

Rexroth IndraLogic XLC IndraMotion MLC 13VRS Automation Interface

R911336356
Edition 01

Application Manual



Title Rexroth IndraLogic XLC
IndraMotion MLC 13VRS
Automation Interface

Type of Documentation Application Manual

Document Typecode DOK-XLCMLC-AUT*INT*V13-AP01-EN-P

Internal File Reference RS-f4cbe24321ce9b5a0a6846a000fd6245-1-en-US-4

Purpose of Documentation This documentation describes the script-based access to IndraWorks project data via the interface of the Automation Interface.

Record of Revision

Edition	Release Date	Notes
Edition 01	04.2012	First edition for 13VRS

Copyright © Bosch Rexroth AG 2012

This document, as well as the data, specifications and other information set forth in it, are the exclusive property of Bosch Rexroth AG. It may not be reproduced or given to third parties without its consent.

Liability The specified data is intended for product description purposes only and shall not be deemed to be a guaranteed characteristic unless expressly stipulated in the contract. All rights are reserved with respect to the content of this documentation and the availability of the product.

Published by Bosch Rexroth AG

Bgm.-Dr.-Nebel-Str. 2 ■ 97816 Lohr a. Main, Germany

Phone +49 9352 18 0 ■ Fax +49 9352 18 8400

<http://www.boschrexroth.com/>

System Development Automation Motion Logic Control, WoP, (SyMu/MePe)

Note This document has been printed on chlorine-free bleached paper.

Table of Contents

	Page
1 About this Documentation.....	3
1.1 Validity of the Documentation.....	3
1.2 Required and Supplementing Documentations.....	3
1.3 Information Representation.....	8
1.3.1 Symbols Used.....	8
1.3.2 Names and Abbreviations.....	8
2 Important Instructions on Use.....	9
2.1 Intended Use.....	9
2.1.1 Introduction.....	9
2.1.2 Areas of Use and Application.....	9
2.2 Unintended Use.....	10
3 Introduction.....	11
3.1 General.....	11
3.2 Scripting.....	12
3.3 Using Automation Scripts	12
4 Basic Objects.....	15
4.1 Object Model.....	15
4.2 Object - Application.....	15
4.3 Object - ProjectManager.....	17
4.4 Object - Project.....	19
4.5 Object - ProjectElement.....	22
4.6 Object - Device (Derived from ProjectElement).....	25
4.7 Object - File (Derived from ProjectElement).....	26
4.8 Object - Folder (Derived from ProjectElement).....	26
4.9 Object - AiCollection.....	26
4.10 Object - AiList.....	26
5 Example Codes - Basic Objects.....	29
5.1 Example Code - Initializing the Automation Interface.....	29
5.2 Example Code - Creating an IndraWorks Project.....	29
5.3 Example Code - Opening an IndraWorks Project.....	30
5.4 Example Code - Adding Folders and Files.....	30
5.5 Example Code - Searching in Projects.....	31
5.6 Example Code - Accessing Devices.....	33
5.7 Example Code - Archiving a Project.....	34
5.8 Example Code - Restoring a Project.....	35
5.9 Example Code - Validating a Project.....	36

Table of Contents

	Page
6 XLC/MLC Objects.....	39
6.1 Object Model.....	39
6.2 Object - XLC (Derived from Device).....	39
6.3 Adding an XLC.....	45
6.4 Object - MLC (Derived from Device).....	47
6.5 Adding an MLC.....	53
6.6 Object - MultiDeviceTable.....	55
7 Example Codes - XLC/MLC Objects.....	57
7.1 Example Code - Access to the XLC/MLC.....	57
7.2 Example Code - Going Online/Offline.....	57
7.3 Example Code - Extended Online/Offline Going.....	58
7.4 Example Code - Reading/Writing Phase.....	59
7.5 Example Code - Importing Parameter.....	60
7.6 Example Code - Extended Parameter Import.....	61
7.7 Example Code - Reading/Writing Parameter.....	62
7.8 Example Code - Firmware Download.....	63
7.9 Example Code - Adding XLC/MLC.....	64
7.10 Example Code - Using/Not Using Multi-Device.....	65
7.11 Example Code - Activating an XLC/MLC from the Multi-Device Table.....	66
7.12 Example Code - Accessing the "Logic" Object.....	67
8 IL Objects.....	69
8.1 Object – Logic.....	69
8.2 Object – Application.....	69
9 Service and Support.....	73
Index.....	75

1 About this Documentation

1.1 Validity of the Documentation

- Target group** This documentation describes the script-based access to the IndraWorks project data via the interface of the Automation Interface.
- This documentation supports the user during the phases:
- Engineering
 - Commissioning

1.2 Required and Supplementing Documentations

Documentation titles with type codes and parts numbers

IndraWorks		MLC	XLC
/36/	Rexroth IndraWorks 13VRS Software Installation DOK-IWORKS-SOFTINS*V13-COxx-EN-P, R911336880 This documentation describes the IndraWorks installation.	X	X
/5/	Rexroth IndraWorks 13VRS Engineering DOK-IWORKS-ENGINEE*V13-APxx-EN-P, R911336870 This documentation describes the application of IndraWorks in which the Rexroth Engineering tools are integrated. It includes instructions on how to work with IndraWorks and how to operate the oscilloscope function.	X	X
/20/	Rexroth IndraMotion MLC 13VRS Functional Description DOK-MLC***-FUNC****V13-APxx-EN-P, R911336295 This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraMotion MLC.	X	
/20/	Rexroth IndraLogic XLC 13VRS Functional Description DOK-XLC***-FUNC****V13-APxx-EN-P, R911336352 This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraLogic XLC.		X
/7/	Rexroth IndraWorks 13VRS CamBuilder DOK-IWORKS-CAMBUIL*V13-APxx-EN-P, R911336291 This documentation describes the basic principles and operation of the CamBuilder, the cam editing tool.	X	X
/37/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Automation Interface DOK-XLCMLC-AUT*INT*V13-APxx-EN-P, R911336356 This documentation describes the script-based access to IndraWorks project data via the interface of the Automation Interface.	X	X
/38/	Rexroth IndraWorks 12VRS FDT Container DOK-IWORKS-FDT*CON*V12-APxx-EN-P, R911334398 This documentation describes the IndraWorks FDT Container functionality. It includes the activation of the functionality in the project and working with DTMs.	X	X

About this Documentation

IndraWorks		MLC	XLC
/29/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Project Conversion DOK-XLCMLC-PROCONV*V13-APxx-EN-P, R911336366 This documentation describes the project conversion of IndraLogic 04VRS and IndraMotion MLC04VRS on IndraWorks version 12 with IndraLogic 2G. Changes with regard to Motion and PLC are described in detail.	X	X
/28/	Rexroth IndraMotion MLC 13VRS Commissioning DOK-MLC***-STARTUP*V13-COxx-EN-P, R911336308 This documentation describes the steps required to commission and perform service on the IndraMotion MLC system. It includes checklists for tasks to be frequently performed and a detailed description of the steps.	X	

xx Corresponding edition
 Fig. 1-1: XCL/MLC documentation overview

Motion		MLC	XLC
/23/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS PLCopen Libraries DOK-XLCMLC-FUNLIB**V13-LIxx-EN-P, R911336360 This documentation describes the function blocks, functions and data types of the RIL_Common-Types, ML_Base and ML_PLCopen libraries for the IndraLogic XLC/IndraMotion MLC. It also includes the error reactions of function blocks.	X	X
/27/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Generic Application Template DOK-XLCMLC-TF*GAT**V13-APxx-EN-P, R911336370 This documentation provides a structured template to the IndraLogic PLC programmer. This template can be used to add and edit the PLC programming code. It includes the template, the template wizard and example applications.	X	X
/31/	Rexroth IndraMotion MLC 13VRS RCL Programming Instruction DOK-MLC***-RCL*PRO*V13-APxx-EN-P, R911336297 This documentation provides information on the RobotControl. The programming language RCL (RobotControl Language) is focused. The program structure, variables, functions, motion statements and the required system parameters are described.	X	
/21/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS Parameters DOK-XLCMLC-PARAM***V13-RExx-EN-P, R911336364 This documentation describes the parameters of the XLC/MLC systems as well as the interaction between parameterization and programming. It includes the axis parameters, control parameters, kinematic parameters, touch probe parameters and programmable limit switch parameters.	X	X
/10/	Rexroth IndraDrive Firmware for Drive Controllers MPH-, MPB-, MPD-, MPC-07 DOK-INDRV*-MP*-07VRS**-FKxx-EN-P, R911328670 This documentation describes all functional properties of the IndraDrive firmware in the variants MPH-07, MPB-07, MPD-07 and MPC-07.		
/11/	Rexroth IndraDrive MPx-17 Functions DOK-INDRV*-MP*-17VRS**-APxx-EN-P, R911331236 This documentation describes all functional properties of the IndraDrive firmware in the variants MPB-17, MPM-17, MPC-17 and MPE-17.		

