)	void Exception im Fehlerfall.	Über diese Methode wird das Projekt exportiert. Parameter: fileName [string in], Dateiname (inklusive Pfadinformation) der Export Datei (*.iwx), die beim Export erzeugt wird.
Import(string fileName, string mode)	void Exception im Fehlerfall.	Über diese Methode werden die Daten eines Projektes, das zuvor per "Project.Export" erzeugt wurden, wieder importiert. Parameter: fileName [string in], Dateiname (inklusive Pfadinformation) der Export Datei (*.iwx oder *.xml für das alte Format), deren Daten importiert werden sollen. Parameter: mode [string in, mode = "Insert" or "Replace"], Importmodus, gibt an ob die zu importierenden Daten hinzugefügt werden sollen ("Insert") oder ob bereits vorhandene Daten ersetzt werden sollen ("Replace"). Der Parameter mode wird für das alte Exportformat (*.xml) nicht ausgewertet, in diesem Fall Parameter bitte auf "None" setzen.

Basis Objekte

Methode	Typ	Beschreibung
CreateChild(string type, object args)	Rexroth.IndraWorks.Automation.ProjectElement	Über diese Methode kann ein ProjectElement vom Typ "File" oder "Folder" oder ein systemspezifisches Gerät als Child des Projektknotens angelegt werden. Gehen Sie zum Anlegen eines neuen ProjectElements wie folgt beschrieben vor: Anlegen einer Datei: type = "File". args = Vollständiger Filename der Datei, die hinzugefügt werden soll, als String. Wird an dieser Stelle null oder Leerstring übergeben, dann öffnet sich ein Dateiauswahldialog im Engineering-Desktop, über den Sie die Datei auswählen können, die hinzugefügt werden soll. Anlegen eines Folders: type = "Folder" args = Name des neuen Ordners als String. Wird an dieser Stelle null oder Leerstring übergeben, dann wird der Name des neuen Ordners automatisch vergeben. Anlegen eines systemspezifischen Gerätes: type = <Geräte-Typbezeichner (string) des neu anzulegenden Gerätes> args = Prüfen Sie mit Hilfe der AI-Dokumentation des entsprechenden Gerätes, wie dieser Parameter zu belegen ist.
DeleteChild(string id)	bool	Löscht ein Child Projektelement mit der angegebenen ID (Der IndraWorks interne Bezeichner für diese ID ist "MnO Guid"). Parameter: id [string in], ID (Guid) des zu löschenden Child Projektelements.
GetChildrenOfType(string objectType, bool recurse)	Rexroth.IndraWorks.Automation.AiCollection	Liefert eine Collection aller Projektelemente des angegebenen Typs. Die Collection beinhaltet Referenzen der Typen Rexroth.IndraWorks.Automation.IDevice, Rexroth.IndraWorks.Automation.IFile, Typ Rexroth.IndraWorks.Automation.IFolder oder aber Typen systemspezifischer Geräte. Parameter: objectType [string in], Typstring der gesuchten Projektelemente. Als Grundtypen existieren "File", "Folder" und "Device". Sofern systemspezifische Geräte wie MTX, MLC, IndraLogic oder HMI existieren, können diese Geräte mit dem entsprechenden Typstring ermittelt werden. Parameter: recurse [bool in], True: Alle Projektelemente des Projektes werden bei der Suche berücksichtigt, False: Nur die Projektelemente, die sich auf der ersten Ebene unterhalb des Projektknotens befinden, werden für die Suche berücksichtigt. Beispielcode siehe: Kapitel 5.5 Codebeispiel - Durchsuchen von Projekten, Seite 33.

Basis Objekte

Methode	Typ	Beschreibung
GetElement(string id)	Rexroth.IndraWorks.Automation.ProjectElement	Liefert die Referenz auf das Projektelement mit der angegebenen ID (Der IndraWorks interne Bezeichner für diese ID ist "MnO Guid"). Parameter: id [string in], ID (Guid) des gesuchten Projektelements.
Validate(Scripting.Dictionary validateArgs)	string	Die Daten des Projekts werden einer Plausibilitätskontrolle unterzogen. Offensichtliche Unstimmigkeiten können hierdurch erkannt und beseitigt werden. Wichtig: Diese Überprüfung kann nicht die Richtigkeit aller Projektdaten verifizieren! Die Rückmeldung des Prüfungsergebnisses erfolgt als String im XML-Format. Die Nodes "Errors", "Warnings" und "Informations" liefern die Anzahl der jeweiligen Meldungen zurück. Wenn "Cancelled" gleich 'True', wurde die Prüfung abgebrochen. Unter "Messages" werden die gefundenen Meldungen ("Message") abgespeichert. Eine Meldung wird durch ihre Attribute beschrieben, sie entsprechen den Spalten des Meldfensters im IndraWorks-Engineering-Desktop: Type - Meldetyp "Error", "Warning", "Info" Id - Kennung der Meldung Text - Meldetext Element - Name des Projektelements Position - Optional: z.B. Zeilen- und Spaltenangabe wenn das Projektelement ein Quelltext Path - relativer Projektpfad Parameter: validateArgs [System.Dictionary, in], In einem Scripting.Dictionary Objekt können Key/Value-Paare als optionale Parameter übergeben werden. Eine leere Tabelle entspricht einer Validierungsfunktion, wie sie per Kontextmenüpunkt am Projektknoten gestartet werden kann. Sollen spezielle Funktionen geprüft werden, kann dies mit Hilfe von speziellen Key/Value-Parametern gesteuert werden. In diesem Fall sind die Namen der Schlüssel und die Bedeutung der zugehörigen Werte der Dokumentation (Systembeschreibung) der jeweiligen Projektelementen zu entnehmen. Beispielcode siehe: Kapitel 5.9 Codebeispiel - Projekt validieren, Seite 38.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <ValidationResults xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
- <Messages>
  <Message Type="Info" Id="PKG-00000007" Text="The element is on the 1. level of the project!" Element="Sample 1" Position="" Path="Sample 1" />
  <Message Type="Error" Id="PKG-00004711" Text="Element is deactivated!" Element="Sample 1" Position="" Path="Sample 1" />
  <Message Type="Info" Id="PKG-00071030" Text="The name "AAA" is not unique!" Element="AAA" Position="Ln 12345 Col 67" Path="Ordner/AAA" />
  <Message Type="Info" Id="PKG-00071030" Text="The name "AAA" is not unique!" Element="AAA" Position="" Path="Ordner/AAA" />
</Messages>
<Errors>1</Errors>
<Warnings>0</Warnings>
<Informations>3</Informations>
<Cancelled>False</Cancelled>
</ValidationResults>
```

Abb.4-2: Project.Validate(..): Rückgabeformat des Prüfungsergebnisses

4.5 Objekt - ProjectElement

ProjectElement ist eine Basisklasse des Automation Interfaces. Diese Basisklasse wird verwendet von den Objekten **Device**, **File** und **Folder** sowie von allen systemspezifischen Geräten wie beispielsweise MLC, MTX und IndraLogic. Das heißt, dass über diese Objekte auf alle Attribute der Basisklasse **ProjectElement** zugegriffen werden kann.

Die Basisklasse **ProjectElement** stellt eine Reihe von Attributen zur Verfügung, über die durch das Projekt navigiert werden kann.

Basis Objekte

Sofern unterhalb des aktuellen ProjektElementes systemspezifische Geräte wie beispielsweise MTX, MLC oder IndraLogic vorhanden sind, kann mit Hilfe der Methode **GetChildrenOfType** und des entsprechenden Typstrings eine Referenz auf diese Objekte ermittelt werden. Über diese Referenz kann auf die erweiterte Funktionalität der systemspezifischen Geräte zugegriffen werden.

Child ProjektElemente können über die Methoden **CreateChild** und **DeleteChild** erzeugt bzw. gelöscht werden.

Über die Methode **Export** können die Daten des ProjectElementes inklusive der Daten aller untergeordneten Elemente (sofern diese einen Export unterstützen) in eine Exportdatei exportiert werden.

Über die Methode **Import** können die Daten einer Exportdatei wieder in das ProjectElement bzw. dessen untergeordnete Elemente importiert werden.

Über das Attribut **SupportsExportImport** kann abgefragt werden, ob der Export Import von diesem ProjectElement unterstützt wird oder nicht.

Über das Attribut **Parent** kann der Eltern-Knoten ermittelt werden, **Children** liefert eine Collection aller Kind-Projektelemente der ersten Ebene und über **OwnerProject** kann das zugeordnete Projekt ermittelt werden.

Attribute von ProjectElement

Attribute	Typ	Beschreibung
Name	string	Lesen/Schreiben. Name (Anzeigename) des Projektelements.
Id	string	Nur Lesen. Eindeutige ID (Guid) des Projektelementes.
Location	string	Nur Lesen. Verzeichnispfad, unter dem das Projektelement gespeichert ist.
OwnerProject	Rexroth.IndraWorks.Automation.Project	Nur lesen. Referenz auf das Projekt, dem das Projektelement zugeordnet ist.
Parent	Rexroth.IndraWorks.Automation.ProjectElement	Nur Lesen. Referenz auf das Eltern-Projektelement. Wenn das Eltern Element der Projektknoten ist, wird null zurückgeliefert.
Children	Rexroth.IndraWorks.Automation.AiCollection	Nur Lesen. Liefert eine Collection mit allen Children der ersten Ebene. Die Collection beinhaltet Referenzen der Typen Rexroth.IndraWorks.Automation.IDevice, Rexroth.IndraWorks.Automation.IFile, Typ Rexroth.IndraWorks.Automation.IFolder oder aber Typen systemspezifischer Geräte.
SupportsExportImport	bool	Nur Lesen. Liefert die Information, ob das ProjectElement die Funktionalität Export und Import unterstützt. True: Export/Import wird unterstützt. False: Export/Import wird nicht unterstützt.
SupportsValidate	bool	Nur Lesen. Liefert die Information, ob das Projektelement die Funktionalität Validate unterstützt. True: Validate wird unterstützt. False: Validate wird nicht unterstützt.

Basis Objekte

Methoden von ProjectElement

Methoden	Typ	Beschreibung
Export(string fileName)	void Exception im Fehlerfall.	Über diese Methode werden die Daten des Projektelementes und aller untergeordneten Projektelemente, die einen Export unterstützen, exportiert. Achtung: Diese Methode wird nicht von jedem ProjectElement unterstützt! Fragen Sie vor der Verwendung dieser Methode über das Attribut "ProjectElement.SupportsExportImport" ab, ob die Methode unterstützt wird. Parameter: fileName [string in], Dateiname (inklusive Pfadinformation) der Export Datei (*.iwx), die beim Export erzeugt wird.
Import(string fileName, string mode)	void Exception im Fehlerfall.	Über diese Methode werden die Daten einer Exportdatei, die zuvor per "ProjectElement.Export" erzeugt wurden, wieder importiert. Achtung: Diese Methode wird nicht von jedem ProjectElement unterstützt! Fragen Sie vor der Verwendung dieser Methode über das Attribut "ProjectElement.SupportsExportImport" ab, ob die Methode unterstützt wird. Parameter: fileName [string in], Dateiname (inklusive Pfadinformation) der Export Datei (*.iwx), deren Daten importiert werden sollen. Parameter: mode [string in, mode = "Insert" or "Replace"], Importmodus, gibt an ob die zu importierenden Daten hinzugefügt werden sollen ("Insert") oder ob bereits vorhandene Daten ersetzt werden sollen ("Replace").
CreateChild(string type, object args)	Rexroth.IndraWorks.Automation.ProjectElement	Über diese Methode kann ein ProjectElement vom Typ "File" oder "Folder" oder ein systemspezifisches Gerät als Child des aktuellen ProjectElements angelegt werden. Gehen Sie zum Anlegen eines neuen ProjectElements wie folgt beschrieben vor: Anlegen einer Datei: type = "File". args = Vollständiger Filenamen der Datei, die hinzugefügt werden soll, als String. Wird an dieser Stelle null oder Leerstring übergeben, dann öffnet sich ein Dateiauswahldialog im Engineering-Desktop, über den Sie die Datei auswählen können, die hinzugefügt werden soll. Anlegen eines Folders: type = "Folder" args = Name des neuen Ordners als String. Wird an dieser Stelle null oder Leerstring übergeben, dann wird der Name des neuen Ordners automatisch vergeben. Anlegen eines systemspezifischen Gerätes: type = <Geräte-Typbezeichner (string) des neu anzulegenden Gerätes> args = Prüfen Sie mit Hilfe der AI-Dokumentation des entsprechenden Gerätes, wie dieser Parameter zu belegen ist.
DeleteChild(string id)	bool	Löscht ein Child Projektelement mit der angegebenen ID (Der IndraWorks interne Bezeichner für diese ID ist "MnO Guid"). Parameter: id [string in], ID (Guid) des zu löschenden Child Projektelements.

Basis Objekte

Methode	Typ	Beschreibung
GetChildrenOf- Type(string object- Type, bool recurse)	Rexroth.IndraWorks.A utomation.AiCollection	Liefert eine Collection aller Projektelemente des angegebenen Typs. Die Collection beinhaltet Referenzen der Typen Rexroth.IndraWorks.Automation.IDevice, Rexroth.IndraWorks.Automation.IFile, Typ Rexroth.IndraWorks.Automation.IFolder oder aber Typen systemspezifischer Geräte. Parameter: objectType [string in], Typstring der gesuchten Projektelemente. Als Grundtypen existieren "File", "Folder" und "Device". Sofern systemspezifische Geräte wie MTX, MLC, IndraLogic oder HMI existieren, können diese Geräte mit dem entsprechenden Typstring ermittelt werden. Parameter: recurse [bool in], True: Alle Projektelemente des Projektes werden bei der Suche berücksichtigt, False: Nur die Projektelemente, die sich auf der ersten Ebene unterhalb des Projektknotens befinden, werden für die Suche berücksichtigt.
Validate(Scripting.Dictionary validateArgs)	string	Die Daten des Projektelementes und aller unterlagerter Projektelemente werden einer Plausibilitätskontrolle unterzogen. Offensichtliche Unstimmigkeiten können hierdurch erkannt und beseitigt werden. Wichtig: Diese Überprüfung kann nicht die Richtigkeit aller Projektdaten verifizieren! Aufrufparameter und Rückgabewert: siehe Objekt Project, Methode Validate Achtung: Diese Methode wird nicht von jedem ProjectElement unterstützt! Fragen Sie vor der Verwendung dieser Methode über das Attribut "ProjectElement.SupportsValidate" ab, ob die Methode unterstützt wird.

4.6 Objekt - Device (abgeleitet von ProjectElement)

Das Objekt **Device** repräsentiert ein Projektelement des Typs **Device**, d.h. ein Gerät innerhalb eines IndraWorks Projektes.

Über die Methoden des Objektes kann das Gerät in den Modus online bzw. offline geschaltet werden. Über das Attribut IsOnline kann abgefragt werden, ob sich das Gerät aktuell im Online-Modus befindet.

Darüberhinaus stehen im Kontext von **Device** alle Attribute und Methoden der Basisklasse **ProjectElement** zur Verfügung.

Attribute von Device

Attribute	Typ	Beschreibung
IsOnline	bool	Nur Lesen. Liefert Information darüber, ob das Device sich im Mode Online oder Offline befindet. Liefert "true", wenn das Gerät online ist und "false", wenn es offline ist.

Basis Objekte

Methoden von Device

Methode	Typ	Beschreibung
GoOnline()	void	Schaltet das Gerät in den Modus Online. Bitte prüfen Sie anhand der gerätespezifischen AI-Dokumentation, ob diese Funktionalität für das Gerät zur Verfügung steht oder nicht.
GoOffline()	void	Schaltet das Gerät in den Modus Offline. Bitte prüfen Sie anhand der gerätespezifischen AI-Dokumentation, ob diese Funktionalität für das Gerät zur Verfügung steht oder nicht.

Beispielcode siehe: Kapitel [5.6 Codebeispiel - Zugriff auf Geräte](#), Seite 35.

4.7 Objekt - File (abgeleitet von ProjectElement)

Das Objekt **File** repräsentiert ein Projektelement des Typs **File**, d.h. eine Datei innerhalb eines IndraWorks Projektes.

Über das Attribut **FileName** kann der Name der Datei gelesen werden.

Darüberhinaus stehen im Kontext von **File** alle Attribute und Methoden der Basisklasse **ProjectElement** zur Verfügung.

Attribute von File

Attribute	Typ	Beschreibung
FileName	string	Nur Lesen. Name der Datei.

Beispielcode siehe: [5.4 Codebeispiel - Hinzufügen von Ordnern und Dateien](#), Seite 32.

4.8 Objekt - Folder (abgeleitet von ProjectElement)

Das Objekt **Folder** repräsentiert ein Projektelement des Typs **Folder**, d.h. ein Verzeichnis innerhalb eines IndraWorks Projektes.

Darüberhinaus stehen im Kontext von **File** alle Attribute und Methoden der Basisklasse **ProjectElement** zur Verfügung.

Beispielcode siehe: Kapitel [5.4 Codebeispiel - Hinzufügen von Ordnern und Dateien](#), Seite 32.

4.9 Objekt - AiCollection

Das Objekt **AiCollection** ist eine unsortierte Liste von Objekten, auf die entsprechend der nachfolgenden Attribute und Methoden lesend zugegriffen werden kann.

Attribute von AiCollection

Attribute	Typ	Beschreibung
Count	int	Nur Lesen. Liefert die Anzahl der Objekte.
Current	object	Nur Lesen. Liefert das aktuell angewählte Objekt.
this[int]	object	Nur Lesen. Indexer. Parameter: index [int in], Index (Basis 0) innerhalb der Collection.

Methoden von AiCollection

Methode	Typ	Beschreibung
GetEnumerator()	System.Collections.IEnumerator	Liefert einen Enumerator.
Contains(object item)	bool	Prüft, ob sich das angegebene Objekt in der Collection befindet. True: Das gesuchte Objekt befindet sich in der Collection. False: Das gesuchte Objekt ist nicht in der Collection.

Beispielcode siehe: Kapitel [5.5 Codebeispiel - Durchsuchen von Projekten](#), Seite 33.

4.10 Objekt - AiList

Das Objekt **AiList** ist eine unsortierte Liste von Objekten, auf die entsprechend der nachfolgenden Attribute und Methoden lesend zugegriffen werden kann. Im Unterschied zur AiCollection bietet das Objekt AiList die Möglichkeit, Objekte über die Methoden **Add()** und **Remove()** zur Liste hinzuzufügen bzw. diese aus der Liste auszutragen.

Attribute von AiList

Attribute	Typ	Beschreibung
Count	int	Nur Lesen. Liefert die Anzahl der Objekte.
Current	object	Nur Lesen. Liefert das aktuell angewählte Objekt.
this[int]	object	Nur Lesen. Indexer. Parameter: index [int in], Index (Basis 0) innerhalb der Collection.

Methoden von AiList

Methode	Typ	Beschreibung
Add(object value)	int	Fügt ein neues Objekt zu der Liste hinzu und liefert den Index zurück, an dem das Objekt eingefügt wurde.
Remove(object value)	void	Löscht das erste Objekte aus der Liste, das mit dem angegebenen Objekt identisch ist.
GetEnumerator()	System.Collections.IEnumerator	Liefert einen Enumerator.
Contains(object item)	bool	Prüft, ob sich das angegebene Objekt in der Collection befindet. True: das gesuchte Objekt befindet sich in der Collection. False: Das gesuchte Objekt ist nicht in der Collection.

5 Codebeispiele - Basis Objekte

5.1 Codebeispiel - Automation Interface initialisieren

AI initialisierung

Programm:

```
<html>
<body>
<script language = "JavaScript">
  try
  {
    //Get Application object of Automation Interface
    var Application = window.external;

    //Alternatively: Initialization of Application object,
    //this forces an Internet Explorer Security Dialog
    //var Application = new ActiveXObject
    //("Rexroth.IndraWorks.Automation.Application");

    //Initialize Automation Interface (Connect to Automation Server)
    Application.Connect("localhost", "automation");

    //
    //ToDo: Access Project data
    //

  }
  catch (e)
  {
    //Show Exception
    alert(e + ", " + e.message);

    //Disconnect from Automation Server
    Application.Disconnect();
  }
</Script>
</Body>
</html>
```

5.2 Codebeispiel - IndraWorks Projekt anlegen

AI initialisieren und Projekt anlegen

Programm:

```
<html>
<body>
<script language = "JavaScript">
  try
  {
    //Get Application object of Automation Interface
    var Application = window.external;

    //Alternatively: Initialization of Application object,
    //this forces an Internet Explorer Security Dialog
    //var Application = new ActiveXObject("Rexroth.IndraWorks.Automation.Application");

    //Initialize Automation Interface (Connect to Automation Server)
    Application.Connect("localhost", "automation");

    //Get Project Manager
    var ProjectManager = Application.ProjectManager;

    //Create Project
    var Project = ProjectManager.CreateProject
    ("C:\\AI-Test\\MyProject_001\\MyProject_001.iwp");

    //
    //ToDo: Access Project data
    //

  }
  catch (e)
```

Codebeispiele - Basis Objekte

```

{
  //Show Exception
  alert(e + ", " + e.message);

  //Disconnect from Automation Server
  Application.Disconnect();
}
</Script>
</Body>
</html>

```

5.3 Codebeispiel - IndraWorks Projekt öffnen

AI initialisieren und Projekt öffnen

Programm:

```

<html>
<body>
<script language = "JavaScript">
  try
  {
    //Get Application object of Automation Interface
    var Application = window.external;

    //Alternatively: Initialization of Application object,
    //this forces an Internet Explorer Security Dialog
    //var Application = new ActiveXObject
    //("Rexroth.IndraWorks.Automation.Application");

    //Initialize Automation Interface (Connect to Automation Server)
    Application.Connect("localhost", "automation");

    //Get Project Manager
    var ProjectManager = Application.ProjectManager;

    //Create Project
    var Project = ProjectManager.OpenProject
    ("C:\\\\AI-Test\\\\MyProject_001\\\\MyProject_001.iwp");

    //
    //ToDo: Access Project data
    //

  }
  catch (e)
  {
    //Show Exception
    alert(e + ", " + e.message);

    //Disconnect from Automation Server
    Application.Disconnect();
  }
</Script>
</Body>
</html>

```

5.4 Codebeispiel - Hinzufügen von Ordnern und Dateien

Erzeugen von Ordnern und Dateien in einem Projekt

Programm:

```

<html>
<body>
<script language = "JavaScript">
  try
  {
    //Get Application object of Automation Interface
    var Application = window.external;

    //Alternatively: Initialization of Application object, this forces an
    //Internet Explorer Security Dialog
    //var Application = new ActiveXObject("Rexroth.IndraWorks.Automation.Application");

    //Initialize Automation Interface (Connect to Automation Server)
    Application.Connect("localhost", "automation");
  }

```

```

//Get Project Access (use Example Code for creating "MyProject_001" before)
var CurrentProject;
var OpenProjects = Application.ProjectManager.Projects;
for(var prjElements = new Enumerator(OpenProjects);
!prjElements.atEnd(); prjElements.moveNext())
{
    CurrentProject = prjElements.item();
    break;
}
if(( CurrentProject == null) || ( CurrentProject.Name != "MyProject_001"))
{
    CurrentProject = Application.ProjectManager.OpenProject
    ("C:\\\\AI-Test\\\\MyProject_001\\\\MyProject_001.iwp");
}

//Create a new Folder with name "MyRootFolder" as child of Project node
var RootFolder = CurrentProject.CreateChild("Folder", "MyRootFolder");

//Create 20 further folders as Children of RootFolder and set names
var ChildFolder;
for( var i=1; i<21; i++)
{
    ChildFolder = RootFolder.CreateChild("Folder", "MyRootFolder_" + i);
}

//Create (Add) a File as a child of the Project node (FileOpen Dilaog will open)
CurrentProject.CreateChild("File", null);

//Create (Add) some Files as Child of CurrentProject (!!!no FileOpen Dialog!!!)
CurrentProject.CreateChild("File",
    ".\\ATLHost.dll");
CurrentProject.CreateChild("File",
    ".\\Indraworks.Automation.dll");
CurrentProject.CreateChild("File",
    ".\\Indraworks.AutomationServer.dll");
CurrentProject.CreateChild("File",
    ".\\Indraworks.AutomationServer.Interfaces.dll");
CurrentProject.CreateChild("File",
    ".\\Indraworks.FilePackage.dll");
CurrentProject.CreateChild("File",
    ".\\Indraworks.Formatter.dll");

//Create (Add) a File as a child of the RootFolder (FileOpen Dilaog will open)
RootFolder.CreateChild("File", null);

//Create (Add) some Files as Child of RootFolder (!!!no FileOpen Dialog!!!)
RootFolder.CreateChild("File", ".\\ATLHost.dll");
RootFolder.CreateChild("File", ".\\Indraworks.Automation.dll");
RootFolder.CreateChild("File", ".\\Indraworks.AutomationServer.dll");

//
//ToDo: Access Project data
//
}
catch (e)
{
    //Show Exception
    alert(e + ", " + e.message);

    //Disconnect from Automation Server
    Application.Disconnect();
}
</Script>
</Body>
</html>

```

5.5 Codebeispiel - Durchsuchen von Projekten

Suchen in einem Projekt

Programm:

```

<html>
<body>
<script language = "JavaScript">
    try
    {
        //Get Application object of Automation Interface

```

Codebeispiele - Basis Objekte

```

var Application = window.external;

//Alternatively: Initialization of Application object, this forces an
//Internet Explorer Security Dialog
//var Application = new ActiveXObject
//("Rexroth.IndraWorks.Automation.Application");

//Initialize Automation Interface
//Connect to Automation Server
Application.Connect("localhost", "automation");

//Get Project Access
//use Example Code for creating "MyProject_001" before)
var CurrentProject;
var OpenProjects = Application.ProjectManager.Projects;
for(var prjElements = new Enumerator
    (OpenProjects);
    !prjElements.atEnd(); prjElements.moveNext())
{
    CurrentProject = prjElements.item();
    break;
}
if(( CurrentProject == null) ||
    ( CurrentProject.Name != "MyProject_001"))
{
    CurrentProject = Application.ProjectManager.OpenProject
    ("C:\\\\AI-Test\\\\MyProject_001\\\\MyProject_001.iwp");
}

//
//Note: The Examples for creating files, folders and devices
//      should be executed before this example in order to have
//      some data available.
//

//-----
//Get children (first level) of current project
//-----
var Children = CurrentProject.Children;

//Show children
var childElements = new Enumerator(Children);
var child;
document.write("Children (first level) of Project:<br>");
document.write("=====<br>");
for(; !childElements.atEnd(); childElements.moveNext())
{
    child = childElements.item();
    document.write("Name: '" + child.Name +
        "' | Type: '" + child.ObjectType + "'<br>");
}

//-----
//Get all children of type "File", scan all nodes of project
//-----
Children = CurrentProject.GetChildrenOfType("File", true);

//Show children
childElements = new Enumerator(Children);
document.write("<br>All Files of Project:<br>");
document.write("=====<br>");
for(; !childElements.atEnd(); childElements.moveNext())
{
    child = childElements.item();
    document.write("Name: '" + child.Name +
        "' | Type: '" + child.ObjectType + "'<br>");
}

//-----
//Get all children of type "Folder", scan all nodes of project
//-----
Children = CurrentProject.GetChildrenOfType("Folder", true);

//Show children
childElements = new Enumerator(Children);
document.write("<br>All Folders of Project:<br>");
document.write("=====<br>");
for(; !childElements.atEnd(); childElements.moveNext())
{
    child = childElements.item();

```

```

        document.write("Name: '" + child.Name +
            "' | Type: '" + child.ObjectType + "'<br>");
    }

    //-----
    //Get children (first level) of type "Device"
    //-----
    Children = CurrentProject.GetChildrenOfType("Device", false);

    //Show children
    childElements = new Enumerator(Children);
    document.write("<br>Devices (first level) of Project:<br>");
    document.write("=====<br>");
    for(; !childElements.atEnd(); childElements.moveNext())
    {
        child = childElements.item();
        document.write("Name: '" + child.Name +
            "' | Type: '" + child.ObjectType + "'<br>");
    }

    //
    //ToDo: Access Project data
    //

}
catch (e)
{
    //Show Exception
    alert(e + ", " + e.message);

    //Disconnect from Automation Server
    Application.Disconnect();
}
</Script>
</Body>
</html>

```

5.6 Codebeispiel - Zugriff auf Geräte

Zugriff auf Geräte (= gerätespezifische AI-Erweiterung)

Programm:

```

<html>
<body>
<script language = "JavaScript">
    try
    {
        //Get Application object of Automation Interface
        var Application = window.external;

        //Alternatively: Initialization of Application object, this forces an
        //Internet Explorer Security Dialog
        //var Application = new ActiveXObject
        //("Rexroth.IndraWorks.Automation.Application");

        //Initialize Automation Interface (Connect to Automation Server)
        Application.Connect("localhost", "automation");

        //Get Project Access (use Example Code for creating "MyProject_001" before)
        var CurrentProject;
        var OpenProjects = Application.ProjectManager.Projects;
        for(var prjElements = new Enumerator(OpenProjects);
            !prjElements.atEnd(); prjElements.moveNext())
        {
            CurrentProject = prjElements.item();
            break;
        }
        if(( CurrentProject == null) || ( CurrentProject.Name != "MyProject_001"))
        {
            CurrentProject = Application.ProjectManager.OpenProject
            ("C:\\VAI-Test\\MyProject_001\\MyProject_001.iwp");
        }

        //-----
        //Get Device with Type name "MtxTestDevice_V001"
        //This example will cause an Exception cause the given typestring is
        //not a valid one. Do use this code to query for a valid typestring
        //of a device like MTX, HMI, IL or MLC.
        //-----
    }
    catch (e)
    {
        //Show Exception
        alert(e + ", " + e.message);

        //Disconnect from Automation Server
        Application.Disconnect();
    }
}
</script>
</body>
</html>

```

Codebeispiele - Basis Objekte

```

Children = CurrentProject.GetChildrenOfType("MtxTestDevice_V001", true);