xx Corresponding edition
 Fig. 1-2: XCL/MLC documentation overview

About this Documentation

Field buses		MLC	XLC
/39/	Rexroth IndraMotion MLC 11VRS PLCopen Field Bus DOK-IM*ML*-PLCFBUS*V11-APxx-EN-P, R911333896 This documentation describes the creation of field bus drives in an IndraWorks project, function blocks, functions and data types of the RIL_CommonTypes.library (excerpt for field bus drives), the RMB_PLCOpenFieldBus.library and the RIL_Utilitys.library (excerpt for field bus drives). It also includes the error reactions of function blocks.	X	X
/4/	Rexroth IndraWorks 13VRS Field Buses DOK-IWORKS-FB*****V13-APxx-EN-P, R911336872 This documentation describes the supported field buses and their diagnostic function blocks.	X	X
/63/	SERCOS System Manual for I/O Devices DOK-CONTRL-ILS3*****-APxx-EN-P, R911333512 This documentation describes the configuration, parameterization, commissioning and diagnostics of I/O devices with a SERCOS interface.	X	X

xx Corresponding edition
 Fig. 1-3: XCL/MLC documentation overview

HMI		MLC	XLC
/8/	Rexroth IndraWorks 13VRS HMI DOK-IWORKS-HMI****V13-APxx-EN-P, R911336874 This documentation describes the functions, configuration and operation of the user interfaces IndraWorks HMI Engineering and IndraWorks HMI Operation.	X	X
/6/	Rexroth IndraWorks 13VRS WinStudio DOK-IWORKS-WINSTUD*V13-APxx-EN-P, R911336882 This documentation describes the installation of the software, working with WinStudio and the creation and operation of applications.	X	X
/50/	Rexroth IndraLogic XLC IndraMotion MLC 13VRS HMI Connection DOK-XLCMLC-HMI****V13-APxx-EN-P, R911336362 This documentation describes the visualization systems supported by the IndraLogic XLC and IndraMotion MLC and their connection.	X	X

xx Corresponding edition
 Fig. 1-4: XCL/MLC documentation overview

PLC		MLC	XLC
/3/	Rexroth IndraWorks 13VRS IndraLogic 2G Programming Instruction DOK-IWORKS-IL2GPRO*V13-APxx-EN-P, R911336876 This documentation describes the PLC programming tool IndraLogic 2G and its use. It includes the basic use, first steps, visualization, menu items and editors.	X	X
/33/	Rexroth IndraWorks 13VRS, Basic Libraries, IndraLogic 2G DOK-IL*2G*-BASLIB**V13-LIxx-EN-P, R911336285 This documentation describes the system-comprehensive PLC libraries.	X	X

xx Corresponding edition
 Fig. 1-5: XCL/MLC documentation overview

About this Documentation

Technology		MLC	XLC
/30/	Rexroth IndraMotion MLC 13VRS Technology Libraries DOK-MLC***-TF*LIB**V13-LIxx-EN-P, R911336324 This documentation describes the function blocks, functions and data types of the "ML_TechInterface.library", "ML_TechMotion.library", "RMB_TechCam.library" and "ML_TechBase.library". It also includes libraries for the winder functionality, register controller functionality and CrossCutter functionality.	X	
/60/	Rexroth IndraMotion MLC 13VRS RegisterControl (Library) DOK-MLC***-REGI*CO*V13-LIxx-EN-P, R911336306 This documentation describes the inputs and outputs of the individual function blocks and provides notes on their use.	X	
/62/	Rexroth IndraMotion MLC 13VRS RegisterControl (Application Manual) DOK-MLC***-REGI*CO*V13-APxx-EN-P, R911336304 This documentation describes the application of the integrated register control for a rotogravure printing machine. The components of the mark stream sensor, the HMI application and the error recovery options are described. This instruction provides information on how to operate the register control, react to errors and query diagnostics. This documentation is written for machine setters and machine operators.	X	
/49/	Rexroth IndraMotion MLC 13VRS Winder Function Application DOK-MLC***-TF*WIND*V13-APxx-EN-P, R911336326 This application-related system documentation describes the application of the winder technology functions.	X	

xx Corresponding edition
 Fig. 1-6: XCL/MLC documentation overview

Hardware		MLC	XLC
/1/	Rexroth IndraControl L45/L65/L85 Control DOK-CONTRL-ICL45L65L85-PRxx-EN-P, R911332116 This documentation describes the IndraControl L45/L65/L85 controls.	X	X
/2/	Rexroth IndraControl L25 DOK-CONTRL-IC*L25*****-PRxx-EN-P, R911328474 This documentation describes the IndraControl L25 controls.	X	X
/24/	Rexroth IndraControl Lxx 13VRS Function Modules DOK-CONTRL-FM*LXX**V13-APxx-EN-P, R911336283 This documentation describes all function modules of the Lxx controls including engineering and diagnostics.	X	X
/12/	Rexroth IndraDrive Drive Controllers MPx-02 to MPx-07 DOK-INDRV*-GEN-**VRS**-PAxx-EN-P, R911297317 This documentation describes all parameters implemented in the firmware for drive controllers of the IndraDrive family. It supports the parameterization of the drive controllers.		

About this Documentation

Hardware		MLC	XLC
/13/	<p>Rexroth IndraDrive MPx-02 to MPx-07 and HMV</p> <p>DOK-INDRV*-GEN-**VRS**-WAxx-EN-P, R911297319</p> <p>This documentation describes all diagnostics implemented in the following firmwares:</p> <ul style="list-style-type: none"> • Drive controller firmwares from MPx-02 to MPx-08 and • firmwares of the supply devices of the type "HMV". <p>It supports the operating crew as well as the programmer at troubleshooting.</p>		
/35/	<p>Rexroth IndraDrive Drive Controller Control Sections CSB01, CSH01, CDB01</p> <p>DOK-INDRV*-CSH*****-PRxx-EN-P, R911295012</p> <p>This documentation is a project planning manual to select and use the control sections CSB01, CDB01 and CSH01 for drive controllers of the product families Rexroth IndraDrive M and Rexroth IndraDrive C.</p>		

xx Corresponding edition
 Fig. 1-7: XCL/MLC documentation overview

Diagnostics and service		MLC	XLC
/26/	<p>Rexroth IndraMotion MLC/MLP/XLC 11VRS IndraMotion Service Tool</p> <p>DOK-IM*ML*-IMST***V11-RExx-EN-P, R911331940</p> <p>This documentation describes the IndraMotion Service Tool (IMST). It is a web-based diagnostic tool to access the control systems IndraMotion MLC L25, L45 or 65 as well as the MLP via an Ethernet high-speed connection. The IMST allows OEMs, end users and service engineers to access and remotely diagnose a system. A PC with Internet Explorer version 6 or 7 is required.</p>	X	X
/22/	<p>Rexroth IndraLogic XLC IndraMotion MLC 13VRS Diagnostics</p> <p>DOK-XLCMLC-DIAG***V13-RExx-EN-P, R911336358</p> <p>This documentation includes all control parameters implemented in the control systems IndraLogic XLC and IndraMotion MLC.</p>	X	X

xx Corresponding edition
 Fig. 1-8: XCL/MLC documentation overview

System Overview		MLC	XLC
/48/	<p>Rexroth IndraMotion for Printing 13VRS System Overview</p> <p>DOK-IM*PR*-SYSTEM**V13-PRxx-EN-P, R911336289</p> <p>This documentation describes the product IndraMotion for Packaging. It introduces the control systems, drive systems and I/O systems as well as the commissioning and programming.</p>	X	
/48/	<p>Rexroth IndraMotion for Packaging 13VRS System Overview</p> <p>DOK-IM*PA*-SYSTEM**V13-PRxx-EN-P, R911336287</p> <p>This documentation describes the product IndraMotion for Packaging. It introduces the control systems, drive systems and I/O systems as well as the commissioning and programming.</p>	X	

About this Documentation

System Overview		MLC	XLC
/9/	Rexroth IndraMotion MLC 13VRS System Overview DOK-MLC***-SYSTEM**V13-PRxx-EN-P, R911336310 This documentation provides an overview on the hardware/software components of the automation system IndraMotion MLC in the mentioned version. It helps assembling a system.	X	
/9/	Rexroth IndraLogic XLC 13VRS System Overview DOK-XLC***-SYSTEM**V13-PRxx-EN-P, R911336354 This documentation provides an overview on hardware/software components of the automation system IndraLogic XLC in the respective version. It helps assembling a system.		X

xx Corresponding edition
 Fig.1-9: XCL/MLC documentation overview

First steps		MLC	XLC
/25/	Rexroth IndraMotion MLC 13VRS, First Steps DOK-MLC***-F*STEP**V13-COxx-EN-P, R911336293 This documentation describes the first steps of the IndraMotion MLC and the RobotControl. It includes the hardware and software prerequisites as well as the creation of a project.	X	
/25/	Rexroth IndraLogic XLC 13VRS, First Steps DOK-XLC***-F*STEP**V13-COxx-EN-P, R911336350 This documentation describes the first steps of the IndraLogic XLC. It includes the hardware and software prerequisites as well as the creation of a project.		X

xx Corresponding edition
 Fig.1-10: XCL/MLC documentation overview

1.3 Information Representation

1.3.1 Symbols Used

Note Notes are represented as follows:



This is a note for the user.

Tip Tips are represented as follows:



This is a tip for the user.

1.3.2 Names and Abbreviations

Term	Meaning
AI	Automation Interface
HMI	Human machine interface
HTML	Hypertext markup language
IDE	Integrated development environment
MLC	MotionLogic Control
OLE	Object linking and embedding
VB	Visual Basic

Fig.1-11: Names and abbreviations used

2 Important Instructions on Use

2.1 Intended Use

2.1.1 Introduction

Rexroth products represent state-of-the-art developments and manufacturing. The products are tested prior to delivery to ensure operating safety and reliability.

The products may only be used according to their intended use. If they are not used as intended, situations occur that can result in damage to property and injury to persons.



Bosch Rexroth as the manufacturer is not liable for any damages resulting from unintended use. In such cases, the guarantee and the right to payment of damages resulting from unintended use are forfeited. The user alone carries all responsibility of the risks.

Before using Rexroth products, please ensure that all the pre-requisites for an intended use of the products are met:

- Anybody using Bosch Rexroth products in any way has to first read and understand the relevant safety instructions and be familiar with the intended use of the products.
- In case of hardware products, the products have to retain their original state, in other words, no structural changes are permitted. The decompilation of software products or the alteration of source codes is not permitted.
- Do not install or operate damaged or faulty products.
- Ensure that the products have been installed according to the provisions described in the relevant documentation.

2.1.2 Areas of Use and Application

Rexroth IndraMotion MLC

Rexroth IndraMotion MLC and its function modules are intended for Motion/Logic applications.



The Rexroth IndraMotion MLC and its function modules can only be used with the accessories and mounting parts listed in this documentation. Components that are not expressly mentioned can neither be attached nor connected. The same applies to cables and lines.

The systems can only be operated with the hardware component configurations and combinations that are expressly specified and with the software and firmware and specified in the respective documentation and functional descriptions.

Rexroth IndraMotion MLC and its function modules have been developed for use in single and multi-axis drive and control tasks.

For the application-specific use of the machine operating and visualization terminals, device models featuring different equipment and different interfaces are available.

Typical areas of application for the Rexroth IndraMotion MLC and its function modules:

- Handling and mounting systems

Important Instructions on Use

- Packaging and food machines
- Printing and paper-processing machines
- Machine tools

Rexroth IndraMotion MLC and the function modules can only be operated under the mounting and installation conditions, the position, and the ambient conditions (temperature, type of protection, moisture, EMC, etc.) specified in the relevant documentations.

2.2 Unintended Use

The use of Rexroth IndraMotion MLC and its function modules in applications other than those specified or described in the documentation and the technical data is considered as "unintended".

The Rexroth IndraMotion MLC and its function modules may not be used if they are

- exposed to operating conditions which do not correspond to the specified ambient conditions. Operation under water, extreme temperature fluctuations or extreme maximum temperatures etc. is prohibited.
- Furthermore, the IndraMotion MLC and its function modules cannot be used in any applications not expressly released by Bosch Rexroth. Therefore, please read the information specified in the general safety instructions!

3 Introduction

3.1 General

The Automation Interface provides a simple, clearly structured script interface for IndraWorks, permitting access to the IndraWorks project data.

IndraWorks automation scripts (from now on called AI scripts) can be generated in script languages such as VBScript and JavaScript and subsequently executed in the IndraWorks Engineering Desktop or in the Windows console. During the execution in IndraWorks, it is possible to integrate AI scripts or to insert them as a separate node in an IndraWorks project.

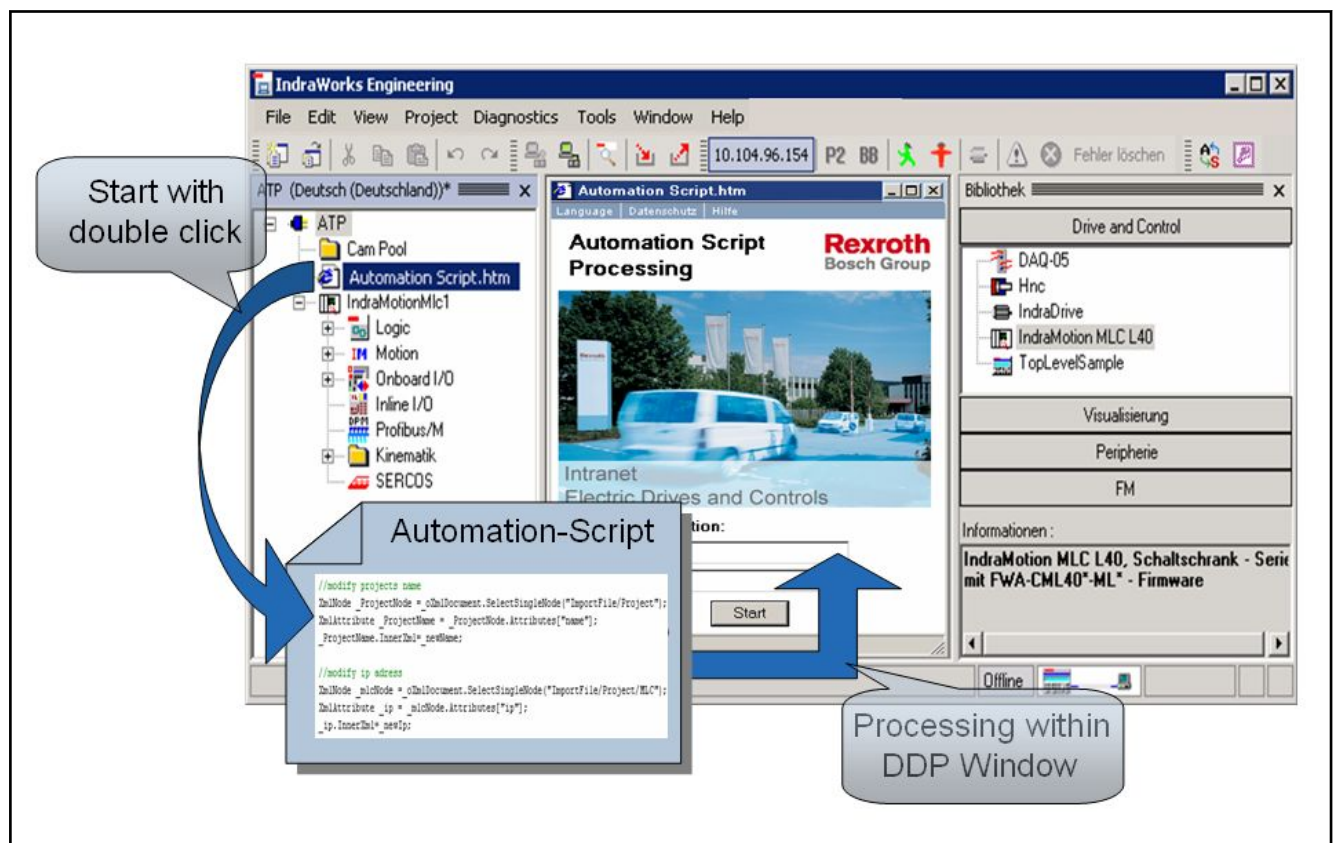


Fig.3-1: Automation scripts in IndraWorks

A listing of typical applications that can be processed using the Automation Interface is shown in the following:

- Creation of customer-specific reports.
- Automation of periodically recurring modifications in the IndraWorks projects.
- Automated creation of IndraWorks projects using the script-based processing of parts lists.
- Creation of simple customer-specific GUIs in the IndraWorks Engineering Desktop.



For the code examples described in the following chapters, there are partially requirements that are not met automatically during the installation. Thus, it can be possible that a script does not run immediately in each case if, for example, the security settings of the browser impede the creation of ActiveX objects.

Introduction

3.2 Scripting

IndraWorks Automation allows other applications to use the IndraWorks project data via the Windows OLE automation (ActiveX). The Automation Interface is normally used with script clients such as JavaScript or VBScript.

Scripts can either be executed via Microsoft Scripting Host (command line version: or integrated in the IndraWorks Engineering Desktop. Thus, an HTML page has to be created in which the automation script is embedded (for an example, refer to the following chapter). An HTML site can be subsequently added as individual menu item in the IndraWorks main menu (main menu **Tools** ▶ **External applications...** or as separate node to the IndraWorks project and started from there.

Some important comments on the scripting are listed below:

- Scripts do not have types, i.e. the script space contains only objects, but no special types (classes).
- Type resolution using LateBinding, i.e. it is not checked before runtime whether the addressed type/method can be terminated.
- When creating AI scripts, refer to this Automation Interface manual. Since scripts have no type, there is no IDE IntelliSense help.
- During scripting, all methods and attributes of the basic classes are available in a derived object; for example, the methods and attributes of **Device** and **ProjectElement** can be accessed in the **MLC** object.

3.3 Using Automation Scripts

Executing HTML files in embedded scripts

Scripts can also be used when embedded in an HTML file. The script is automatically executed if the HTML site is opened with a browser.

Example:

The following example show the structure of an HTML site with embedded JavaScript.

Program:

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

Executing script files in a console with "CScript.exe"

Scripts can be executed in a console window when using the "CScript.exe" application in Microsoft Windows. Script outputs are directly written to the console window.

Example:

After opening a console (command prompt) the JavaScript file "MyScript.js" can be executed as follows after entering "CScript.exe":

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

A helpful option to develop scripts is "//D".

```
With CScript.exe MyScript.js //D,
```

a script can be debugged. Therefore, a debugger like "Visual Studio" has to be installed on the computer executing the script. As breakpoint within a script, the keyword "debugger" is to be used. As soon as the program counter detects the keyword "debugger" when processing the script, the further pro-

Introduction

cessing of the script is canceled and the debugger to be used can be started in which the script will be subsequently processed further.

More "CScript.exe" options can be listed by calling

```
CScript.exe
```

(without arguments).

4 Basic Objects

4.1 Object Model

The object hierarchy of the Automation Interface is shown in the following diagram. In terms of structure, the diagram corresponds to an IndraWorks Project.