//Show children of Type "MtxTestDevice_V001"
if( Children.Count == 0)
{
    alert("No ProjectElements of type 'MtxTestDevice_V001' found!");
}
else
{
    childElements = new Enumerator(Children);
    document.write("<br>All Files of Project:<br>");
    document.write("=====<br>");
    for(; !childElements.atEnd(); childElements.moveNext())
    {
        child = childElements.item();
        document.write("Name: '" + child.Name +
            "' | Type: '" + child.ObjectType + "'<br>");
    }
}

//
//ToDo: Access Project data
//

}
catch (e)
{
    //Show Exception
    alert(e + ", " + e.message);

    //Disconnect from Automation Server
    Application.Disconnect();
}
</Script>
</Body>
</html>

```

5.7 Codebeispiel - Projekt archivieren

Erzeugen eines IndraWorks Projektarchives (*.zip)

Programm:

```

<html>
<body>
<script language = "JavaScript">
try
{
    //Get Application object of Automation Interface
    var Application = window.external;

    //Alternatively: Initialization of Application object, this forces
    //an Internet Explorer Security Dialog
    //var Application = new ActiveXObject
    //("Rexroth.IndraWorks.Automation.Application");

    //Initialize Automation Interface (Connect to Automation Server)
    Application.Connect("localhost", "automation");

    //Get Project Manager
    var ProjectManager = Application.ProjectManager;

    try
    {
        //-----
        //Archive Project 'IndraWorksProject_001'
        //-----
        //Create Scripting.Dictionary object
        var archiveArgs = new ActiveXObject("Scripting.Dictionary");

        //Add the archive arguments as Key-/Value- pairs
        archiveArgs.Add('ProjectFile',
            'D:\\Projects\\IndraWorksProject_001\\IndraWorksProject_001.iwp');
        archiveArgs.Add('ArchiveFile', 'C:\\IWArchive_001.zip');
        archiveArgs.Add('ArchiveComment', 'Das ist der Archiv Kommentar...');
        archiveArgs.Add('Password', 'MyPassword-123');
        archiveArgs.Add('CloseProject', 'True');

        //Do Archive process
        ProjectManager.ArchiveProject( archiveArgs);
    }
}

```

```

    }
    catch (e)
    {
        //Show Exception
        alert(e + ", " + e.message);

        //
        //ToDo: Exception handling
        //
    }
}
catch (e)
{
    //Show Exception
    alert(e + ", " + e.message);

    //Disconnect from Automation Server
    Application.Disconnect();
}
</Script>
</Body>
</html>

```

5.8 Codebeispiel - Projekt wiederherstellen

Wiederherstellen eines IndraWorks Projektes, aus einem Projektarchiv (*.zip)

Programm:

```

<html>
<body>
<script language = "JavaScript">
    try
    {
        //Get Application object of Automation Interface
        var Application = window.external;

        //Alternatively: Initialization of Application object, this forces
        //an Internet Explorer Security Dialog
        //var Application = new ActiveXObject
        //("Rexroth.IndraWorks.Automation.Application");

        //Initialize Automation Interface (Connect to Automation Server)
        Application.Connect("localhost", "automation");

        //Get Project Manager
        var ProjectManager = Application.ProjectManager;

        try
        {
            //-----
            //Restore IndraWorks Project Archive 'IWArchive_001.zip'
            //-----
            //Create Scripting.Dictionary object for restore parameters
            var restoreArgs = new ActiveXObject("Scripting.Dictionary");

            //Add the restore arguments as Key-/Value- pairs
            restoreArgs.Add('ArchiveFile', 'C:\\IWArchive_001.zip');
            restoreArgs.Add('RestoreDir', 'D:\\Projects');
            restoreArgs.Add('NewProjectName', 'IndraWorksProject_001_NewName');
            restoreArgs.Add('Password', 'MyPassword-123');
            restoreArgs.Add('OpenProject', 'True');

            //Do Restore process
            ProjectManager.RestoreProject( restoreArgs);
        }
        catch (e)
        {
            //Show Exception
            alert(e + ", " + e.message);

            //
            //ToDo: Exception handling
            //
        }
    }
    catch (e)
    {
        //Show Exception
        alert(e + ", " + e.message);
    }
}

```

Codebeispiele - Basis Objekte

```

        //Disconnect from Automation Server
        Application.Disconnect();
    }
</Script>
</Body>
</html>

```

5.9 Codebeispiel - Projekt validieren

Durchführen einer Default Validierung für ein Projekt mit anschließender Ausgabe der Ergebnisse

Programm:

```

<html>
<body>
<script language = "JavaScript">

    var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
    try
    {
        debugger;

        //Get Application object of Automation Interface
        var Application = window.external;

        //Alternatively: Initialization of Application object,
        // this forces an Internet Explorer Security Dialog
        //var Application = new ActiveXObject("Rexroth.IndraWorks.Automation.Application");

        //Initialize Automation Interface (Connect to Automation Server)
        Application.Connect("localhost", "automation");

        var CurrentProject;
        var OpenProjects = Application.ProjectManager.Projects;
        for(var prjElements=new Enumerator(OpenProjects); !prjElements.atEnd(); prjElements.moveNext())
        {
            CurrentProject = prjElements.item();
            break;
        }

        document.write("<b>Scanning project elements =====</b><br>");
        var thisMn0DoesntSupportValidate;
        for(var mnos = new Enumerator(CurrentProject.Children); !mnos.atEnd(); mnos.moveNext())
        {
            var mno = mnos.item();
            document.write("'" + mno.Name + "': ");
            if (mno.SupportsValidate == false)
            {
                thisMn0DoesntSupportValidate = mno; //Take the last element found
                document.write("NOT ");
            }
            document.write("supported<br>");
        }
        document.write("<br>");

        //-----
        //Do validation for ProjectNode
        //-----
        document.write("<b>Validation of '" + CurrentProject.Name + "' =====</b><br>");

        //Create Scripting.Dictionary object
        var args = new ActiveXObject("Scripting.Dictionary");

        //Add the arguments as Key-/Value- pairs
        args.Add("PKG-FirstLevelDevice","");
        //This is a key used in the platform SamplePackage (PKG)

        var result;
        var errorCode = 0;
        var errorsFound = 0;
        var cancelled = 0;
        try
        {
            result = CurrentProject.Validate(args);
        }
        catch (exception)
        {
            errorCode = exception.number;

```

```

        document.write("ErrorCode=" + errorCode + "<br>");
        errorsFound = errorCode & 0x0080;
        cancelled = errorCode & 0x0001;
        result = exception.message;
    }
    if (errorsFound != 0)
    {
        document.write(" Error(s) found!" + "<br>");
    }
    if (cancelled != 0)
    {
        document.write(" Validation aborted!" + "<br>");
    }
}

//Store the XML result to a file
var fso = Application.CreateComInstance("Scripting.FileSystemObject");
var xmlFile = "c:\\\\ValidationResults.xml";
var fh = fso.CreateTextFile(xmlFile, true);
fh.WriteLine(result);
fh.Close();

//Show XML result
document.write("Returned XML values: -----<br>");

xmlDoc.loadXML(result);
var childNodes = xmlDoc.documentElement.childNodes;
var xmlNodes = new Enumerator(childNodes);
var xmlElement;
var xmlMessages;
for( ; !xmlNodes.atEnd(); xmlNodes.moveNext())
{
    xmlElement = xmlNodes.item();
    if (xmlElement.nodeName == "Messages")
    {
        xmlMessages = xmlElement;
    }
    else
    {
        //Errors, Warnings,...
        document.write(xmlElement.nodeName + "=" + xmlElement.text + "<br>");
    }
}

document.write("Messages:<br>");
xmlNodes = new Enumerator(xmlMessages.childNodes);
for( ; !xmlNodes.atEnd(); xmlNodes.moveNext())
{
    xmlElement = xmlNodes.item();
    document.write(
        ""
        + xmlElement.getAttribute("Type") + "' '"
        + xmlElement.getAttribute("Id") + "' '"
        + xmlElement.getAttribute("Text") + "' '"
        + xmlElement.getAttribute("Element") + "' '"
        + xmlElement.getAttribute("Position") + "' '"
        + xmlElement.getAttribute("Path") + "'<br>");
}

//Now validate a project element --> we expect an InvalidOperationException
try
{
    if (thisMn0DoesntSupportValidate != null)
    {
        document.write(
            "<br><b>Validation of a project element that doesn't support validation =====</b><br>");
        result = thisMn0DoesntSupportValidate.Validate(args); //Start validation
    }
}
catch (exception)
{
    //catch InvalidOperationException
    document.write("Expected InvalidOperationException:<br>");
    document.write("ErrorCode = " + exception.number + "<br>");
    document.write("Message = " + exception.message);
}
}
catch (e)
{
    document.write("<br><br>!!!!!! Exception occurred in JavaScript !!!!!!!<br>");
    document.write("Error name=" + e.name + "<br>");
    document.write("Error code=" + e.number + "<br>");
    document.write("Error message=" + e.message + "<br>");
}

```

Codebeispiele - Basis Objekte