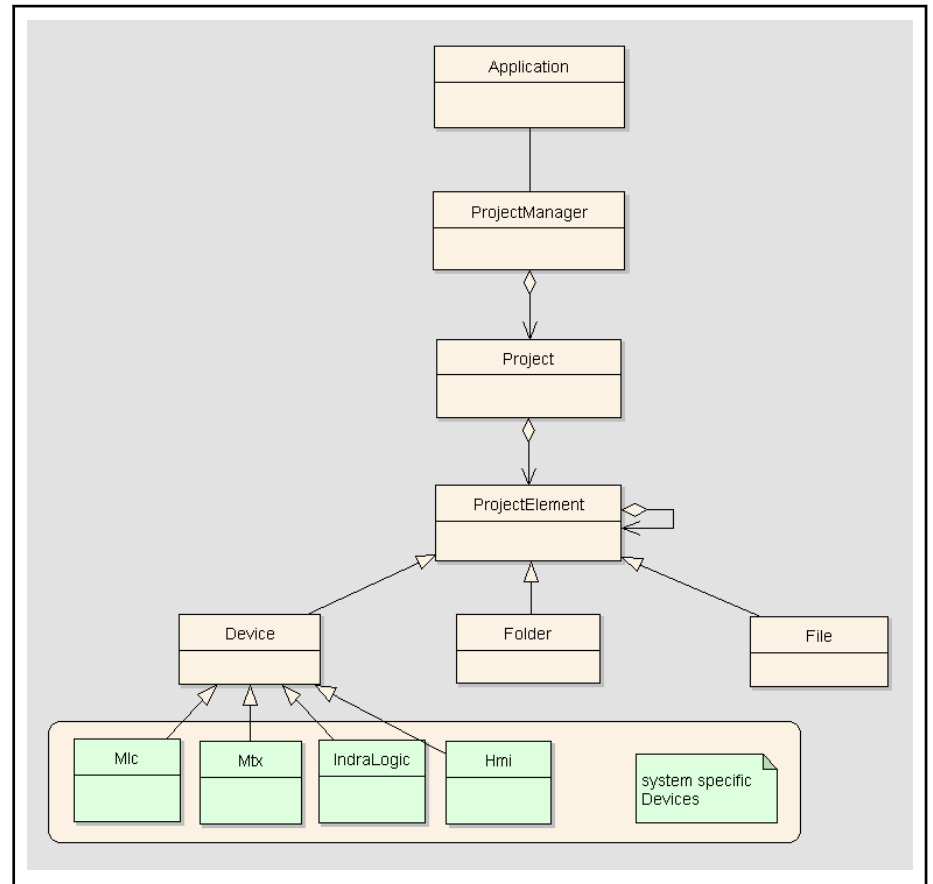


Fig.4-1: Object model of the Automation Interface to access from AI script

The object shown in beige in the diagram is the **Basis Automation Interface** described in this documentation. The objects shown in green are system-specific devices, such as HMI, MTX, MLC and IndraLogic adding system-specific aspects to the Basis Automation Interface. The description of these objects - if available for a system - is part of the respective system documentation.

4.2 Object - Application

The **Application** object is the starting object used to initialize the Automation Interface. It represents a running IndraWorks Engineering Desktop instance. When an AI script is executed in IndraWorks, the object is already created by IndraWorks and can be read from the property **window.external** in the script. After the initialization is completed, IndraWorks project data can be accessed. Apart from the access via **window.external**, the **Application** object can also be initialized via the **Rexroth.IndraWorks.Automation.Application** namespace. In this case, a security query of the Internet Explorer appears when the script is executed.

Basic Objects

Attributes of "Application"

Attribute	Type	Description
ProjectManager	Rexroth.Indraworks.Automation.ProjectManager	Read-only. Reference to ProjectManager.

Application methods

Method	Type	Description
Connect (string dest-ComputerName, string mode)	void	<p>Initializes the Automation Interface and establishes a connection to the Automation Server.</p> <p>Parameters: destComputerName [string in], target computer name on that the Automation Server - to which a connection should be established - is located. Currently, only local connections are supported (destComputerName = "localhost").</p> <p>Parameters: mode [string in], dtm / automation / tci.</p> <p>Currently, only the mode = "automation" is supported.</p>
GetService(string serviceName)	object	<p>This method provides access to the services registered at the AI.</p> <p>When it is executed successfully, the method returns a reference to the new instance of the queried service. "Zero" is returned in case of error. The service description, that is the interface description of the respective service is provided by the service provider (e.g. MTX, MLC, IndraLogic, Drive...).</p> <p>Parameters: serviceName [string in], unique name of the service registered at the AI. An access object should be returned for this service.</p>
Disconnect()	void	Disconnects the connection to the Automation Server. The Engineering Desktop is not shut down.

Basic Objects

Method	Type	Description
Shutdown()	void	Shuts down the IndraWorks Engineering Desktop to which a connection is established. All active procedures are canceled in the Engineering Desktop.
CreateComInstance (string progId)	object	<p>This method can be used to create COM objects that cannot be created in the scripting using "new ActiveXObject(..)" due to the security settings in the Internet Explorer.</p> <p>If, for example, a file access is required via the FileSystemObject in the Automation Script, the FileSystemObject is created with the following call in the script:</p> <pre>var fso = new ActiveXObject("Scripting.FileSystemObject")</pre> <p>However, this is only possible if the corresponding ActiveX privileges are set in the browser security settings. If the security settings do not allow new objects to be created using "new ActiveXObject(..)", the required FileSystemObject can be generated by calling:</p> <pre>var fso = Application.CreateComInstance("Scripting.FileSystemObject");</pre> <p>via the Automation Interface. As a result, a file can be accessed even if it would normally not be allowed due to the security settings.</p> <p>When it is executed successfully, the method returns a reference to the new instance. If an error occurs, a COMException with the HRESULT code: 0x800A01AD ("Can't create ActiveX object!") is triggered.</p> <p>Parameters: progId [string in]: ProgId of the registered COM class that is to be instanced by this call.</p> <p>Example: The ProgId for FileSystemObject is "Scripting.FileSystemObject".</p>

For an example code, please refer to: [Chapter 5.2 Example Code - Creating an IndraWorks Project, page 29.](#)

4.3 Object - ProjectManager

The **ProjectManager** object can be used to manage, i.e. create, open, delete, archive or restore, the various projects. While creating and opening, a reference to a **Project** object is provided.

The **Projects** attribute can be used to read a collection of all opened projects.

Attributes of "ProjectManager"

Attribute	Type	Description
Projects	Rexroth.IndraWorks.Automation.AiCollection	<p>Read-only. List of all opened projects. The list of type "AiCollection" contains the references to "Project" objects.</p> <p>In the current IndraWorks implementation, only one project can be opened.</p>

Basic Objects

ProjectManager methods - Part 1

Method	Type	Description
CreateProject(string fileName)	Rexroth.IndraWorks.Automation.Project	Creates a new project with the specified name and returns a reference to it. Parameters: fileName [string in], path and name of the project file (*.iwp file), contains drive and directory name.
OpenProject(string fileName)	Rexroth.IndraWorks.Automation.Project	Opens an existing project with the specified name and returns a reference to it. Parameters: fileName [string in], path and name of the project file (*.iwp file), contains drive and directory name.
DeleteProject(Rexroth.IndraWorks.Automation.Project iProject)	void	Deletes the specified project.
ArchiveProject(Scripting.Dictionary archiveArgs)	void Exception in case of error.	Creates a ZIP archive of an IndraWorks project. Only the data located in the project directory is zipped. If necessary, system-specific archiving procedures have to be executed before. Only closed projects can be archived. While archiving an open project, it is to be closed using the 'CloseProject' (True/False) key. Parameters: archiveArgs [System.Dictionary, in], all parameters for the archiving as key/value pairs in a Scripting.Dictionary object are transferred. The following keys (parameters) are available for archiving. Key: 'ProjectFile' [string in], Value: [string in] project file (*.iwp) of the project to be archived (including path name). Key: 'ArchiveFile' [string in], Value: [string in] name of the archive file (*.zip) that is to be created during archiving (including path information). Key: 'ArchiveComment' [string in], Value: [string in] [optional parameter], comment on the archive. Key: 'Password' [string in], Value: [string in] [optional parameter], password used to protect the archive. If this key is not specified or if the password is an empty string, the archive is not protected. Key: 'CloseProject' [string in], Value: [string in, True or False], this parameter can be used to specify whether the project to be archived is automatically closed before the archiving. The parameter is especially reasonable if the automatic archiving is time-controlled using AI script while the user is editing the IndraWorks project. If "CloseProject" is set to "False", the AI cannot close the project automatically while it is edited in the interface. True: The project is closed during archiving and then reopened. False: The project is not closed. Caution: An open project can only be archived if this parameter is set to "True". For an example code, please refer to: Chapter 5.7 Example Code - Archiving a Project, page 34.

ProjectManager methods - Part 2

Method	Type	Description
RestoreProject(Scripting.Dictionary restoreArgs)	void Exception in case of error.	Restores a previously created IndraWorks project archive. The archive is unzipped and the IndraWorks project is restored. Parameters: restoreArgs [System.Dictionary, in], all parameters for the restoration are transferred as key/value pairs in a Scripting.Dictionary object. The following keys (parameters) are available for restoration. Key: 'ArchiveFile' [string in], Value: [string in] file of the IndraWorks project archive (*.zip) that is to be restored (including path information). Key: 'RestoreDir' [string in], Value: [string in], target directory for restoration. Specifies the directory in which the IndraWorks project archive is to be restored. The project directory is created below this directory while restoring it. Key: 'NewProjectName' [string in], Value: [string in] [optional parameter], if the project to be restored should get a new name, it can be specified here. In this case, project directory, project file and project name are changed accordingly during restoration. Key: 'Password' [string in], Value: [string in] [optional parameter], password used to protect the archive. If this key is not specified or if the password is an empty string, it is tried to restore the archive without a password. Key: 'OpenProject' [string in], Value: [string in, True or False], Boolean variable determining whether the restored project should be opened after restoration. True: The project is automatically opened after the restoration (the currently open project is closed before). False: The project is not reopened after the restoration. For an example code, please refer to: Chapter 5.8 Example Code - Restoring a Project, page 35 .

For an example code, please refer to: [Chapter 5.2 Example Code - Creating an IndraWorks Project, page 29](#).

4.4 Object - Project

The **Project** object represents an IndraWorks project. The project methods can be used to create new subordinate project elements and search for project elements or delete them. Project data can be exported/imported via **Export/Import**.

If the project contains system-specific devices such as MTX, MLC or IndraLogic, the **GetChildrenOfType** method and the corresponding type string can be used to determine a reference to these objects. This reference can be used to access the extended functionality of the system-specific devices.

The **Children** attribute provides a collection of all project elements of the project (all nodes of the first level below the project node).

Attributes of "Project"

Attribute	Type	Description
Name	string	Read/write. Project name.
Id	string	Read-only. Unique project ID (guid).

Basic Objects

Attribute	Type	Description
Location	string	Read-only. Directory path in which the project (project file) is saved.
Children	Rexroth.IndraWorks.Automation.AiCollection	Read-only. Provides a collection of all children (all nodes of the first level below the project node). The collection contains references to objects of the type Rexroth.IndraWorks.Automation.Device, Rexroth.IndraWorks.Automation.File, type Rexroth.IndraWorks.Automation.Folder or types of system-specific devices.

Project methods

Method	Type	Description
Save()	void	Saves the project.
SaveAs(string fileName)	Rexroth.IndraWorks.Automation.Project	Saves the current project under a new name and returns a reference to the new project. Caution: The previous "Project" object (with the old name) becomes invalid after the call! Parameters: fileName [string in], name of the new project file (*.iwp) containing drive and directory name.
Close()	void	Closes the project.
Export(string fileName)	void Exception in case of error.	The project is exported via this method. Parameters: fileName [string in], file name (including path information) of the export file (*.iwx) created during export.
Import(string fileName, string mode)	void Exception in case of error.	This method is used to re-import the data of a project created earlier with "Project.Export". Parameters: fileName [string in], file name (including path information) of the export file (*.iwx or *.xml for the previous format) whose data is to be imported. Parameters: mode [string in, mode = "Insert" or "Replace"], import mode indicates whether the data to be imported should be inserted ("Insert") or whether already existing data should be replaced ("Replace"). The parameter is not evaluated for the previous export format (*.xml). Please set the parameter to "None".

Basic Objects

Method	Type	Description
CreateChild(string type, object args)	Rexroth.IndraWorks.Automation.ProjectElement	<p>This method can be used to create a ProjectElement either of type "file" or "folder" or a system-specific device as child of the project node. Proceed as follows to create a new ProjectElement:</p> <p>Creating a file: type = "File". args = Complete file name to be added as string. If zero or an empty string is transferred, a file selection dialog appears in the Engineering Desktop that can be used to select a file to be added.</p> <p>Creating a folder: type = "Folder" args = Name of the new folder as string. If zero or an empty string is transferred, the name of the new folder is assigned automatically.</p> <p>Creating a system-specific device: type = <Device type identifier (string) of the device to be newly created> args = Use the AI documentation of the corresponding device to check how this parameter is to be assigned.</p>
DeleteChild(string id)	bool	<p>Deletes a child project element with the given ID ("MnO Guid" is the internal identifier for this ID in IndraWorks).</p> <p>Parameters: id [string in]: ID (guid) of the child project element to be deleted.</p>
GetChildrenOfType(string objectType, bool recurse)	Rexroth.IndraWorks.Automation.AiCollection	<p>Provides a collection of all project elements of the type specified. The collection contains references of the types Rexroth.IndraWorks.Automation.IDevice, Rexroth.IndraWorks.Automation.IFile, typeRexroth.IndraWorks.Automation.IFolder or types of system-specific devices.</p> <p>Parameters: objectType [string in], type string of the project elements searched for. The basic types are "File", "Folder" and "Device". If system-specific devices such as MTX, MLC, IndraLogic or HMI exist, they can be determined using the respective type string.</p> <p>Parameters: recurse [bool in],</p> <p>True: All project elements are taken into account during the search False: Only the project elements located on the first level below the project node are taken into account during the search.</p> <p>For an example code, please refer to: Chapter 5.5 Example Code - Searching in Projects, page 31.</p>

Basic Objects

Method	Type	Description
GetElement(string id)	Rexroth.IndraWorks.Automation.ProjectElement	Provides the reference to the project element with the given ID ("MnO Guid" is the internal identifier in IndraWorks for this ID). Parameters: id [string in]: ID (guid) of the project element searched for.
Validate(Scripting.Dictionary validateArgs)	string	The project data is subject to a plausibility check. Obvious discrepancies can thus be detected and eliminated. Important: This check cannot verify the correctness of all project data! The feedback about the check result is given as string in XML format. The nodes "Errors", "Warnings" und "Information" return the number of the respective messages. If "Cancelled" is equal to 'True', the check was aborted. The messages found are stored under "Messages". A message is described by its attributes. They corresponds to the columns of the message window in the IndraWorks Engineering Desktop: Type - Message type "Error", "Warning", "Info" Id - Message ID Text - Message text Element - Name of the project element Position - Optional: e.g. line and column specification if the project element is a source text Path - Relative project path Parameters: validateArgs [System.Dictionary, in], in a Scripting.Dictionary, key/value pairs can be transferred as optional parameters. An empty table corresponds to a validation function as it can be started at the project node via the context menu item. If special function should be checked, they can be controlled via special key/value parameters. In this case, the names of the keys and the meaning of the respective values of the documentations (System Description) can be found in the respective project elements. For an example code, please refer to: Chapter 5.9 Example Code - Validating a Project, page 36 .

```
<?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>
```

Fig.4-2: Project.Validate(..): Return format of the check results

4.5 Object - ProjectElement

ProjectElement is a basic class of the Automation Interface. This basic class is used by the objects **Device**, **File** and **Folder** as well as by all system-specific devices such as MLC, MTX and IndraLogic. This means that these objects can be used to access all attributes of the basic class **ProjectElement**.

The basic class **ProjectElement** provides a series of attributes that can be used to navigate through the project.

If there are system-specific devices such as MTX, MLC or IndraLogic under the current ProjectElement, the **GetChildrenOfType** method and the respective type string can be used to determine a reference to these objects. This

Basic Objects

reference can be used to access the extended functionality of the system-specific devices.

Child ProjectElements can be created/deleted using the **CreateChild** and **DeleteChild** methods.

The **Export** method can export all ProjectElement data including the data of all child elements (if these support an export) to an export file.

The **Import** method can re-import the data from an export file to the Project-Element or its child elements.

The **SupportsExportImport** attribute can be used to query whether the export/import is supported by this ProjectElement.

The **Parent** attribute can be used to determine the parent node. The attribute **Children** provides a collection of all child project elements of the first level and the assigned project can be determined via **OwnerProject**.

Attributes of "ProjectElement"

Attributes	Type	Description
Name	string	Read/write. Project element name (display name).
Id	string	Read-only. Unique project element ID (guid).
Location	string	Read-only. Directory path under which the project element is saved.
OwnerProject	Rexroth.IndraWorks.Automation.Project	Read-only. Reference to the project to which the project element is assigned.
Parent	Rexroth.IndraWorks.Automation.ProjectElement	Read-only. Reference to the parent project element. Zero is returned if the parent element is the project node.
Children	Rexroth.IndraWorks.Automation.AiCollection	Read-only. Provides a collection of all children of the first level. The collection contains references of the types Rexroth.IndraWorks.Automation.IDevice, Rexroth.IndraWorks.Automation.IFile, typeRexroth.IndraWorks.Automation.IFolder or types of system-specific devices.
SupportsExportImport	bool	Read-only. Provides information on whether the ProjectElement supports the "Export" and "Import" functionalities. True: Export/Import is supported. False: Export/Import is not supported.
SupportsValidate	bool	Read-only. Provides information on whether the project element supports the "Validate" functionality. True: "Validate" is supported. False: "Validate" is not supported.

Basic Objects

ProjectElement methods

Method	Type	Description
Export(string fileName)	void Exception in case of error.	The project element data and all subordinate project elements supporting an export are exported using this method. Caution: This method is not supported by each ProjectElement! Check before whether this method is supported using the "ProjectElement.SupportsExportImport" attribute. Parameters: fileName [string in], file name (including path information) of the export file (*.iwx) created during export.
Import(string fileName, string mode)	void Exception in case of error.	This method re-imports export file data that was created earlier with "ProjectElement.Export". Caution: This method is not supported by each ProjectElement! Check before whether this method is supported using the "ProjectElement.SupportsExportImport" attribute. Parameters: fileName [string in], file name (including path information) of the export file (*.iwx) whose data should be imported. Parameters: mode [string in, mode = "Insert" or "Replace"], import mode indicates whether the data to be imported should be inserted ("Insert") or whether already existing data should be replaced ("Replace").
CreateChild(string type, object args)	Rexroth.IndraWorks.Automation.ProjectElement	This method can be used to create a ProjectElement of the types "File" or "Folder" or a system-specific device as child of the current ProjectElement. Proceed as follows to create a new ProjectElement: Creating a file: type = "File". args = Complete file name to be added as string. If zero or an empty string is transferred, a file selection dialog appears in the Engineering Desktop that can be used to select a file to be added. Creating a folder: type = "Folder" args = Name of the new folder as string. If zero or an empty string is transferred, the name of the new folder is assigned automatically. Creating a system-specific device: type = <Device type identifier (string) of the device to be newly created> args = Use the AI documentation of the corresponding device to check how this parameter is to be assigned.
DeleteChild(string id)	bool	Deletes a child project element with the given ID ("MnO Guid" is the internal identifier for this ID in IndraWorks). Parameters: id [string in]: ID (guid) of the child project element to be deleted.