```
        //Disconnect from Automation Server
        Application.Disconnect();
    }
</Script>
</Body>
</html>
```

6 XLC/MLC-Objekte

6.1 Objekt Modell

Das folgende Diagramm zeigt die Objekthierarchie des Automation Interfaces für XLC/MLC-Objekte.

Die im Diagramm grün dargestellten Objekte sind systemspezifische Objekte der XLC bzw. MLC, das beige dargestellte Objekt **Device** ist Bestandteil des **Basis Automation Interfaces**.

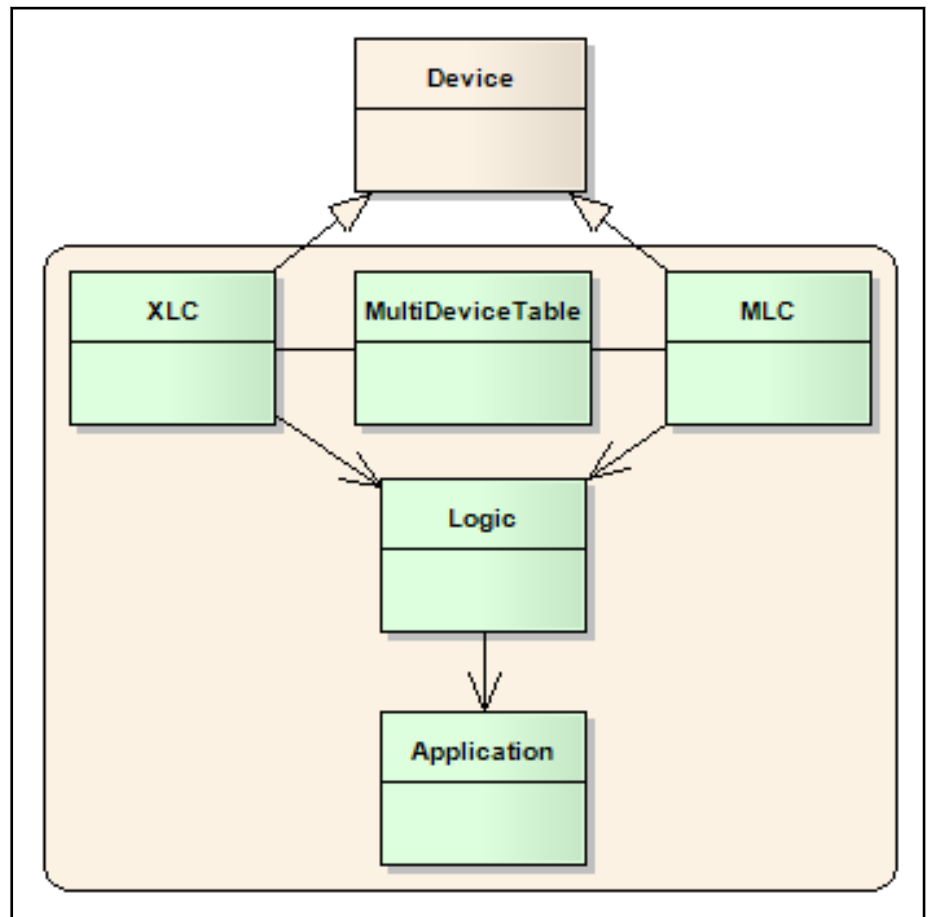


Abb.6-1: Objektmodell: Objekt Device des Basis Automation Interfaces und zugeordnete XLC/MLC-Objekte

6.2 Objekt - XLC (abgeleitet von Device)

Das Objekt **XLC** repräsentiert ein Gerät des Typs **XLC**, d.h. ein systemspezifisches Gerät innerhalb eines IndraWorks Projektes.

Das Objekt **XLC** ist abgeleitet vom Objekt **Device** und bietet daher alle Attribute und Methoden dieses Objektes an. Hierzu gehören z.B. das Attribut **IsOnline** sowie die Methoden **GoOnline** und **GoOffline**.

Auf das unterhalb der **XLC** liegende Objekt **Logic** kann mit Hilfe von **GetObject** zugegriffen werden.

XLC/MLC-Objekte

Attribute von XLC

Attribut	Typ	Beschreibung
IpAddress	string	Ermöglicht das Lesen und Schreiben der IP-Adresse der Steuerung.
IsMultiDeviceEnabled	bool	Nur Lesen. Liefert die Information zurück, ob die Multi-Device Funktionalität verwendet wird oder nicht. True: Multi-Device wird verwendet. False: Multi-Device wird nicht verwendet.
IsOnline	bool	Nur Lesen, von "Device" geerbt. True: XLC im IndraWorks-Projekt ist mit physikalischer XLC verbunden. False: Keine Verbindung zwischen XLC im IndraWorks-Projekt und physikalischer XLC.
MultiDeviceTable	object	Nur Lesen. Liefert eine Referenz auf das XLC-interne Objekt "MultiDeviceTable" zurück.

Methoden von XLC

Methode	Typ	Beschreibung
ClearErrors()	void	Löscht alle Fehler auf der Steuerung und in den angeschlossenen sercos Antrieben. Hierzu muss die XLC im Zustand "Online" sein.
DisableMultiDevice()	void	Multi-Device Funktionalität nicht verwenden. Kann die Multi-Device Funktionalität nicht deaktiviert werden, wird ein Ausnahmefehler ausgegeben.
DownloadFirmware(string fileName)	void	Lädt die in der Datei "fileName" enthaltene Firmware auf die Steuerung. Hierzu muss sich die XLC im Downloadmodus befinden (sercos Phase P0) und die SPS muss gestoppt sein. Parameter: fileName [string], enthält den vollständigen Namen und Pfad einer Firmwaredatei vom Typ "FW". Beispiel: "C:\FW\CML65s-XLs-12T01.0040.fw"
EnableMultiDevice()	void	Multi-Device Funktionalität verwenden. Hierzu darf die XLC weder im Zustand "online" noch im Zustand "eingeloggt" sein. Kann die Multi-Device Funktionalität nicht verwendet werden, weil z.B. die XLC im Zustand "online" und/oder "eingeloggt" ist, wird ein Ausnahmefehler ausgegeben.
GetObject(string object)	object	Liefert eine Referenz auf ein der XLC zugeordnetes Objekt zurück. Ist das angeforderte Objekt unbekannt, wird ein Ausnahmefehler ausgegeben. Parameter: object [string], String des gesuchten Objekts. Als Objekt existiert derzeit "Logic".
GetPhase()	int	Liefert die aktuelle sercos Phase zurück. Hierzu muss die XLC im Zustand "online" sein. Kann die Phase nicht gelesen werden, weil z.B. die XLC nicht im Zustand "online" ist, wird ein Ausnahmefehler ausgegeben.
GoOffline()	void	Von "Device" geerbt. Trennt eine Verbindung zwischen der im IndraWorks-Projekt enthaltenen XLC und einer physikalischen XLC. Kann nicht offline gegangen werden, wird ein Ausnahmefehler ausgegeben.

XLC/MLC-Objekte

Methode	Typ	Beschreibung
GoOnline()	void	Von "Device" geerbt. Stellt eine Verbindung zwischen der im IndraWorks-Projekt enthaltenen XLC und einer physikalischen XLC her. Kann die Online-Verbindung nicht hergestellt werden, weil z.B. die SPS angehalten werden muss oder die projektierte Antriebskonfiguration nicht mit der realen Antriebskonfiguration übereinstimmt, wird der Verbindungsaufbau abgebrochen und ein Ausnahmefehler ausgegeben.

XLC/MLC-Objekte

Methode	Typ	Beschreibung
GoOnline2(Scripting.Dictionary args)	void	Ermöglicht dem Benutzer, das Verhalten beim online Schalten gezielt zu steuern. Parameter: args [System.Dictionary], In einem System.Dictionary Objekt werden alle Parameter für das online Schalten als Key/Value Pairs übergeben. Es stehen die nachfolgend beschriebenen Key/Value Pairs zur Verfügung. Bei ungültigen Angaben oder falls ein online Schalten nicht möglich ist, wird ein Ausnahmefehler (Exception) generiert. Key: "AdjustC2CMotionSercosCycleTimes" [string] [Optionales Key/Value-Paar] Value: [bool], gibt an, ob bei Verwendung eines C2C-Querkommunikationsmoduls (Aufsteck-Modul zur Kommunikation zwischen Steuerungen) die Zykluszeiten für die C2C-Querkommunikation, für die Motion (Motion-Zykluszeit) sowie für die Kommunikation mit den sercos Geräten (Zykluszeit der Führungskommunikation) abgeglichen werden sollen oder nicht, falls die Zykluszeiten unterschiedlich sind. True: Bei Verwendung eines C2C-Querkommunikationsmoduls werden die Zykluszeiten für die Motion und für die Führungskommunikation, falls ungleich der Zykluszeit für die C2C-Querkommunikation, gleich dem Wert der Zykluszeit für die C2C-Querkommunikation gesetzt. Dies ist Voraussetzung, um den sercos Ring in den Betriebsmodus (sercos Phase 4) schalten zu können. False (Standardwert): Bei Verwendung eines C2C-Querkommunikationsmoduls werden die Zykluszeiten für die Motion und für die Führungskommunikation, falls ungleich der Zykluszeit für die C2C-Querkommunikation, so belassen wie sie sind. In diesem Fall (Zykluszeit der C2C-Querkommunikation ungleich der Zykluszeiten für Motion und Führungskommunikation) kann bei Verwendung eines C2C-Querkommunikationsmoduls nicht in den Betriebsmodus (sercos Phase 4) geschaltet werden. Beispiel (JavaScript): <pre>args.Add("AdjustC2CMotionSercosCycleTimes", true);</pre> Key: "CanOverwriteProject" [string] [Optionales Key/Value-Paar] Value: [bool], gibt an, ob ein in der Steuerung enthaltenes fremdes Projekt überschrieben werden darf. True (Standardwert): Ein in der Steuerung enthaltenes fremdes Projekt wird beim online Schalten überschrieben. False: Ein in der Steuerung enthaltenes fremdes Projekt wird beim online Schalten nicht überschrieben. Das online Schalten wird in diesem Fall abgebrochen. Beispiel (JavaScript): <pre>args.Add("CanOverwriteProject", false);</pre> Key: "CanStopPlcWhenAxesAreNotActive" [string] [Optionales Key/Value-Paar] Value: [bool], gibt an, ob die SPS bei nicht aktiven Achsen gestoppt werden darf. True: Falls die Achskonfiguration in IndraWorks geändert wurde, die SPS läuft (Zustand "RUN"), und keine Achse aktiv ist, wird die SPS beim online Schalten gestoppt. Ist jedoch mindestens eine Achse aktiv (reale Achsen "Power on", virtuelle Achsen nicht in "Standstill") kann die SPS grundsätzlich nicht gestoppt werden. In diesem Fall (SPS kann nicht gestoppt werden) wird das online Schalten abgebrochen und es wird ein Ausnahmefehler (Exception) generiert. False (Standardwert): Falls die Achskonfiguration in IndraWorks geändert wurde und die SPS läuft (Zustand "RUN"), wird die SPS beim online Schalten nicht gestoppt. Das gilt auch dann, wenn keine Achse aktiv ist. In diesem Fall wird das online Schalten abgebrochen, und es wird ein Ausnahmefehler (Exception) generiert.

XLC/MLC-Objekte

Methode	Typ	Beschreibung
		Beispiel (JavaScript): <pre>args.Add("CanStopPlcWhenAxesAreNotActive", true);</pre> Key: "CanIgnoreSercosDeviceInconsistencies" [string] [Optionales Key/Value-Paar] Value: [bool], gibt an, ob eine nicht vorhandene Übereinstimmung zwischen projektierten sercos Geräten und am sercos Bus tatsächlich vorhandenen Geräten möglichst ignoriert werden soll. True (Standardwert): Wird beim online Schalten festgestellt, dass die am sercos Bus angeschlossenen Geräte nicht mit den projektierten sercos Geräten übereinstimmt, wird dennoch versucht, online zu schalten. Ist dies nicht möglich, wird das online Schalten abgebrochen und es wird ein Ausnahmefehler (Exception) generiert. False: Wird beim online Schalten festgestellt, dass die am sercos Bus angeschlossenen Geräte nicht mit den projektierten sercos Geräten übereinstimmt, wird das online Schalten abgebrochen und es wird ein Ausnahmefehler (Exception) generiert. Beispiel (JavaScript): <pre>args.Add("CanIgnoreSercosDeviceInconsistencies", false);</pre>
ImportParameter(string fileName)	string	Importiert die in der Datei "fileName" enthaltenen sercos Parameter. Hierzu muss die XLC im Zustand "online" sein. Eventuelle Schreibfehler wie z.B. "Datum zur Zeit schreibgeschützt" werden durch die Methode gesammelt und am Ende des Parameter Imports in Form eines Strings zurückgeliefert. Das Format einer Zeile ist "IDN Parametername Ergebnis". Parameter: fileName [string], enthält den vollständigen Namen und Pfad der zu importierenden Datei vom Typ "PAR". Beispiel: "C:\Archive\Parameters.par"

XLC/MLC-Objekte

Methode	Typ	Beschreibung
ImportParameter2(string fileName, Scripting.Dictionary args)	string	Importiert die in der Datei "fileName" enthaltenen sercos Parameter. Hierzu muss die XLC im Zustand "online" sein. Im Unterschied zu "ImportParameter" muss angegeben werden, welche Parametertypen in welche Achsen (bzw. Steuerungen) importiert werden sollen. Eventuelle Schreibfehler wie z.B. "Datum zur Zeit schreibgeschützt" werden durch die Methode gesammelt und am Ende des Parameter Imports in Form eines Strings zurückgeliefert. Das Format einer Zeile ist "IDN Parametername Ergebnis". Parameter: fileName [string], enthält den vollständigen Namen und Pfad der zu importierenden Datei vom Typ "PAR". Beispiel: "C:\Archive\Parameters.par". args [System.Dictionary] In einem System.Dictionary Objekt werden alle spezifischen Angaben zum Parameter Import übergeben. Folgende Keys und Values stehen zur Verfügung: Key: "C-ParameterImportListByNumber" [string] [Optionales Key/Value-Paar], Steuerungsparameter. Value: [string], gibt an, ob C-Parameter geladen werden sollen oder nicht (Default). Enthält die Parameterdatei C-Parameter und sollen die C-Parameter importiert werden, so ist der String "*->*" anzugeben. Wird der Key "C-ParameterImportListByNumber" nicht angegeben oder wird ein Leerstring "" angegeben (Defaultwert), so werden C-Parameter nicht geladen. Enthält die Parameterdatei keine C-Parameter, so wird die Angabe ignoriert. Beispiel (JavaScript): <pre>args.Add("C-ParameterImportListByNumber", "*->*");</pre> Lädt in der Parameterdatei enthaltene C-Parameter in die Steuerung. Beispiel (JavaScript): <pre>args.Add("C-ParameterImportListByNumber", "");</pre> Hat zur Folge, dass in der Parameterdatei enthaltene C-Parameter nicht geladen werden. Key: "M-ParameterImportListByNumber" [string] [Optionales Key/Value-Paar], Messtasterparameter. Value: [string], gibt an, ob M-Parameter geladen werden sollen oder nicht (Default). Gleiches Verhalten wie "C-ParameterImportListByNumber" Key: "O-ParameterImportListByNumber" [string] [Optionales Key/Value-Paar], Oszilloskopparameter. Value: [string], gibt an, ob O-Parameter geladen werden sollen oder nicht (Default). Gleiches Verhalten wie "C-ParameterImportListByNumber". Key: "A-ParameterImportListByNumber" [string] [Optionales Key/Value-Paar], Antriebsbezogene Parameter in der Steuerung. Value: [string], gibt an, ob A-Parameter in welche Achse geladen werden sollen oder nicht (Default). Enthält die Parameterdatei A-Parameter und sollen die A-Parameter geladen werden, so können mit einer Komma separierten Liste die Nummern der Achsen angegeben werden, auf die die A-Parameter zu laden sind. Es können mit "-" auch Bereiche definiert werden. Mit "" werden die A-Parameter nicht geladen (Defaultwert). Wird der Schlüssel "A-ParameterImportListByNumber" nicht angegeben, werden in der Parameterdatei enthaltene A-Parameter nicht geladen. Enthält die Parameterdatei keine A-Parameter, so wird die Angabe ignoriert.

XLC/MLC-Objekte

Methode	Typ	Beschreibung
		Beispiel (JavaScript): <pre>args.Add("A-ParameterImportListByNumber", "1->1; 2->2,3-6,8");</pre> Lädt in der Parameterdatei enthaltene A-Parameter für die Achse mit der Nummer 1 in die konfigurierte Achse mit der Nummer 1. Die in der Parameterdatei enthaltenen A-Parameter für die Achse mit der Nummer 2 werden in die konfigurierten Achsen mit den Nummern 2, 3, 4, 5, 6 und 8 geladen. Beispiel (JavaScript): <pre>args.Add("A-ParameterImportListByNumber", "*->*");</pre> Lädt in der Parameterdatei enthaltene A-Parameter in die in der Parameterdatei angegebenen Achsen. Beispiel (JavaScript): <pre>args.Add("A-ParameterImportListByNumber", "");</pre> Hat zur Folge, dass in der Parameterdatei enthaltene A-Parameter nicht geladen werden. Beispiel (JavaScript): <pre>// nothing</pre> Wird der Schlüssel "A-ParameterImportListByNumber" nicht angegeben, werden in der Parameterdatei enthaltene A-Parameter nicht geladen. Key: "SP-ParameterImportListByNumber" [string] [Optionales Key/Value-Paar], Antriebsparameter im Antrieb. Value: [string], gibt an, ob S/P-Parameter in welchen Antrieb geladen werden sollen oder nicht (Default). Gleiches Verhalten wie "A-ParameterImportListByNumber". Key: "SP-ParameterImportListByAddress" [string] [Optionales Key/Value-Paar], Antriebsparameter im Antrieb. Value: [string], gibt an, ob S/P-Parameter in welchen Antrieb geladen werden sollen oder nicht (Default). Gleiches Verhalten wie "SP-ParameterImportListByNumber", mit dem Unterschied, dass bei "SP-ParameterImportListByAddress" sercos Adressen anstelle logischer Achsnummern anzugeben sind.
ReadListParameter(string idn)	Rexroth .IndraWorks.Automaton.Ai-List	Liest einen Listenparameter der Steuerung oder eines sercos Geräts (Listenparameter = Parameter variabler Länge). Hierzu muss die MLC im Zustand "online" sein. Die einzelnen Elemente des Listenparameters werden in einer "AiList" als eine Liste von Strings [string] zurückgegeben. Parameter: idn [string], Identnummer des Parameters, also z.B. "C-0-0001" oder "C-0-0001.0.0". Bei achsbezogenen Parametern (A-, S-, P-Parameter) und bei Kinematik-Parametern (K-Parameter) muss die dreistellige Bezugsnummer mit angegeben werden, also z.B. "A001:A-0-0001" oder "A001:A-0-0001.0.0", sowie "A001:S-0-0001" oder "K001:K-0-0001".
ReadParameter(string idn)	string	Liest einen "normalen" Parameter der Steuerung oder eines sercos Geräts. Hierzu muss die MLC im Zustand "online" sein. Parameter: idn [string], Identnummer des Parameters. Für die Schreibweise der Identnummer gilt das Gleiche wie unter "ReadListParameter".

XLC/MLC-Objekte

Methode	Typ	Beschreibung
SwitchPhase(int phase)	void	Schaltet die sercos Phase um. Hierzu muss die XLC im Zustand "online" sein. Kann die Phase nicht umgeschaltet werden, weil z.B. eine ungültige Phase vorgegeben ist oder die Leitachse freigegeben ist, wird ein Ausnahmefehler ausgegeben. Parameter: phase [int], enthält die Zielphase in numerischer Form. Folgende Zielphasen sind zulässig: • 0 (Downloadmodus) • 2 (Parametriermodus) • 4 (Betriebsmodus)
WriteListParameter(string idn, Rexroth.IndraWorks.Automation.AiList values)	void	Schreibt einen Listenparameter der Steuerung oder eines sercos Geräts (Listenparameter = Parameter variabler Länge). Die einzelnen Elemente des Listenparameters müssen in einer "AiList" als Liste von Strings [string] übergeben werden. Parameter: idn [string], Identnummer des Parameters. Für die Schreibweise der Identnummer gilt das Gleiche wie unter "ReadListParameter". values [Rexroth.IndraWorks.Automation.AiList], enthält die Elemente des zu schreibenden Listenparameters als Liste von Strings [string].
WriteParameter(string idn, string value)	void	Schreibt einen "normalen" Parameter der Steuerung oder eines sercos Geräts. Parameter: idn [string], Identnummer des Parameters. Für die Schreibweise der Identnummer gilt das Gleiche wie unter "ReadListParameter". value [string], enthält den zu schreibenden Wert.

6.3 Hinzufügen einer XLC

Unter Verwendung des Basis Automation Interface Objekts **Project** können dem aktuellen Projekt unterhalb des Projektknotens XLC-Geräte hinzugefügt werden. Hierzu ist die Methode **CreateChild** des Objekts **Project** zu verwenden.

Der Methode **CreateChild** sind zwei Funktionsargumente zu übergeben: der Typ des zu generierenden Objekts (**XLC**) sowie ein **Scripting.Dictionary** Objekt, das die speziellen Eigenschaften der XLC in Form von Key/Value Pairs enthält.

Methode CreateChild aus Objekt Project

Methode	Typ	Beschreibung
CreateChild(string childLibGuid, object args)	Rexroth .IndraW orks.Au- tomati- on.Pro- jectEle- ment	Über diese Methode des Basis Automation Interfaces kann ein "ProjectElement" vom Typ "File", "Folder" oder "Device" als Child des Projektknotens angelegt werden. Zum Anlegen eines "ProjectElements" vom Typ "XLC" müssen bzw. können die folgenden Argumente verwendet werden. Anlegen einer XLC: Parameter: childLibGuid [string], das zu erstellende Objekt ("XLC"). args [System.Dictionary] In einem System.Dictionary Objekt werden Eigenschaftswerte der XLC als Key/Value Paare übergeben. Folgende Keys und Values stehen zur Verfügung: Key: "Hardware" [string] Value: [string], XLC Hardware Bezeichnung. Folgende Formate werden unterstützt: • "CML25.1-3N" • "CML25.1-3N-400-NA-NNNN-NW" • "CML25.1-PN" • "CML25.1-PN-400-NA-NNNN-NW" • "CML45.1-3P" • "CML45.1-3P-500-NA-NNNN-NW" • "CML45.1-NP" • "CML45.1-NP-500-NA-NNNN-NW" • "CML65.1-3P" • "CML65.1-3P-504-NA-NNNN-NW" • "CML65.1-NP" • "CML65.1-NP-504-NA-NNNN-NW" Ausgewertet wird derzeit nur der vordere Teil, also z.B. "CML25.1-3N". Wird der hintere Teil mit angegeben, so muss das Format dem hier angegebenen Format entsprechen.

XLC/MLC-Objekte

Methode	Typ	Beschreibung
		Key: "Firmware" [string] Value: [string] XLC Firmware Bezeichnung. Folgende Firmware-Bezeichnungen werden unterstützt: • "FWA-CMLx5*-XL*-12V02" • "FWA-CMLx5*-XL*-12V04" • "FWA-CMLx5*-XL*-12V04.1" • "FWA-CMLx5*-XL*-12V04.2" • "FWA-CMLx5*-XL*-12V06" • "FWA-CMLx5*-XL*-12VRS" • "FWA-CMLx5*-XL*-13V02" • "FWA-CMLx5*-XL*-13VRS" Legende: CMLx5 = CML25, CML45, CML65 Die Firmware muss zur Hardware "passen", es kann also z.B. keine Hardware "CML25.1-3N" zusammen mit der Firmware "FWA-CML45*-XL*-12VRS" verwendet werden (Konflikt L25, L45). Bei der Angabe von "VRS" anstelle von z.B. "V02" wird automatisch das neueste Firmware-Release verwendet. Key: "Name" [string] [Optionales Key/Value-Paar] Value: [string], Name der Steuerung. Ist keine Name angegeben, wird automatisch ein Name vergeben. Key: "IpAddress" [string] [Optionales Key/Value-Paar] Value: [string], IP-Adresse der Steuerung. Ist keine IP-Adresse angegeben, wird eine IP-Adresse automatisch vergeben. Key: "PlcGateway" [string] [Optionales Key/Value-Paar] Value: [string], IP-Adresse des SPS-Gateways. Ist kein SPS-Gateway angegeben, wird "localhost" verwendet. Key: "PlcSecureOnlineMode" [string] [Optionales Key/Value-Paar] Value: [string], "True" wenn der sichere Online-Modus verwendet werden soll, ansonsten "False" (Default). Key: "Author" [string] [Optionales Key/Value-Paar] Value: [string], Name der Person, die die Steuerung dem Projekt hinzugefügt hat. Key: "Comment" [string] [Optionales Key/Value-Paar] Value: [string], Kommentar.

6.4 Objekt - MLC (abgeleitet von Device)

Das Objekt **MLC** repräsentiert ein Device des Typs **MLC**, d.h. ein systemspezifisches Gerät innerhalb eines IndraWorks Projektes. Seine Schnittstelle ist identisch zum Objekt **XLC**, d.h. es besitzt die gleichen Attribute und Methoden.