Basic Objects

Method	Type	Description
GetChildrenOf- Type(string object- Type, bool recurse)	Rexroth.IndraWorks.A utomation.AiCollection	Provides a collection of all project elements of the type specified. The collection contains references of the types Rexroth.IndraWorks.Automation.IDevice, Rexroth.IndraWorks.Automation.IFile, typeRexroth.IndraWorks.Automation.IFolder or types of system-specific devices. Parameters: objectType [string in], type string of the project elements searched for. The basic types are "File", "Folder" and "Device". If system-specific devices such as MTX, MLC, IndraLogic or HMI exist, they can be determined using the respective type string. Parameters: recurse [bool in], True: All project elements are taken into account during the search False: Only the project elements located on the first level below the project node are taken into account during the search.
Validate(Scripting.Dictionary validateArgs)	string	The project element data and all subordinate project elements are subject to a plausibility check. Obvious discrepancies can thus be detected and eliminated. Important: This check cannot verify the correctness of all project data! Call parameter of the return value: see object "Project", method "Validate" Caution: This method is not supported by each ProjectElement! Check before if this method is supported using the "ProjectElement.SupportsValidate" attribute.

4.6 Object - Device (Derived from ProjectElement)

The **Device** object represents a project element of the **Device** type, i.e. a device in an IndraWorks project.

The methods of the object can be used to go online or offline with the device. The "IsOnline" attribute can be used to query whether the device is currently in online mode.

In addition, all attributes and methods of the basic class **ProjectElement** are available in the **Device** context.

Attributes of "Device"

Attributes	Type	Description
IsOnline	bool	Read-only. Provides information on whether the device is online or offline. Provides "true" if the device is online and "false" if it is offline.

Methods of "Device"

Method	Type	Description
GoOnline()	void	Switches the device to online mode. Use the device-specific AI documentation to check whether this function is available for the device.
GoOffline()	void	Switches the device to offline mode. Use the device-specific AI documentation to check whether this function is available for the device.

For an example code, please refer to: [Chapter 5.6 Example Code - Access to Devices, page 33](#).

Basic Objects

4.7 Object - File (Derived from ProjectElement)

The **File** object represents a project element of the **File** type, i.e. a file in an IndraWorks project.

The **FileName** attribute can be used to read the file name.

In addition, all attributes and methods of the basic class **ProjectElement** are available in the **File** context.

Attributes of "File"

Attributes	Type	Description
FileName	string	Read-only. File name.

For an example code, please refer to: [5.4 Example Code - Adding Folders and Files, page 30](#).

4.8 Object - Folder (Derived from ProjectElement)

The **Folder** object represents a project element of the **Folder** type, i.e. a directory in an IndraWorks project.

In addition, all attributes and methods of the basic class **ProjectElement** are available in the **File** context.

For an example code, please refer to: Chapter [5.4 Example Code - Adding Folders and Files, page 30](#).

4.9 Object - AiCollection

The **AiCollection** object is an unsorted list of objects that can be read according to the following attributes and methods.

Attributes of "AiCollection"

Attributes	Type	Description
Count	int	Read-only. Provides the number of objects.
Current	object	Read-only. Provides the currently selected object.
this[int]	object	Read-only. Indexer. Parameters: Index [int in], index (basis 0) in the collection.

Methods of "AiCollection"

Method	Type	Description
GetEnumerator()	System.Collections.IE-numerator	Provides an enumerator.
Contains(object item)	bool	Checks whether the specified object is located in the collection. True: The object searched for is located in the collection. False: The object is not located in the collection.

For an example code, please refer to: Chapter [5.5 Example Code - Searching in Projects, page 31](#).

4.10 Object - AiList

The **AiList** object is an unsorted list of objects that can be read according to the following attributes and methods. In contrast to the "AiCollection", the "Ai-

Basic Objects

List" object allows objects to be added to the list or to be removed from it using the methods **Add()** and **Remove()**.

Attributes of "AiList"

Attributes	Type	Description
Count	int	Read-only. Provides the number of objects.
Current	object	Read-only. Provides the currently selected object.
this[int]	object	Read-only. Indexer. Parameters: Index [int in], index (basis 0) in the collection.

Methods of "AiList"

Method	Type	Description
Add(object value)	int	Adds a new object to the list and returns the index at which the object was inserted.
Remove(object value)	void	Removes the first object that is identical to the object specified in the list.
GetEnumerator()	System.Collections.IE-numerator	Provides an enumerator.
Contains(object item)	bool	Checks whether the specified object is located in the collection. True: The object searched for is located in the collection. False: The object is not located in the collection.

5 Example Codes - Basic Objects

5.1 Example Code - Initializing the Automation Interface

AI initialization

Program:

```
<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 Example Code - Creating an IndraWorks Project

Initializing the AI and creating a project

Program:

```
<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)
```

Example Codes - Basic Objects

```

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

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

```

5.3 Example Code - Opening an IndraWorks Project

Initializing the AI and opening a project

Program:

```

<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 Example Code - Adding Folders and Files

Creating folders and files in a project

Program:

```

<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");
    }

```

Example Codes - Basic Objects

```

//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 Example Code - Searching in Projects

Searching in a project

Program:

```

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

```

Example Codes - Basic Objects

```

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();

```

Example Codes - Basic Objects

```

        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 Example Code - Accessing Devices

Accessing devices (= device-specific AI extension)

Program:

```

<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");
        }

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

```

Example Codes - Basic Objects

```

Children = CurrentProject.ChildrenOfType("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 Example Code - Archiving a Project

Creating an IndraWorks project archive (*.zip)

Program:

```

<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);
    }
}
}
</script>
</body>
</html>

```

```

    }
    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 Example Code - Restoring a Project

Restoring an IndraWorks project from a project archive (*.zip)

Program:

```

<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);
    }
}

```

Example Codes - Basic Objects

```

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

```

5.9 Example Code - Validating a Project

Executing a default validation for a project with subsequent result output

Program:

```

<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 thisMnODoesntSupportValidate;
        for(var mnos = new Enumerator(CurrentProject.Children); !mnos.atEnd(); mnos.moveNext())
        {
          var mno = mnos.item();
          document.write("'" + mno.Name + "': ");
          if (mno.SupportsValidate == false)
          {
            thisMnODoesntSupportValidate = 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;
        }
      }
    </script>
  </body>
</html>

```

Example Codes - Basic Objects

```

        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 (thisMnODoesntSupportValidate != null)
        {
            document.write(
                "<br><b>Validation of a project element that doesn't support validation =====</b><br>");
            result = thisMnODoesntSupportValidate.Validate(args); //Start validation
        }
    }
    catch (exception)
    {
        //catch InvalidOperationException
        document.write("Expected InvalidOperationException:<br>");
        document.write("Error code=" + exception.number + "<br>");
        document.write("Message=" + exception.message);
    }
}
catch (e)
{
    document.write("<br><br>!!!!!! Exception occured in JavaScript !!!!!!!:<br>");
    document.write("Error name=" + e.name + "<br>");
    document.write("Error code=" + e.number + "<br>");
    document.write("Error message=" + e.message + "<br>");

    //Disconnect from Automation Server
    Application.Disconnect();
}

```

Example Codes - Basic Objects

```
    }  
  </Script>  
</Body>  
</html>
```

6 XLC/MLC Objects

6.1 Object Model

The object hierarchy of the Automation Interface for the XLC/MLC objects is shown in the following diagram.

The objects shown in the diagram are system-specific XLC/MLC objects, the **Device** object shown in beige is part of the **Basis Automation Interface**.

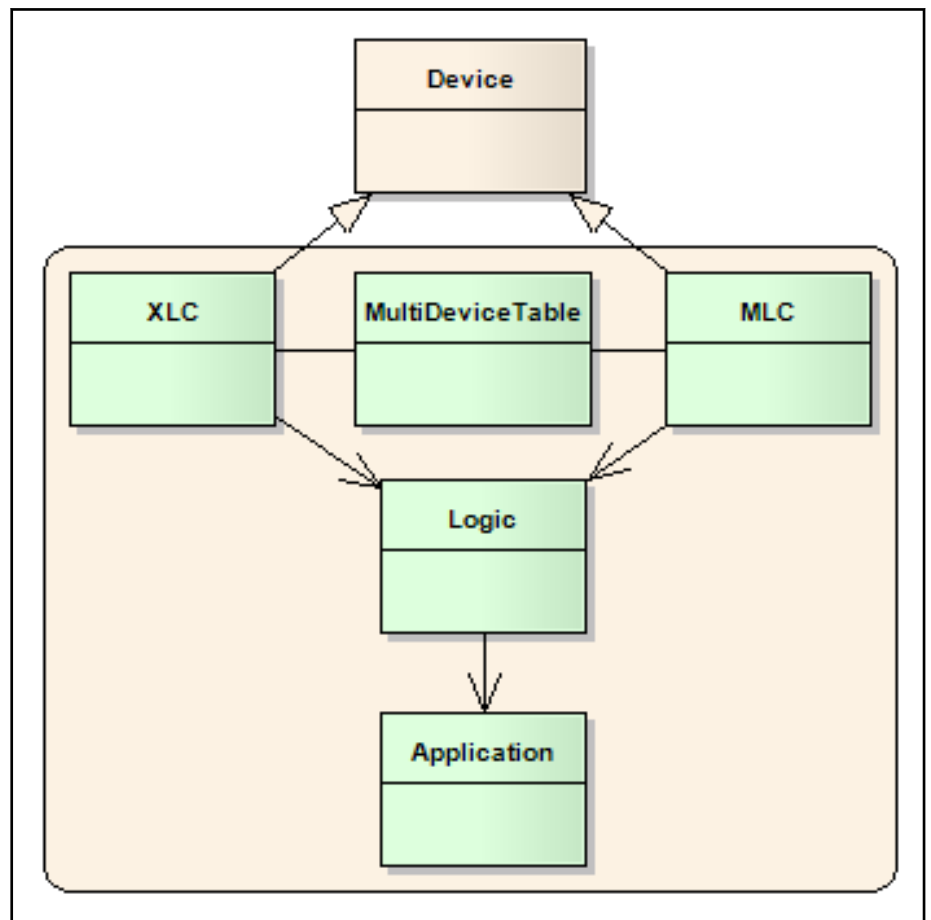


Fig.6-1: Object model: "Object Device" of the Basis Automation Interfaces and the assigned XLC/MLC objects

6.2 Object - XLC (Derived from Device)

The **XLC** object represents a device of the **XLC** type, i.e. a system-specific device in an IndraWorks project.