Attribute von MLC

Attribut	Typ	Beschreibung
IpAddress	string	Ermöglicht das Lesen und Schreiben der IP-Adresse der Steuerung.
IsMultiDeviceEnabled	bool	Nur Lesen. Liefert die Information zurück, ob die Multi-Device Funktionalität verwendet wird oder nicht. True: Multi-Device wird verwendet. False: Multi-Device wird nicht verwendet.
IsOnline	bool	Nur Lesen, von "Device" geerbt. True: MLC im IndraWorks-Projekt ist mit physikalischer MLC verbunden. False: Keine Verbindung zwischen MLC im IndraWorks-Projekt und physikalischer MLC.
MultiDeviceTable	object	Nur Lesen. Liefert eine Referenz auf das MLC-interne Objekt "MultiDeviceTable" zurück.

Methoden von MLC

Methode	Typ	Beschreibung
ClearErrors()	void	Löscht alle Fehler auf der Steuerung und in den angeschlossenen sercos Antrieben. Hierzu muss die MLC im Zustand "Online" sein.
DisableMultiDevice()	void	Multi-Device Funktionalität nicht verwenden. Kann die Multi-Device Funktionalität nicht deaktiviert werden, wird ein Ausnahmefehler ausgegeben.
DownloadFirmware(string fileName)	void	Lädt die in der Datei "fileName" enthaltene Firmware auf die Steuerung. Hierzu muss sich die MLC im Downloadmodus befinden (sercos Phase P0) und die SPS muss gestoppt sein. Parameter: fileName [string], enthält den vollständigen Namen und Pfad der die Firmware enthaltenden Datei vom Typ "FW". Beispiel: "C:\FW\CML45s-MLs-12T01.0040.fw"
EnableMultiDevice()	void	Multi-Device Funktionalität verwenden. Hierzu darf die MLC weder im Zustand "online" noch im Zustand "eingeloggt" sein. Kann die Multi-Device Funktionalität nicht verwendet werden, weil z.B. die MLC im Zustand "online" und/oder "eingeloggt" ist, wird ein Ausnahmefehler ausgegeben.
GetObject(string object)	object	Liefert eine Referenz auf ein der MLC zugeordnetes Objekt zurück. Ist das angeforderte Objekt unbekannt, wird ein Ausnahmefehler ausgegeben. Parameter: object [string], String des gesuchten Objekts. Als Objekt existiert derzeit "Logic".
GetPhase()	int	Liefert die aktuelle sercos Phase zurück. Hierzu muss die MLC im Zustand "online" sein. Kann die Phase nicht gelesen werden, weil z.B. die MLC nicht im Zustand "online" ist, wird ein Ausnahmefehler ausgegeben.

XLC/MLC-Objekte

Methode	Typ	Beschreibung
GoOffline()	void	Von "Device" geerbt. Trennt eine Verbindung zwischen der im IndraWorks-Projekt enthaltenen MLC und einer physikalischen MLC. Kann nicht offline gegangen werden, wird ein Ausnahmefehler ausgegeben.
GoOnline()	void	Von "Device" geerbt. Stellt eine Verbindung zwischen der im IndraWorks-Projekt enthaltenen MLC und einer physikalischen MLC her. Kann die Online-Verbindung nicht hergestellt werden, weil z.B. die SPS angehalten werden muss oder die projektierte Antriebskonfiguration nicht mit der realen Antriebskonfiguration übereinstimmt, wird der Verbindungsaufbau abgebrochen und ein Ausnahmefehler ausgegeben.

XLC/MLC-Objekte

Methode	Typ	Beschreibung
GoOnline2(Scripting.Dictionary args)	void	Ermöglicht dem Benutzer, das Verhalten beim online Schalten gezielt zu steuern. Parameter: args [System.Dictionary], In einem System.Dictionary Objekt werden alle Parameter für das online Schalten als Key/Value Pairs übergeben. Es stehen die nachfolgend beschriebenen Key/Value Pairs zur Verfügung. Bei ungültigen Angaben oder falls ein online Schalten nicht möglich ist, wird ein Ausnahmefehler (Exception) generiert. Key: "AdjustC2CMotionSercosCycleTimes" [string] [Optionales Key/Value-Paar] Value: [bool], gibt an, ob bei Verwendung eines C2C-Querkommunikationsmoduls (Aufsteck-Modul zur Kommunikation zwischen Steuerungen) die Zykluszeiten für die C2C-Querkommunikation, für die Motion (Motion-Zykluszeit) sowie für die Kommunikation mit den sercos Geräten (Zykluszeit der Führungskommunikation) abgeglichen werden sollen oder nicht, falls die Zykluszeiten unterschiedlich sind. True: Bei Verwendung eines C2C-Querkommunikationsmoduls werden die Zykluszeiten für die Motion und für die Führungskommunikation, falls ungleich der Zykluszeit für die C2C-Querkommunikation, gleich dem Wert der Zykluszeit für die C2C-Querkommunikation gesetzt. Dies ist Voraussetzung, um den sercos Ring in den Betriebsmodus (sercos Phase 4) schalten zu können. False (Standardwert): Bei Verwendung eines C2C-Querkommunikationsmoduls werden die Zykluszeiten für die Motion und für die Führungskommunikation, falls ungleich der Zykluszeit für die C2C-Querkommunikation, so belassen wie sie sind. In diesem Fall (Zykluszeit der C2C-Querkommunikation ungleich der Zykluszeiten für Motion und Führungskommunikation) kann bei Verwendung eines C2C-Querkommunikationsmoduls nicht in den Betriebsmodus (sercos Phase 4) geschaltet werden. Beispiel (JavaScript): <pre>args.Add("AdjustC2CMotionSercosCycleTimes", true);</pre> Key: "CanOverwriteProject" [string] [Optionales Key/Value-Paar] Value: [bool], gibt an, ob ein in der Steuerung enthaltenes fremdes Projekt überschrieben werden darf. True (Standardwert): Ein in der Steuerung enthaltenes fremdes Projekt wird beim online Schalten überschrieben. False: Ein in der Steuerung enthaltenes fremdes Projekt wird beim online Schalten nicht überschrieben. Das online Schalten wird in diesem Fall abgebrochen. Beispiel (JavaScript): <pre>args.Add("CanOverwriteProject", false);</pre> Key: "CanStopPlcWhenAxesAreNotActive" [string] [Optionales Key/Value-Paar] Value: [bool], gibt an, ob die SPS bei nicht aktiven Achsen gestoppt werden darf. True: Falls die Achskonfiguration in IndraWorks geändert wurde, die SPS läuft (Zustand "RUN"), und keine Achse aktiv ist, wird die SPS beim online Schalten gestoppt. Ist jedoch mindestens eine Achse aktiv (reale Achsen "Power on", virtuelle Achsen nicht in "Standstill") kann die SPS grundsätzlich nicht gestoppt werden. In diesem Fall (SPS kann nicht gestoppt werden) wird das online Schalten abgebrochen, und es wird ein Ausnahmefehler (Exception) generiert. False (Standardwert): Falls die Achskonfiguration in IndraWorks geändert wurde und die SPS läuft (Zustand "RUN"), wird die SPS beim online Schalten nicht gestoppt. Das gilt auch dann, wenn keine Achse aktiv ist. In diesem Fall wird das online Schalten abgebrochen, und es wird ein Ausnahmefehler (Exception) generiert.

XLC/MLC-Objekte

Methode	Typ	Beschreibung
		Beispiel (JavaScript): <pre>args.Add("CanStopPlcWhenAxesAreNotActive", true);</pre> Key: "CanIgnoreSercosDeviceInconsistencies" [string] [Optionales Key/Value-Paar] Value: [bool], gibt an, ob eine nicht vorhandene Übereinstimmung zwischen projektierten sercos Geräten und am sercos Bus tatsächlich vorhandenen Geräten möglichst ignoriert werden soll. True (Standardwert): Wird beim online Schalten festgestellt, dass die am sercos Bus angeschlossenen Geräte nicht mit den projektierten sercos Geräten übereinstimmt, wird dennoch versucht online zu schalten. Ist dies nicht möglich, wird das online Schalten abgebrochen und es wird ein Ausnahmefehler (Exception) generiert. False: Wird beim online Schalten festgestellt, dass die am sercos Bus angeschlossenen Geräte nicht mit den projektierten sercos Geräten übereinstimmt, wird das online Schalten abgebrochen und es wird ein Ausnahmefehler (Exception) generiert. Beispiel (JavaScript): <pre>args.Add("CanIgnoreSercosDeviceInconsistencies", false);</pre>
ImportParameter(string fileName)	string	Importiert die in der Datei "fileName" enthaltenen sercos Parameter. Hierzu muss die MLC im Zustand "online" sein. Eventuelle Schreibfehler wie z.B. "Datum zur Zeit schreibgeschützt" werden durch die Methode gesammelt und am Ende des Parameter Imports in Form eines Strings zurückgeliefert. Das Format einer Zeile ist "IDN Parametername Ergebnis". Parameter: fileName [string], enthält den vollständigen Namen und Pfad der zu importierenden Datei vom Typ "PAR". Beispiel: "C:\Archive\Parameters.par"

XLC/MLC-Objekte

Methode	Typ	Beschreibung
ImportParameter2(string fileName, Scripting.Dictionary args)	string	Importiert die in der Datei "fileName" enthaltenen sercos Parameter. Hierzu muss die MLC im Zustand "online" sein. Im Unterschied zu "ImportParameter" muss angegeben werden, welche Parametertypen in welche Achsen (bzw. Steuerungen) importiert werden sollen. Eventuelle Schreibfehler wie z.B. "Datum zur Zeit schreibgeschützt" werden durch die Methode gesammelt und am Ende des Parameter Imports in Form eines Strings zurückgeliefert. Das Format einer Zeile ist "IDN Parametername Ergebnis". Parameter: fileName [string], enthält den vollständigen Namen und Pfad der zu importierenden Datei vom Typ "PAR". Beispiel: "C:\Archive\Parameters.par" args [System.Dictionary] In einem System.Dictionary Objekt werden alle spezifischen Angaben zum Parameter Import übergeben. Folgende Keys und Values stehen zur Verfügung: Key: "C-ParameterImportListByNumber" [string] [Optionales Key/Value-Paar], Steuerungsparameter. Value: [string], gibt an, ob C-Parameter geladen werden sollen oder nicht (Default). Enthält die Parameterdatei C-Parameter und sollen die C-Parameter importiert werden, so ist der String "*" anzugeben. Wird der Key "C-ParameterImportListByNumber" nicht angegeben oder wird ein Leerstring "" angegeben (Defaultwert), so werden C-Parameter nicht geladen. Enthält die Parameterdatei keine C-Parameter, so wird die Angabe ignoriert. Beispiel (JavaScript): <pre>args.Add("C-ParameterImportListByNumber", "*->*");</pre> Lädt in der Parameterdatei enthaltene C-Parameter in die Steuerung. Beispiel (JavaScript): <pre>args.Add("C-ParameterImportListByNumber", "");</pre> Hat zur Folge, dass in der Parameterdatei enthaltene C-Parameter nicht geladen werden. Key: "K-ParameterImportListByNumber" [string] [Optionales Key/Value-Paar], Kinematikparameter. Value: [string], gibt an, ob K-Parameter geladen werden sollen oder nicht (Default). Gleiches Verhalten wie "C-ParameterImportListByNumber" Key: "M-ParameterImportListByNumber" [string] [Optionales Key/Value-Paar], Messtasterparameter. Value: [string], gibt an, ob M-Parameter geladen werden sollen oder nicht (Default). Gleiches Verhalten wie "C-ParameterImportListByNumber" Key: "O-ParameterImportListByNumber" [string] [Optionales Key/Value-Paar], Oszilloskopparameter. Value: [string], gibt an, ob M-Parameter geladen werden sollen oder nicht (Default). Gleiches Verhalten wie "C-ParameterImportListByNumber".

XLC/MLC-Objekte

Methode	Typ	Beschreibung
		Key: "A-ParameterImportListByNumber" [string] [Optionales Key/Value-Paar], Antriebsbezogene Parameter in der Steuerung. Value: [string], gibt an, ob A-Parameter in welche Achse geladen werden sollen oder nicht (Default). Enthält die Parameterdatei A-Parameter und sollen die A-Parameter geladen werden, so können mit einer Komma separierten Liste die Nummern der Achsen angegeben werden, auf die die A-Parameter zu laden sind. Es können mit "-" auch Bereiche definiert werden. Mit "" werden die A-Parameter nicht geladen (Defaultwert). Wird der Schlüssel "A-ParameterImportListByNumber" nicht angegeben, werden in der Parameterdatei enthaltene A-Parameter nicht geladen. Enthält die Parameterdatei keine A-Parameter, so wird die Angabe ignoriert. Beispiel (JavaScript): <pre>args.Add("A-ParameterImportListByNumber", "1->1; 2->2,3-6,8");</pre> Lädt in der Parameterdatei enthaltene A-Parameter für die Achse mit der Nummer 1 in die konfigurierte Achse mit der Nummer 1. Die in der Parameterdatei enthaltenen A-Parameter für die Achse mit der Nummer 2 werden in die konfigurierten Achsen mit den Nummern 2, 3, 4, 5, 6 und 8 geladen. Beispiel (JavaScript): <pre>args.Add("A-ParameterImportListByNumber", "*->*");</pre> Lädt in der Parameterdatei enthaltene A-Parameter in die in der Parameterdatei angegebenen Achsen. Beispiel (JavaScript): <pre>args.Add("A-ParameterImportListByNumber", "");</pre> Hat zur Folge, dass in der Parameterdatei enthaltene A-Parameter nicht geladen werden. Beispiel (JavaScript): <pre>// nothing</pre> Wird der Schlüssel "A-ParameterImportListByNumber" nicht angegeben, werden in der Parameterdatei enthaltene A-Parameter nicht geladen. Key: "SP-ParameterImportListByNumber" [string] [Optionales Key/Value-Paar], Antriebsparameter im Antrieb. Value: [string], gibt an, ob S/P-Parameter in welchen Antrieb geladen werden sollen oder nicht (Default). Gleiches Verhalten wie "A-ParameterImportListByNumber". Key: "SP-ParameterImportListByAddress" [string] [Optionales Key/Value-Paar], Antriebsparameter im Antrieb. Value: [string], gibt an, ob S/P-Parameter in welchen Antrieb geladen werden sollen oder nicht (Default). Gleiches Verhalten wie "SP-ParameterImportListByNumber", mit dem Unterschied, dass bei "SP-ParameterImportListByAddress" sercos Adressen anstelle logischer Achsnummern anzugeben sind.

XLC/MLC-Objekte

Methode	Typ	Beschreibung
ReadListParameter(string idn)	Rexroth .IndraWorks.Automation.AiList	Liest einen Listenparameter der Steuerung oder eines sercos Geräts (Listenparameter = Parameter variabler Länge). Die einzelnen Elemente des Listenparameters werden in einer "AiList" als eine Liste von Strings [string] zurückgegeben. Parameter: idn [string], Identnummer des Parameters, also z.B. "C-0-0001" oder "C-0-0001.0.0". Bei achsbezogenen Parametern (A-, S-, P-Parameter) und bei Kinematik-Parametern (K-Parameter) muss die dreistellige Bezugsnummer mit angegeben werden, also z.B. "A001:A-0-0001" oder "A001:A-0-0001.0.0", sowie "A001:S-0-0001" oder "K001:K-0-0001".
ReadParameter(string idn)	string	Liest einen "normalen" Parameter der Steuerung oder eines sercos Geräts. Parameter: idn [string], Identnummer des Parameters. Für die Schreibweise der Identnummer gilt das Gleiche wie unter "ReadListParameter".
SwitchPhase(int phase)	void	Schaltet die sercos Phase um. Hierzu muss die MLC im Zustand "online" sein. Kann die Phase nicht umgeschaltet werden, weil z.B. eine ungültige Phase vorgegeben ist oder die Leitachse freigegeben ist, wird ein Ausnahmefehler ausgegeben. Parameter: phase [int], enthält die Zielphase in numerischer Form. Folgende Zielphasen sind zulässig: • 0 (Downloadmodus) • 2 (Parametriermodus) • 4 (Betriebsmodus)
WriteListParameter(string idn, Rexroth.IndraWorks.Automation.AiList values)	void	Schreibt einen Listenparameter der Steuerung oder eines sercos Geräts (Listenparameter = Parameter variabler Länge). Hierzu muss die MLC im Zustand "online" sein. Die einzelnen Elemente des Listenparameters müssen in einer "AiList" als Liste von Strings [string] übergeben werden. Parameter: idn [string], Identnummer des Parameters. Für die Schreibweise der Identnummer gilt das Gleiche wie unter "ReadListParameter". values [Rexroth.IndraWorks.Automation.AiList], enthält die Elemente des zu schreibenden Listenparameters als Liste von Strings [string].
WriteParameter(string idn, string value)	void	Schreibt einen "normalen" Parameter der Steuerung oder eines sercos Geräts. Hierzu muss die MLC im Zustand "online" sein. Parameter: idn [string], Identnummer des Parameters. Für die Schreibweise der Identnummer gilt das Gleiche wie unter "ReadListParameter". value [string], enthält den zu schreibenden Wert.

6.5 Hinzufügen einer MLC

Unter Verwendung des Basis Automation Interface Objekts **Project** können dem aktuellen Projekt MLC-Geräte unterhalb des Projektknotens hinzugefügt werden. Identisch zum Gerät XLC, ist auch zum Hinzufügen einer MLC die Methode **CreateChild** des Objekts **Project** aufzurufen.

XLC/MLC-Objekte

Objekt - Project, Methode CreateChild

Methode	Typ	Beschreibung
CreateChild(string childLibGuid, object args)	Rexroth .IndraWorksAutomation.ProjectElement	Über diese Methode des Basis Automation Interfaces kann ein "ProjectElement" vom Typ "File", "Folder" oder "Device" als Child des Projektknotens angelegt werden. Zum Anlegen eines "ProjectElements" vom Typ "MLC" müssen bzw. können die folgenden Argumente verwendet werden. Anlegen einer MLC: Parameter: childLibGuid [string], das zu erstellende Objekt ("MLC"). args [System.Dictionary] In einem System.Dictionary Objekt werden Eigenschaftswerte der MLC als Key/Value Paare übergeben. Folgende Keys und Values stehen zur Verfügung: Key: "Hardware" [string] Value: [string], Hardware Bezeichnung. Folgende Formate werden unterstützt: • "CML25.1-3N" • "CML25.1-3N-400-NA-NNNN-NW" • "CML45.1-3P" • "CML45.1-3P-500-NA-NNNN-NW" • "CML65.1-3P" • "CML65.1-3P-504-NA-NNNN-NW" Ausgewertet wird derzeit nur der vordere Teil, also z.B. "CML25.1-3N". Wird der hintere Teil mit angegeben, so muss das Format dem hier angegebenen Format entsprechen. Key: "Firmware" [string] Value: [string], Firmware Bezeichnung. Folgende Firmware-Bezeichnungen werden unterstützt: • "FWA-CMLx5*-ML*-12V02" • "FWA-CMLx5*-ML*-12V04" • "FWA-CMLx5*-ML*-12V04.1" • "FWA-CMLx5*-ML*-12V04.2" • "FWA-CMLx5*-ML*-12V06" • "FWA-CMLx5*-ML*-12VRS" • "FWA-CMLx5*-ML*-13V02" • "FWA-CMLx5*-ML*-13VRS" Legende: CMLx5 = CML25, CML45, CML65

XLC/MLC-Objekte

Methode	Typ	Beschreibung
		Die Firmware muss zur Hardware "passen", es kann also z.B. keine Hardware "CML25.1-3N" zusammen mit der Firmware "FWA-CML45*-ML*-12VRS" verwendet werden (Konflikt L25, L45). Bei der Angabe von "VRS" anstelle von z.B. "V02" wird automatisch das neueste Firmware-Release verwendet. Key: "Name" [string] [Optionales Key/Value-Paar] Value: [string], Name der Steuerung. Ist keine Name angegeben, wird automatisch ein Name vergeben. Key: "IpAddress" [string] [Optionales Key/Value-Paar] Value: [string], IP-Adresse der Steuerung. Ist keine IP-Adresse angegeben, wird eine IP-Adresse automatisch vergeben. Key: "PlcGateway" [string] [Optionales Key/Value-Paar] Value: [string], IP-Adresse des SPS-Gateways. Ist kein SPS-Gateway angegeben, wird "localhost" verwendet. Key: "PlcSecureOnlineMode" [string] [Optionales Key/Value-Paar] Value: [string], "True" wenn der sichere Online-Modus verwendet werden soll, ansonsten "False" (Default). Key: "Author" [string] [Optionales Key/Value-Paar] Value: [string], Name der Person, die die Steuerung dem Projekt hinzugefügt hat. Key: "Comment" [string] [Optionales Key/Value-Paar] Value: [string], Kommentar.

6.6 Objekt - MultiDeviceTable

Das Objekt **MultiDeviceTable** ist ein XLC/MLC-spezifisches Objekt, welches ausschließlich Attribute und Methoden zur Multi-Device -Funktionalität zur Verfügung stellt. Der Zugriff auf dieses Objekt erfolgt ausgehend vom Objekt **XLC** bzw. **MLC** über das Attribut **MultiDeviceTable**.

Attribute von MultiDeviceTable

Attribut	Typ	Beschreibung
DeviceNames	Rexroth.IndraWorks.Automation.AiCollection	Nur Lesen. Liefert eine Collection von Strings mit den Namen aller in der Multi-Device Tabelle enthaltenen Geräte zurück.

XLC/MLC-Objekte

Methoden von MultiDeviceTable

Methode	Typ	Beschreibung
ActivateDevice(string deviceName)	void	Aktiviert ein in der Multi-Device Tabelle enthaltenes Gerät. Hierzu darf die XLC bzw. MLC weder im Zustand "online" noch im Zustand "eingeloggt" sein. Das aktivierte Gerät kann anschließend in IndraWorks verwendet werden, z.B. für einen SPS-Programmdownload oder einen Parameterimport. Kann das angegebene Gerät nicht aktiviert werden, weil z.B. das angegebene nicht in der Multi-Device Tabelle enthalten ist, wird ein Ausnahmefehler ausgegeben. Parameter: deviceName [string]: Name eines in der Multi-Device Tabellen enthaltenen Geräts, z.B. "XLC_21" oder "MLC_21" .

7 Codebeispiele - XLC/MLC-Objekte

7.1 Codebeispiel - Zugriff auf die XLC/MLC

Das Beispiel kann auch für eine XLC verwendet werden, wenn der Aufruf
 GetChildrenOfType("MLC", false) durch
 GetChildrenOfType("XLC", false) ersetzt wird.

Programm:

```
// IW application object
var objIWApplication = null;
// true = open new IW project, false = use already opened IW project
var booOpenProject = false;

try
{
    // Connect to IndraWorks
    var objIWApplication = WScript.GetObject("", "Rexroth.IndraWorks.Automation.Application");
    objIWApplication.Connect("localhost", "automation");

    // Project-Manager
    var objProjectManager = objIWApplication.ProjectManager;

    // Open or use an already existing project
    var objProject = null;
    if (booOpenProject)
    {
        objProject = objProjectManager.OpenProject("C:\\IW-Projects\\Project.iwp");
    }
    else
    {
        objProject = (new Enumerator(objProjectManager.Projects)).item();
    }

    // Get all devices of type "MLC"
    var objMLCs = objProject.GetChildrenOfType("MLC", false); // "XLC" <=> "MLC"
    WScript.Echo("" + objMLCs.Count + " MLC(s) in project found");

    // Iterate over all MLCs in the actual project until the specific MLC is found
    var objMLC = null;
    for (var enuMLCs = new Enumerator(objMLCs); !enuMLCs.atEnd(); enuMLCs.moveNext())
    {
        objMLC = enuMLCs.item();
        if (objMLC.Name == "IndraMotionMlc3")
        {
            // Do something with IndraMotionMlc3
            break;
        }
    }

    if (objMLC != null)
    {
        WScript.Echo("MLC found")
    }
    else
    {
        WScript.Echo("MLC not found")
    }
}
catch (ex)
{
    WScript.Echo("");
    WScript.Echo("=== Error ===");
    WScript.Echo("Message: " + ex.message);
    WScript.Echo("");
}
finally
{
    // Disconnect from IndraWorks
    if (objIWApplication != null)
    {
        objIWApplication.Disconnect();
    }
}
```

7.2 Codebeispiel - Online- / Offline-Schalten

Das Beispiel kann auch für eine XLC verwendet werden, wenn der Aufruf
 GetChildrenOfType("MLC", false) durch
 GetChildrenOfType("XLC", false) ersetzt wird.

Codebeispiele - XLC/MLC-Objekte

Programm:

```
// IW application object
var objIWApplication = null;
// true = open new IW project, false = use already opened IW project
var booOpenProject = false;

try
{
    // Connect to IndraWorks
    var objIWApplication = WScript.GetObject("", "Rexroth.IndraWorks.Automation.Application");
    objIWApplication.Connect("localhost", "automation");

    // Project-Manager
    var objProjectManager = objIWApplication.ProjectManager;

    // Open or use an already existing project
    var objProject = null;
    if (booOpenProject)
        objProject = objProjectManager.OpenProject("C:\\IW-Projects\\Project.iwp");
    else
        objProject = (new Enumerator(objProjectManager.Projects)).item();

    // Get first MLC in project
    var objMLCs = objProject.GetChildrenOfType("MLC", false); // "XLC" <=> "MLC"
    var objMLC = (new Enumerator(objMLCs)).item();

    // Go online
    objMLC.GoOnline();
    if (objMLC.IsOnline)
        WScript.Echo("MLC is online");
    else
        WScript.Echo("MLC is offline");

    // Go offline
    objMLC.GoOffline();
    if (objMLC.IsOnline)
        WScript.Echo("MLC is online");
    else
        WScript.Echo("MLC is offline");
}
catch (ex)
{
    WScript.Echo("");
    WScript.Echo("=== Error ===");
    WScript.Echo("Message: " + ex.message);
    WScript.Echo("");
}
finally
{
    // Disconnect from IndraWorks
    if (objIWApplication != null)
    {
        objIWApplication.Disconnect();
    }
}
}
```

7.3 Codebeispiel - Erweitertes Online- / Offline-Schalten

Das Beispiel kann auch für eine XLC verwendet werden, wenn der Aufruf
 GetChildrenOfType("MLC", false) durch
 GetChildrenOfType("XLC", false) ersetzt wird.

Programm:

```
// IW application object
var objIWApplication = null;
// true = open new IW project, false = use already opened IW project
var booOpenProject = false;

try
{
    // Connect to IndraWorks
    var objIWApplication = WScript.GetObject("", "Rexroth.IndraWorks.Automation.Application");
    objIWApplication.Connect("localhost", "automation");

    // Project-Manager
```

```

var objProjectManager = objIwApplication.ProjectManager;

// Open or use an already existing project
var objProject = null;
if (booOpenProject)
    objProject = objProjectManager.OpenProject("C:\\IW-Projects\\Project.iwp");
else
    objProject = (new Enumerator(objProjectManager.Projects)).item();

// Get first MLC in project
var objMLCs = objProject.GetChildrenOfType("MLC", false); // "XLC" <=> "MLC"
var objMLC = (new Enumerator(objMLCs)).item();

// Create dictionary object for storing the specific key/value pairs
var args = new ActiveXObject("Scripting.Dictionary")

// If there's another project than this on the control, don't overwrite it
args.Add("CanOverwriteProject", false);
// Adjust cycle times for C2C, motion and sercos
args.Add("AdjustC2CMotionSercosCycleTimes", true);
// If necessary, stop PLC, but only when axes are not active
args.Add("CanStopPlcWhenAxesAreNotActive", true);
// If there are inconsistencies between projected axes and axes on the sercos bus,
// cancel the online switching procedure
args.Add("CanIgnoreSercosDeviceInconsistencies", false);

try
{
    // Go online with specific settings
    objMLC.GoOnline2(args);
}
catch (ex)
{
    WScript.Echo("Error: Can't go online");
    WScript.Echo("Message: " + ex.message);
}
}
catch (ex)
{
    WScript.Echo("");
    WScript.Echo("=== Error ===");
    WScript.Echo("Message: " + ex.message);
    WScript.Echo("");
}
finally
{
    // Disconnect from IndraWorks
    if (objIwApplication != null)
    {
        objIwApplication.Disconnect();
    }
}
}

```

7.4 Codebeispiel - Phase lesen / schreiben

Das Beispiel kann auch für eine XLC verwendet werden, wenn der Aufruf
 GetChildrenOfType("MLC", false) durch
 GetChildrenOfType("XLC", false) ersetzt wird.

Programm:

```

// IW application object
var objIwApplication = null;
// true = open new IW project, false = use already opened IW project
var booOpenProject = false;

try
{
    // Connect to IndraWorks
    var objIwApplication = WScript.GetObject("", "Rexroth.IndraWorks.Automation.Application");
    objIwApplication.Connect("localhost", "automation");

    // Project-Manager
    var objProjectManager = objIwApplication.ProjectManager;

    // Open or use an already existing project
    var objProject = null;

```

Codebeispiele - XLC/MLC-Objekte

```

if (booOpenProject)
    objProject = objProjectManager.OpenProject("C:\\IW-Projects\\Project.iwp");
else
    objProject = (new Enumerator(objProjectManager.Projects)).item();

// Get first MLC in project
var objMLCs = objProject.ChildrenOfType("MLC", false); // "XLC" <=> "MLC"
var objMLC = (new Enumerator(objMLCs)).item();

// Go online
objMLC.GoOnline();

WScript.Echo("Actual phase: " + objMLC.GetPhase());

// Switch phase if not in P4
if (objMLC.GetPhase() != 4)
    objMLC.SwitchPhase(4);

WScript.Echo("Actual phase: " + objMLC.GetPhase());

// Go offline
objMLC.GoOffline();
}
catch (ex)
{
    WScript.Echo("");
    WScript.Echo("=== Error ===");
    WScript.Echo("Message: " + ex.message);
    WScript.Echo("");
}
finally
{
    // Disconnect from IndraWorks
    if (objIwApplication != null)
    {
        objIwApplication.Disconnect();
    }
}
}

```

7.5 Codebeispiel - Parameter importieren

Das Beispiel kann auch für eine XLC verwendet werden, wenn der Aufruf
 GetChildrenOfType("MLC", false) durch
 GetChildrenOfType("XLC", false) ersetzt wird.

Programm:

```

// IW application object
var objIwApplication = null;
// true = open new IW project, false = use already opened IW project
var booOpenProject = false;

try
{
    // Connect to IndraWorks
    var objIwApplication = WScript.GetObject("", "Rexroth.IndraWorks.Automation.Application");
    objIwApplication.Connect("localhost", "automation");

    // Project-Manager
    var objProjectManager = objIwApplication.ProjectManager;

    // Open or use an already existing project
    var objProject = null;
    if (booOpenProject)
        objProject = objProjectManager.OpenProject("C:\\IW-Projects\\Project.iwp");
    else
        objProject = (new Enumerator(objProjectManager.Projects)).item();

    // Get first MLC in project
    var objMLCs = objProject.ChildrenOfType("MLC", false); // "XLC" <=> "MLC"
    var objMLC = (new Enumerator(objMLCs)).item();

    // Go online
    objMLC.GoOnline();

    // Switch phase if not in P2
    if (objMLC.GetPhase() != 2)

```

```

    objMLC.SwitchPhase(2);

    // Import parameters
    var strErrors = objMLC.ImportParameter("C:\\IW-Projects\\Parameter.par")
    WScript.Echo("Import erros:");
    WScript.Echo(strErrors);

    // Go offline
    objMLC.GoOffline();
}
catch (ex)
{
    WScript.Echo("");
    WScript.Echo("=== Error ===");
    WScript.Echo("Message: " + ex.message);
    WScript.Echo("");
}
finally
{
    // Disconnect from IndraWorks
    if (objIwApplication != null)
    {
        objIwApplication.Disconnect();
    }
}
}

```

7.6 Codebeispiel - Erweitertes Parameter importieren

Das Beispiel kann auch für eine XLC verwendet werden, wenn der Aufruf
 GetChildrenOfType("MLC", false) durch
 GetChildrenOfType("XLC", false) ersetzt wird.

Programm:

```

// IW application object
var objIwApplication = null;
// true = open new IW project, false = use already opened IW project
var booOpenProject = false;

try
{
    // Connect to IndraWorks
    var objIwApplication = WScript.GetObject("", "Rexroth.IndraWorks.Automation.Application");
    objIwApplication.Connect("localhost", "automation");

    // Project-Manager
    var objProjectManager = objIwApplication.ProjectManager;

    // Open or use an already existing project
    var objProject = null;
    if (booOpenProject)
        objProject = objProjectManager.OpenProject("C:\\IW-Projects\\Project.iwp");
    else
        objProject = (new Enumerator(objProjectManager.Projects)).item();

    // Get first MLC in project
    var objMLCs = objProject.GetChildrenOfType("MLC", false); // "XLC" <=> "MLC"
    var objMLC = (new Enumerator(objMLCs)).item();

    // Go online
    objMLC.GoOnline();

    // Switch phase if not in P2
    if (objMLC.GetPhase() != 2)
        objMLC.SwitchPhase(2);

    // Create dictionary object for storing the specific key/value pairs
    var args = new ActiveXObject("Scripting.Dictionary")

    // *** Import only A- and S/P-parameters ***
    // Import A-parameters from axis 1 into axes 11-15, 21-25 and 31-35
    args.Add("A-ParameterImportListByNumber", "1->1-10,21-25,31-35");
    // Import S/P-parameters from drive axis 1 into drive axes 11-15, 21-25 and 31-35
    args.Add("SP-ParameterImportListByNumber", "1->1-10,21-25,31-35");

    // Import A- and S/P-parameters
    var strErrors = objMLC.ImportParameter("C:\\IW-Projects\\Parameter.par")
}

```

Codebeispiele - XLC/MLC-Objekte

```

WScript.Echo("Import erros:");
WScript.Echo(strErrors);

// Go offline
objMLC.GoOffline();
}
catch (ex)
{
    WScript.Echo("");
    WScript.Echo("=== Error ===");
    WScript.Echo("Message: " + ex.message);
    WScript.Echo("");
}
finally
{
    // Disconnect from IndraWorks
    if (objIwApplication != null)
    {
        objIwApplication.Disconnect();
    }
}
}

```

7.7 Codebeispiel - Parameter lesen / schreiben

Das Beispiel kann auch für eine XLC verwendet werden, wenn der Aufruf
 GetChildrenOfType("MLC", false) durch
 GetChildrenOfType("XLC", false) ersetzt wird.

Programm:

```

// IW application object
var objIwApplication = null;
// true = open new IW project, false = use already opened IW project
var booOpenProject = false;

try
{
    // Connect to IndraWorks
    var objIwApplication = WScript.GetObject("", "Rexroth.IndraWorks.Automation.Application");
    objIwApplication.Connect("localhost", "automation");

    // Project-Manager
    var objProjectManager = objIwApplication.ProjectManager;

    // Open or use an already existing project
    var objProject = null;
    if (booOpenProject)
        objProject = objProjectManager.OpenProject("C:\\IW-Projects\\Project.iwp");
    else
        objProject = (new Enumerator(objProjectManager.Projects)).item();

    // Get first MLC in project
    var objMLCs = objProject.GetChildrenOfType("MLC", false); // "XLC" <=> "MLC"
    var objMLC = (new Enumerator(objMLCs)).item();

    // Go online
    objMLC.GoOnline();

    // Switch phase if not in P2
    if (objMLC.GetPhase() != 2)
        objMLC.SwitchPhase(2);

    // Fixed length parameters

    // Read parameter
    var objValue = objMLC.ReadParameter("C-0-0081");
    WScript.Echo("Value of C-0-0081: " + objValue);

    // Write parameter
    objMLC.WriteParameter("C-0-0081", "New value of C-0-0081");

    // Reread parameter
    objValue = objMLC.ReadParameter("C-0-0081");
    WScript.Echo("Value of C-0-0081: " + objValue);

    // List parameters

```

Codebeispiele - XLC/MLC-Objekte

```

// Read list parameter values
var objAiList = objMLC.ReadListParameter("C-0-0484");
WScript.Echo("C-0-0484 contains a fixed length of " + objAiList.Count + " entries:");
for (var numIndex = 0; numIndex < objAiList.Count; numIndex++)
{
    WScript.Echo(objAiList.Item(numIndex));
}

// Write list parameter values
objAiList = new ActiveXObject("Rexroth.IndraWorks.Automation.AiList");
objAiList.Add("16");
objAiList.Add("0");
objAiList.Add("16");
objAiList.Add("0");

objMLC.WriteListParameter("C-0-0484", objAiList);

// Reread list parameter values
objAiList = objMLC.ReadListParameter("C-0-0484");
WScript.Echo("C-0-0484 contains a fixed length of " + objAiList.Count + " entries:");
for (var numIndex = 0; numIndex < objAiList.Count; numIndex++)
{
    WScript.Echo(objAiList.Item(numIndex));
}

// Go offline
objMLC.GoOffline();
}
catch (ex)
{
    WScript.Echo("");
    WScript.Echo("=== Error ===");
    WScript.Echo("Message: " + ex.message);
    WScript.Echo("");
}
finally
{
    // Disconnect from IndraWorks
    if (objIwApplication != null)
    {
        objIwApplication.Disconnect();
    }
}
}

```

7.8 Codebeispiel - Firmware-Download

Das Beispiel kann auch für eine XLC verwendet werden, wenn der Aufruf
 GetChildrenOfType("MLC", false) durch
 GetChildrenOfType("XLC", false) ersetzt wird.

Programm:

```

// IW application object
var objIwApplication = null;
// true = open new IW project, false = use already opened IW project
var booOpenProject = false;

try
{
    // Connect to IndraWorks
    var objIwApplication = WScript.GetObject("", "Rexroth.IndraWorks.Automation.Application");
    objIwApplication.Connect("localhost", "automation");

    // Project-Manager
    var objProjectManager = objIwApplication.ProjectManager;

    // Open or use an already existing project
    var objProject = null;
    if (booOpenProject)
        objProject = objProjectManager.OpenProject("C:\\IW-Projects\\Project.iwp");
    else
        objProject = (new Enumerator(objProjectManager.Projects)).item();

    // Get first MLC in project
    var objMLCs = objProject.GetChildrenOfType("MLC", false); // "XLC" <=> "MLC"
    var objMLC = (new Enumerator(objMLCs)).item();
}

```

Codebeispiele - XLC/MLC-Objekte

```
// Get Logic object
var objLogic = objMLC.GetObject("Logic");

// Get PLC application object (see PLC documentation)
var objPlcApplication = objLogic.GetObject("Application");

// Stop PLC
objPlcApplication.Stop();

// Go online
objMLC.GoOnline();

// Switch phase if not in P0 (download mode)
if (objMLC.GetPhase() != 0)
    objMLC.SwitchPhase(0);

// Download firmware
WScript.Echo("Downloading firmware...");
objMLC.DownloadFirmware("C:\\IW-Projects\\CML65s-MLs-13V02.0xxx.fw");

// Switch to P4
objMLC.SwitchPhase(4);

// Clear errors
objMLC.ClearErrors();

// Go offline
objMLC.GoOffline();
}
catch (ex)
{
    WScript.Echo("");
    WScript.Echo("=== Error ===");
    WScript.Echo("Message: " + ex.message);
    WScript.Echo("");
}
finally
{
    // Disconnect from IndraWorks
    if (objIwApplication != null)
    {
        objIwApplication.Disconnect();
    }
}
```

7.9 Codebeispiel - XLC/MLC hinzufügen

Programm:

```
// IW application object
var objIwApplication = null;
// true = open new IW project, false = use already opened IW project
var booOpenProject = false;

try
{
    // Connect to IndraWorks
    var objIwApplication = WScript.GetObject("", "Rexroth.IndraWorks.Automation.Application");
    objIwApplication.Connect("localhost", "automation");

    // Project-Manager
    var objProjectManager = objIwApplication.ProjectManager;

    // Open or use an already existing project
    var objProject = null;
    if (booOpenProject)
        objProject = objProjectManager.OpenProject("C:\\IW-Projects\\Project.iwp");
    else
        objProject = (new Enumerator(objProjectManager.Projects)).item();

    // *** XLC ***

    // Dictionary object for creating a XLC
    var args = new ActiveXObject("Scripting.Dictionary");

    // Mandatory arguments
    args.Add("Hardware", "CML25.1-3N-400-NA-NNNN-NW");
    args.Add("Firmware", "FWA-CML25*-XL*-13V02");
}
```

```

// Optional arguments
args.Add("Name", "XLC_110");
args.Add("IpAddress", "10.52.10.110");

// Create XLC
objProject.CreateChild("XLC", args);

// *** MLC ***

// Reuse the existing dictionary object
args.RemoveAll();

// Mandatory arguments
args.Add("Hardware", "CML65.1-3P-504-NA-NNNN-NW");
args.Add("Firmware", "FWA-CML65*-ML*-13V02");

// Optional arguments
args.Add("Name", "MLC_212");
args.Add("IpAddress", "10.52.10.212");
args.Add("PlcGateway", "localhost");
args.Add("PlcSecureOnlineMode", true);
args.Add("Author", "Thomas");
args.Add("Comment", "Control Unit 212");

// Create MLC
objProject.CreateChild("MLC", args);
}
catch (ex)
{
    WScript.Echo("");
    WScript.Echo("=== Error ===");
    WScript.Echo("Message: " + ex.message);
    WScript.Echo("");
}
finally
{
    // Disconnect from IndraWorks
    if (objIwApplication != null)
    {
        objIwApplication.Disconnect();
    }
}
}

```

7.10 Codebeispiel - Multi-Device verwenden / nicht verwenden

Das Beispiel kann auch für eine MLC verwendet werden, wenn der Aufruf
 GetChildrenOfType("XLC", false) durch
 GetChildrenOfType("MLC", false) ersetzt wird.

Programm:

```

// IW application object
var objIwApplication = null;
// true = open new IW project, false = use already opened IW project
var booOpenProject = false;

try
{
    // Connect to IndraWorks
    var objIwApplication = WScript.GetObject("", "Rexroth.IndraWorks.Automation.Application");
    objIwApplication.Connect("localhost", "automation");

    // Project-Manager
    var objProjectManager = objIwApplication.ProjectManager;

    // Open or use an already existing project
    var objProject = null;
    if (booOpenProject)
        objProject = objProjectManager.OpenProject("C:\\IW-Projects\\Project.iwp");
    else
        objProject = (new Enumerator(objProjectManager.Projects)).item();

    // Get first MLC in project
    var objMLCs = objProject.GetChildrenOfType("MLC", false); // "XLC" <=> "MLC"
    var objMLC = (new Enumerator(objMLCs)).item();
}

```

Codebeispiele - XLC/MLC-Objekte

```

// Be sure that we are offline with the motion
objMLC.GoOffline();

// Be sure that we are offline with the motion
var objLogic = objMLC.GetObject("Logic");

// Get PLC application object (see PLC documentation)
var objPlcApplication = objLogic.GetObject("Application");

// Be sure that we are logged out from the PLC
objPlcApplication.Logout();

// Enable multi-device
objMLC.EnableMultiDevice();
WScript.Echo("Multi-Device is " + (objMLC.IsMultiDeviceEnabled ? "enabled" : "disabled"));

// Disable multi-device
objMLC.DisableMultiDevice();
WScript.Echo("Multi-Device is " + (objMLC.IsMultiDeviceEnabled ? "enabled" : "disabled"));
}
catch (ex)
{
    WScript.Echo("");
    WScript.Echo("=== Error ===");
    WScript.Echo("Message: " + ex.message);
    WScript.Echo("");
}
finally
{
    // Disconnect from IndraWorks
    if (objIwApplication != null)
    {
        objIwApplication.Disconnect();
    }
}
}

```

7.11 Codebeispiel - Aktivieren einer XLC/MLC aus der Multi-Device Tabelle

Das Beispiel kann auch für eine MLC verwendet werden, wenn der Aufruf
 GetChildrenOfType("XLC", false) durch
 GetChildrenOfType("MLC", false) ersetzt wird.

Programm:

```

// IW application object
var objIwApplication = null;
// true = open new IW project, false = use already opened IW project
var booOpenProject = false;

try
{
    // Connect to IndraWorks
    var objIwApplication = WScript.GetObject("", "Rexroth.IndraWorks.Automation.Application");
    objIwApplication.Connect("localhost", "automation");

    // Project-Manager
    var objProjectManager = objIwApplication.ProjectManager;

    // Open or use an already existing project
    var objProject = null;
    if (booOpenProject)
        objProject = objProjectManager.OpenProject("C:\\IW-Projects\\Project.iwp");
    else
        objProject = (new Enumerator(objProjectManager.Projects)).item();

    // Get first MLC in project
    var objMLCs = objProject.GetChildrenOfType("MLC", false); // "XLC" <=> "MLC"
    var objMLC = (new Enumerator(objMLCs)).item();

    // Note: We must be offline (motion) and logged out (PLC)

    // Enable multi-device
    objMLC.EnableMultiDevice()

    // Get multi-device table object

```

```

var objMultiDeviceTable = objMLC.MultiDeviceTable;

// Iterate over all devices inside the AiCollection
var objDeviceNames = objMultiDeviceTable.DeviceNames;
for (var enuDeviceNames = new Enumerator(objDeviceNames); !enuDeviceNames.atEnd(); enuDeviceNames.moveNext())
{
    // Look for specific device inside the multi-device table
    var strDeviceName = enuDeviceNames.item();
    if (strDeviceName == "MLC3")
    {
        // Activate specific device
        objMultiDeviceTable.ActivateDevice(strDeviceName);
        break;
    }
}
}
catch (ex)
{
    WScript.Echo("");
    WScript.Echo("=== Error ===");
    WScript.Echo("Message: " + ex.message);
    WScript.Echo("");
}
finally
{
    // Disconnect from IndraWorks
    if (objIwApplication != null)
    {
        objIwApplication.Disconnect();
    }
}
}

```

7.12 Codebeispiel - Zugriff auf das Objekt Logic

Das Beispiel kann auch für eine XLC verwendet werden, wenn der Aufruf
 GetChildrenOfType("MLC", false) durch
 GetChildrenOfType("XLC", false) ersetzt wird.

Programm:

```

// IW application object
var objIwApplication = null;
// true = open new IW project, false = use already opened IW project
var booOpenProject = false;

try
{
    // Connect to IndraWorks
    var objIwApplication = WScript.GetObject("", "Rexroth.IndraWorks.Automation.Application");
    objIwApplication.Connect("localhost", "automation");

    // Project-Manager
    var objProjectManager = objIwApplication.ProjectManager;

    // Open or use an already existing project
    var objProject = null;
    if (booOpenProject)
    {
        objProject = objProjectManager.OpenProject("C:\\IW-Projects\\Project.iwp");
    }
    else
    {
        objProject = (new Enumerator(objProjectManager.Projects)).item();
    }

    // Get first MLC in project
    var objMLCs = objProject.GetChildrenOfType("MLC", false); // "XLC" ==> "MLC"
    var objMLC = (new Enumerator(objMLCs)).item();

    // Get Logic object
    var objLogic = objMLC.GetObject("Logic");

    // Get PLC application object (see PLC documentation)
    var objPlcApplication = objLogic.GetObject("Application");

    // Do something with objPlcApplication (see PLC documentation)
}
catch (ex)
{
    WScript.Echo("");
    WScript.Echo("=== Error ===");
}

```

Codebeispiele - XLC/MLC-Objekte

```
WScript.Echo("Message: " + ex.message);
WScript.Echo("");
}
finally
{
    // Disconnect from IndraWorks
    if (objIwApplication != null)
    {
        objIwApplication.Disconnect();
    }
}
```

8 IL-Objekte

8.1 Objekt – Logic

Das Objekt **Logic** repräsentiert das Unterobjekt der SPS-Logic.

Methoden von Logic

Methode	Typ	Beschreibung
GetObject(string object)	object	Liefert die Referenz auf ein Objekt, das dem Logic-Objekt zugeordnet ist. Parameter object [string], Bezeichnung des gesuchten Objekts. Zulässig ist derzeit das Objekt "Application". Beispiel: <code>var logic = device.GetObject("Logic");</code>

8.2 Objekt – Application

Das Objekt **Application** repräsentiert eine Applikation unterhalb des **Logic** Objekts. Über dieses Objekt können verschiedene Funktionen einer Applikation ausgeführt werden.

Für den Zugriff auf eine Applikation über das Automation Interface stehen folgende Kommandos zur Verfügung.

Attribute von Application

Methode	Typ	Beschreibung
IsRunning	bool	Liefert zurück, ob die SPS der Applikation läuft oder nicht. True: SPS läuft. False: SPS läuft nicht.
IsLoggedIn	bool	Gibt zurück, ob man auf der Applikation eingeloggt ist. True: Auf Applikation eingeloggt False: Nicht eingeloggt.

IL-Objekte

Methoden von Application

Methode	Typ	Beschreibung
Login()	string	Einloggen auf die Applikation. Wurde das SPS-Programm geändert, wird es übersetzt und anschließend via Download herunter geladen. Im Fehlerfall (z.B. wenn das SPS-Programm nicht fehlerfrei übersetzt werden konnte) wird ein Ausnahmefehler ausgegeben. Ansonsten werden z.B. Compilerwarnungen und andere Ausgaben als String zurückgegeben. Beispiel: <pre>var application = logic.GetObject("Application"); application.Login();</pre>
Login2(int mode)	string	Einloggen auf die Applikation. Wurde das SPS-Programm geändert, wird es übersetzt und anschließend wie folgt herunter geladen: Parameter: mode [int], • 0: Ohne Änderung einloggen • 1: Mit Online-Change einloggen • 2: Mit Download einloggen • 3: Wenn Online-Change möglich, mit Online-Change einloggen, sonst Download Im Fehlerfall (z.B. wenn das SPS-Programm nicht fehlerfrei übersetzt werden konnte) wird ein Ausnahmefehler ausgegeben. Ansonsten werden z.B. Compilerwarnungen und andere Ausgaben als String zurückgegeben. Beispiel: <pre>var application = logic.GetObject("Application"); application.Login2(0);</pre>
Logout()	void	Führt den Befehl "Ausloggen" aus.
Build()	string	Übersetzt das SPS-Programm. Liefert aufgetretene Fehler als String zurück. Trat kein Fehler auf, wird ein Leerstring zurückgegeben.
Rebuild()	string	Übersetzt das SPS-Programm neu. Liefert aufgetretene Fehler als String zurück. Trat kein Fehler auf, wird ein Leerstring zurückgegeben.
Download()	string	Überträgt das SPS-Programm in die Steuerung. Wurde das SPS-Programm geändert, wird es vorher neu übersetzt. Im Fehlerfall (z.B. wenn das SPS-Programm nicht fehlerfrei übersetzt werden konnte) wird ein Ausnahmefehler ausgegeben. Ansonsten werden z.B. Compilerwarnungen und andere Ausgaben als String zurückgegeben. Beispiel: <pre>var application = logic.GetObject("Application"); var erg = application.Download();</pre>

IL-Objekte

Methode	Typ	Beschreibung
Download2(int mode)	string	Überträgt das SPS-Programm wie folgt in die Steuerung: Parameter: mode [int], • 1: Mit Online-Change einloggen. • 2: Mit Download einloggen. • 3: Wenn Online-Change möglich, mit Online-Change einloggen, sonst Download. Im Fehlerfall (z.B. wenn das SPS-Programm nicht fehlerfrei übersetzt werden konnte) wird ein Ausnahmefehler ausgegeben. Ansonsten werden z.B. Compilerwarnungen und andere Ausgaben als String zurückgegeben. Beispiel: <pre>var application = logic.GetObject("Application"); var erg = application.Download2(0);</pre>
Start()	void	Startet die Applikation.
Stop()	void	Stoppt die Applikation.

9 Service und Support

Für Ihre schnelle und optimale Unterstützung verfügen wir über ein dichtes weltweites Servicenetz. Unsere Experten stehen Ihnen mit Rat und Tat zur Seite. Sie erreichen uns täglich **rund um die Uhr - auch am Wochenende und an Feiertagen**.

Service Deutschland

Unser technologieorientiertes Competence Center in Lohr deckt alle Belange rund um den Service für elektrische Antriebe und Steuerungen ab.

Die **Service Helpdesk & Hotline** erreichen Sie unter:

Telefon: **+49 9352 40 5060**
Fax: **+49 9352 18 4941**
E-Mail: service.svc@boschrexroth.de
Internet: <http://www.boschrexroth.com>

Auf unseren Internetseiten finden Sie ergänzende Hinweise zu Service, Reparatur (z.B. Anlieferadressen) und Training.

Service weltweit

Außerhalb Deutschlands nehmen Sie bitte zuerst Kontakt mit Ihrem Ansprechpartner auf. Die Hotline-Rufnummern entnehmen Sie bitte den Vertriebsadressen im Internet.

Vorbereitung der Informationen

Wir können Ihnen schnell und effizient helfen, wenn Sie folgende Informationen bereithalten:

- detaillierte Beschreibung der Störung und der Umstände
- Angaben auf dem Typenschild der betreffenden Produkte, insbesondere Typenschlüssel und Seriennummern
- Ihre Kontaktdaten (Telefon-, Faxnummer und E-Mail-Adresse)

Index

A

Aktivieren einer XLC/MLC aus der Multi-Device Tabelle	70
Application	
Attribute	73
Methoden	74
Objekt	73
Automation Interface	
Codebeispiele, XLC/MLC	61
IL-Objekte	73
Scripting	14
Verwendung von Automation-Scripts	14
XLC/MLC-Objekte	41
Automation Interface initialisieren.....	31

B

Bestimmungsgemäßer Gebrauch	
Anwendungsbereiche	11
Einleitung	11
Einsatzfälle	11

C

Codebeispiele, XLC/MLC.....	61
Aktivieren einer XLC/MLC aus der Multi-Device Tabelle	70
Erweitertes Parameter importieren	65
Firmware-Download	67
MLC hinzufügen	68
Multi-Device verwenden / nicht verwenden ..	69
Online- / Offline-Schalten	61
Parameter importieren	64
Parameter lesen / schreiben	66
Phase lesen / schreiben	63
XLC hinzufügen	68
Zugriff auf das Logic-Objekt	71
Zugriff auf die MLC	61
Zugriff auf die XLC	61

D

Durchsuchen von Projekten.....	33
--------------------------------	----

E

Erweiterte Codebeispiele, XLC/MLC	
Erweitertes Online- / Offline-Schalten	62
Erweitertes Online- / Offline-Schalten.....	62
Erweitertes Parameter importieren.....	65

F

Firmware-Download.....	67
------------------------	----

H

Hinzufügen von Ordnern und Dateien.....	32
---	----

I

IL-Objekte.....	73
IndraWorks Projekt anlegen.....	31
IndraWorks Projekt öffnen.....	32

L

Logic	
Methoden	73
Objekt	73

M

MLC	
Attribute	51
Hinzufügen	57
Methoden	51
Objekt	50
Objekt Modell	41
MLC hinzufügen.....	68
MultiDeviceTable	
Attribute	59
Methoden	60
Objekt	59
Multi-Device verwenden / nicht verwenden.....	69

N

Nicht-bestimmungsgemäßer Gebrauch.....	12
Folgen, Haftungsausschluss	11

O

Online- / Offline-Schalten.....	61
---------------------------------	----

P

Parameter importieren.....	64
Parameter lesen / schreiben.....	66
Phase lesen / schreiben.....	63
Projekt archivieren.....	36
Projekt validieren.....	38
Projekt wiederherstellen.....	37

S

Scripting.....	14
Support	
siehe Service-Hotline	77

V

Verwendung von Automation-Scripts.....	14
--	----

X

XLC	
Attribute	42
Hinzufügen	48

Index

X

...XLC	
Methoden	42
Objekt	41
Objekt Modell	41
XLC hinzufügen.....	68

Z

Zu dieser Dokumentation.....	3
Gültigkeit der Dokumentation	3

Z

Zugriff auf das Logic-Objekt	71
Zugriff auf die MLC.....	61
Zugriff auf die XLC.....	61
Zugriff auf Geräte.....	35

Notizen

Bosch Rexroth AG
Electric Drives and Controls
Postfach 13 57
97803 Lohr, Deutschland
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr, Deutschland
Tel. +49 9352 18 0
Fax +49 9352 18 8400
www.boschrexroth.com/electrics



R911336355

DOK-XLCMLC-AUT*INT*V13-AP01-DE-P