The object **XLC** is derived from the **Device** object. Therefore, it provides all the attributes and methods of this object. This includes for example the attribute **IsOnline** as well as the methods **GoOnline** and **GoOffline**.

The **Logic** object below the **XLC** can be accessed using **GetObject**.

XLC/MLC Objects

Attributes of XLC

Attribute	Type	Description
IpAddress	string	Allows reading and writing the IP address of the control.
IsMultiDeviceEnabled	bool	Read-only. Returns the information on whether the multi-device functionality is used. True: Multi-device is used. False: Multi-device is not used.
IsOnline	bool	Read-only, inherited from "Device". True: XLC in the IndraWorks project is connected to a physical XLC. False: No connection between the XLC in the IndraWorks project and the physical XLC.
MultiDeviceTable	object	Read-only. Returns a reference to the XLC-internal object "MultiDeviceTable".

Methods of XLC

Method	Type	Description
ClearErrors()	Void	Deletes all errors on the control and in the connected sercos drives. Therefore, the XLC has to be in "online" state.
DisableMultiDevice()	Void	Do not use the multi-device functionality. If the multi-device functionality cannot be disabled, an exception is thrown.
DownloadFirmware(string fileName)	Void	Loads the firmware in the "fileName" file to the control. Therefore, the XLC has to be in download mode (sercos phase P0) and the PLC has to be stopped. Parameters: fileName [string], contains the complete name and path of a firmware file of type "FW". Example: "C:\FW\CML65s-XLs-12T01.0040.fw"
EnableMultiDevice()	Void	Using multi-device functionality. The XLC may neither be in "online" state nor in "logged in" state. If the multi-device functionality cannot be used, since the XLC is in "online" state and/or "logged in", an exception is thrown.
GetObject(string object)	object	Returns a reference to an object assigned to an XLC. If the requested object is not known, an exception is thrown. Parameters: Object [string], string of the searched object. Currently, only "Logic" is available as object.
GetPhase()	int	Returns the current sercos phase. Therefore, the XLC has to be in "online" state. If the phase cannot be read, since the XLC is not in "online" state for example, an exception is thrown.
GoOffline()	Void	Inherited from "Device". Cancels a connection between the XLC in the IndraWorks project and a physical XLC. If it cannot be gone offline, an exception is thrown.

XLC/MLC Objects

Method	Type	Description
GoOnline()	Void	<p>Inherited from "Device".</p> <p>Establishes a connection between the XLC in the IndraWorks project and a physical XLC.</p> <p>If the online connection cannot be established, since the PLC has to be stopped or the planned drive configuration does not match, the established connection is aborted and an exception is thrown.</p>
GoOnline2(Scripting.Dictionary args)	Void	<p>Allows the user to specifically control the behavior when going online.</p> <p>Parameters:</p> <p>args [System.Dictionary], all parameters to go online are transferred as key/value pairs in a System.Dictionary object. The following key/value pairs are available. If the specifications are invalid or if going online is not possible, an exception is thrown.</p> <p>Key: "AdjustC2CMotionSercosCycleTimes" [string] [Optional key/value pair]</p> <p>Value: [bool], indicates whether - when using a C2C cross communication module (plug-in module to communicate between controls) - the cycle times for the C2C communication, for the Motion (Motion cycle time) and for the communication with sercos devices (cycle time of the master communication) are to be adjusted if the cycle times differ.</p> <p>True: When using the C2C cross communication module, the cycle times for the Motion and for the master communication (if unequal to the cycle time for the C2C cross communication) are set to the cycle time value for the C2C cross communication. This is required to switch the sercos ring to the operating mode (sercos phase 4).</p> <p>False (default value): When using the C2C cross communication module, the cycle times for the Motion and for the master communication (if unequal to the cycle time for the C2C cross communication) are not changed. In this case (cycle time of the C2C cross communication not equal to the cycle times for the Motion and the master communication), it cannot be switched to the operating mode (sercos phase 4) when using a C2C cross communication module.</p> <p>Example (JavaScript):</p> <pre>args.Add("AdjustC2CMotionSercosCycleTimes", true);</pre> <p>Key: "CanOverwriteProject" [string] [optional key/value pair]</p> <p>Value: [bool], indicates whether a third-party project in the control may be overwritten.</p> <p>True (default value): A third-party project in the control is overwritten when going online.</p> <p>False: A third-party project in the control is not overwritten when going online. Going online is aborted in this case.</p> <p>Example (JavaScript):</p> <pre>args.Add("CanOverwriteProject", false);</pre> <p>Key: "CanStopPlcWhenAxesAreNotActive" [string] [optional key/value pair]</p> <p>Value: [bool], indicates whether the PLC may be stopped if the axes are not active.</p> <p>True: If the axis configuration was changed in IndraWorks, if the PLC is running ("RUN" state) and if no axis is active, the PLC is stopped when going online. If at least one axis is active (real axis "Power on", virtual axis not in "Standstill"), the PLC cannot be stopped. In this case (PLC cannot be stopped), going online is aborted and an exception is thrown.</p> <p>False (default value): If the axis configuration was changed in IndraWorks and if the PLC is running ("RUN" state), the PLC is not stopped when going online. This applies as well if no axis is active. In this case, going online is aborted and an exception is thrown.</p>

XLC/MLC Objects

Method	Type	Description
		<p>Example (JavaScript):</p> <pre>args.Add("CanStopPlcWhenAxesAreNotActive", true);</pre> <p>Key: "CanIgnoreSercosDeviceInconsistencies" [string] [optional key/value pair]</p> <p>Value: [bool], indicates whether a mismatch between the planned sercos devices and the actually existing devices at the sercos bus is to be ignored</p> <p>True (default value): Even if it is determined that the devices connected at the sercos bus do not match the configured sercos devices, it is still tried to go online. If this is not possible, going online is aborted and an exception is thrown.</p> <p>False: Even if it is determined that the devices connected at the sercos bus do not match the planned sercos devices, going online is aborted and an exception is thrown.</p> <p>Example (JavaScript):</p> <pre>args.Add("CanIgnoreSercosDeviceInconsistencies", false);</pre>
ImportParameter(string fileName)	string	<p>Imports the sercos parameters to the "fileName" file. Therefore, the XLC has to be in "online" state.</p> <p>Possible writing errors such as "date and time write-protected" are collected by the method and returned as string at the end of the parameter import. The line format is "IDN ParameterName Result".</p> <p>Parameters:</p> <p>fileName [string], contains the complete name and path of the file to be imported of type "PAR". Example: "C:\Archive\Parameters.par"</p>

Method	Type	Description
ImportParameter2(string fileName, Scripting.Dictionary args)	string	<p>Imports the sercos parameters to the "fileName" file. Therefore, the XLC has to be in "online" state. In contrast to "ImportParameter", it has to be specified which parameter types are to be imported to which axes (or controls).</p> <p>Possible writing errors such as "date and time write-protected" are collected by the method and returned as string at the end of the parameter import. The line format is "IDN ParameterName Result".</p> <p>Parameters:</p> <p>fileName [string], contains the complete name and path of the file to be imported of type "PAR". Example: "C:\Archive\Parameters.par".</p> <p>args [System.Dictionary]</p> <p>All parameter specifications on the parameter import are transferred to a System.Dictionary object. The following keys and values are available:</p> <p>Key: "C-ParameterImportListByNumber" [string] [optional key/value pair], control parameter.</p> <p>Value: [string] indicates whether C-parameters are to be loaded (default).</p> <p>Contains the parameter file "C-parameters". If the C-parameters are to be imported, the string "*->*" has to be specified. If the key "C-ParameterImportListByNumber" is not specified or an empty string "" is specified (default value), the C-parameters are not loaded. If the parameter file does not contain any C-parameters, the specification is ignored.</p> <p>Example (JavaScript):</p> <pre>args.Add("C-ParameterImportListByNumber", "*->*");</pre> <p>Loads the C-parameters from the parameter file to the control.</p> <p>Example (JavaScript):</p> <pre>args.Add("C-ParameterImportListByNumber", "");</pre> <p>This causes that the C-parameters are not loaded to the parameter file.</p> <p>Key: "M-ParameterImportListByNumber" [string] [optional key/value pair], touch probe parameter.</p> <p>Value: [string] indicates whether M-parameters are to be loaded (default).</p> <p>Same behavior as "C-ParameterImportListByNumber"</p> <p>Key: "O-ParameterImportListByNumber" [string] [optional key/value pair], oscilloscope parameter.</p> <p>Value: [string] indicates whether O-parameters are to be loaded (default).</p> <p>Same behavior as "C-ParameterImportListByNumber"</p> <p>Key: "A-ParameterImportListByNumber" [string] [optional key/value pair], drive-related parameters in the control.</p> <p>Value: [string] indicates whether A-parameters are to be loaded (default) and to which axes.</p> <p>Contains the parameter file "A-parameters" and if the A-parameters are to be loaded, the axis numbers to which the A-parameters have to be loaded can be specified with a comma-separated list. Ranges can also be defined with "-". A-parameters are not loaded with "" (default value). If the key "A-ParameterImportListByNumber" is not specified, the A-parameters in the parameter file are not loaded. If the parameter file does not contain any A-parameters, the specification is ignored.</p>

XLC/MLC Objects

Method	Type	Description
		<p>Example (JavaScript):</p> <pre>args.Add("A-ParameterImportListByNumber", "1->1; 2->2,3-6,8");</pre> <p>Loads the A-parameters from the parameter file for the axis with the number 1 to the configured axis with the number 1. The A-parameters in the parameter file for the axis with the number 2 are loaded to the configured axes with the numbers 2, 3, 4, 5, 6 and 8.</p> <p>Example (JavaScript):</p> <pre>args.Add("A-ParameterImportListByNumber", "*->*");</pre> <p>Loads the A-parameters from the parameter file to the axes specified in the parameter file.</p> <p>Example (JavaScript):</p> <pre>args.Add("A-ParameterImportListByNumber", "");</pre> <p>This causes that the A-parameters are not loaded to the parameter file.</p> <p>Example (JavaScript):</p> <pre>// nothing</pre> <p>If the key "A-ParameterImportListByNumber" is not specified, the A-parameters in the parameter file are not loaded.</p> <p>Key: "SP-ParameterImportListByNumber" [string] [optional key/value pair], drive parameters in the drive.</p> <p>Value: [string], indicates whether S/P-parameters can be loaded to the drive (default). Same behavior as "A-ParameterImportListByNumber"</p> <p>Key: "SP-ParameterImportListByAddress" [string] [optional key/value pair], drive parameters in the drive.</p> <p>Value: [string], indicates whether S/P-parameters can be loaded to the drive (default). Same behavior as "SP-ParameterImportListByNumber". Only the sercos addresses have to be specified at "SP-ParameterImportListByAddress" instead of logic axis numbers.</p>
ReadListParameter(string idn)	Rexroth .IndraWorks.Automation.AiList	<p>Reads a list parameter of the control or a sercos device (list parameter = parameter of variable length). Therefore, the MLC has to be in "online" state. The individual elements of the list parameter are returned in an "AiList" as a list of strings [string].</p> <p>Parameters:</p> <p>idn [string], ident number of the parameter, such as "C-0-0001" or "C-0-0001.0.0". For axis-related parameters (A-, S-, P-parameters) and for kinematic parameters (K-parameters), the 3-digit reference number has to be specified as well such as "A001:A-0-0001" or "A001:A-0-0001.0.0" as well as "A001:S-0-0001" or "K001:K-0-0001".</p>
ReadParameter(string idn)	string	<p>Reads a "normal" parameter of the control or a sercos device. Therefore, the MLC has to be in "online" state.</p> <p>Parameters:</p> <p>idn [string], ident number of the parameter. The same as under "ReadListParameter" applies to the spelling of the ident number.</p>

Method	Type	Description
SwitchPhase(int phase)	Void	<p>Switches to the sercos phase. Therefore, the XLC has to be in "online" state.</p> <p>If the phase cannot be switched, since an invalid phase was specified for example or the master axis is enabled, an exception is thrown.</p> <p>Parameters:</p> <p>phase [int], contains the target phase in numbers. The following target phases are allowed:</p> <ul style="list-style-type: none"> • 0 (download mode) • 2 (parameterization mode) • 4 (operating mode)
WriteListParameter(string idn, Rexroth.IndraWorks.Automation.AiList values)	Void	<p>Writes a list parameter of the control or a sercos device (list parameter = parameter of variable length). The individual elements of the list parameter have to be transferred to an "AiList" as a list of strings [string].</p> <p>Parameters:</p> <p>idn [string], ident number of the parameter. The same as under "ReadListParameter" applies to the spelling of the ident number.</p> <p>values [Rexroth.IndraWorks.Automation.AiList], contains the elements of the list parameter to be written as list of strings [string].</p>
WriteParameter(string idn, string value)	Void	<p>Writes a "normal" parameter of the control or a sercos device.</p> <p>Parameters:</p> <p>idn [string], ident number of the parameter. The same as under "ReadListParameter" applies to the spelling of the ident number.</p> <p>value [string], contains the value to be written.</p>

6.3 Adding an XLC

Using the Basis Automation Interface object **Project**, XLC devices can be added to the current project below the project node. Therefore, use the **CreateChild** method of the **Project** object.

Two function arguments have to be transferred to the **CreateChild** method: the type of the object (**XLC**) to be generated as well as a **Scripting.Dictionary** object containing the special XLC characteristics as key/value pairs.

XLC/MLC Objects

CreateChild method from the object "Project"

Method	Type	Description
CreateChild(string childLibGuid, object args)	Rexroth .IndraWorks.Automation.ProjectElement	<p>This method of the Basis Automation Interface can create a "ProjectElement" of the types "File", "Folder" or "Device" as child of the project node. To create a "ProjectElement" of the type "XLC", the following arguments have to or can be used.</p> <p>Creating an XLC:</p> <p>Parameters:</p> <p>childLibGuid [string], the object ("XLC") to be created.</p> <p>args [System.Dictionary]</p> <p>The property values of the XLC are transferred as key/value pairs to a System.Dictionary object. The following keys and values are available:</p> <p>Key: "Hardware" [string]</p> <p>Value: [string], XLC, hardware name. The following formats are supported:</p> <ul style="list-style-type: none"> • "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" <p>Only the front part is currently evaluated, such as "CML25.1-3N". If the back part is also indicated, the format has to correspond to the given format.</p>

Method	Type	Description
		<p>Key: "Firmware" [string] Value: [string] XLC, firmware name. The following firmware names are supported:</p> <ul style="list-style-type: none"> • "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" <p>Key: CMLx5 = CML25, CML45, CML65 The firmware has to "fit" the hardware. No hardware "CML25.1-3N" may be used with the firmware "FWA-CML45*-XL*-12VRS" for example (conflict L25, L45). When specifying "VRS" instead of "V02" for example, the latest firmware release is automatically used.</p> <p>Key: "Name" [string] [optional key/value pair] Value: [string], Control name. If no name is given, it is automatically specified.</p> <p>Key: "IpAddress" [string] [optional key/value pair] Value: [string], IP address of the control. If no IP address is specified, it is automatically specified.</p> <p>Key: "PlcGateway" [string] [optional key/value pair] Value: [string], IP address of the PLC gateways. If no PLC gateway is specified, "localhost" is used.</p> <p>Key: "PlcSecureOnlineMode" [string] [optional key/value pair] Value: [string], "True" if the secure online mode is to be used, otherwise "False" (default).</p> <p>Key: "Author" [string] [optional key/value pair] Value: [string], Name of the person who added the control to the project.</p> <p>Key: "Comment" [string] [optional key/value pair] Value: [string], Comment.</p>

6.4 Object - MLC (Derived from Device)

The **MLC** object represents a device of the **MLC** type, i.e. a system-specific device in an IndraWorks project. Its interface is identical to the **XLC** object. That means that it is provided with the same attributes and methods.

XLC/MLC Objects

Attributes of MLC

Attribute	Type	Description
IpAddress	string	Allows reading and writing the IP address of the control.
IsMultiDeviceEnabled	bool	Read-only. Returns the information on whether the multi-device functionality is used. True: Multi-device is used. False: Multi-device is not used.
IsOnline	bool	Read-only, inherited from "Device". True: MLC in the IndraWorks project connected to the physical MLC. False: No connection between MLC in the IndraWorks project and the physical MLC.
MultiDeviceTable	object	Read-only. Returns a reference to the MLC-internal object "MultiDeviceTable".

Methods of MLC

Method	Type	Description
ClearErrors()	Void	Deletes all errors on the control and in the connected sercos drives. Therefore, the MLC has to be in "online" state.
DisableMultiDevice()	Void	Do not use the multi-device functionality. If the multi-device functionality cannot be disabled, an exception is thrown.
DownloadFirmware(string fileName)	Void	Loads the firmware in the "fileName" file to the control. Therefore, the MLC has to be in download mode (sercos phase P0) and the PLC has to be stopped. Parameters: fileName [string], contains the complete name and path of the file of type "FW" in the firmware. Example: "C:\FW\CML45s-MLs-12T01.0040.fw"
EnableMultiDevice()	Void	Using multi-device functionality. The MLC may neither be in "online" state nor in "logged in" state. If the multi-device functionality cannot be used, since the MLC is in "online" state and/or "logged in", an exception is thrown.
GetObject(string object)	object	Returns a reference to an object assigned to the MLC. If the requested object is not known, an exception is thrown. Parameters: Object [string], string of the searched object. Currently, only "Logic" is available as object.
GetPhase()	int	Returns the current sercos phase. Therefore, the MLC has to be in "online" state. If the phase cannot be read, since the MLC is not in "online" state for example, an exception is thrown.
GoOffline()	Void	Inherited from "Device". Cancels a connection between the MLC in the IndraWorks project and a physical MLC. If it cannot be gone offline, an exception is thrown.

XLC/MLC Objects

Method	Type	Description
GoOnline()	Void	<p>Inherited from "Device".</p> <p>Establishes a connection between the MLC in the IndraWorks project and a physical MLC.</p> <p>If the online connection cannot be established, since the PLC has to be stopped or the planned drive configuration does not match, the established connection is aborted and an exception is thrown.</p>
GoOnline2(Scripting.Dictionary args)	Void	<p>Allows the user to specifically control the behavior when going online.</p> <p>Parameters:</p> <p>args [System.Dictionary], all parameters to go online are transferred as key/value pairs in a System.Dictionary object. The following key/value pairs are available. If the specifications are invalid or if going online is not possible, an exception is thrown.</p> <p>Key: "AdjustC2CMotionSercosCycleTimes" [string] [Optional key/value pair]</p> <p>Value: [bool], indicates whether - when using a C2C cross communication module (plug-in module to communicate between controls) - the cycle times for the C2C communication, for the Motion (Motion cycle time) and for the communication with sercos devices (cycle time of the master communication) are to be adjusted if the cycle times differ.</p> <p>True: When using the C2C cross communication module, the cycle times for the Motion and for the master communication (if unequal to the cycle time for the C2C cross communication) are set to the cycle time value for the C2C cross communication. This is required to switch the sercos ring to the operating mode (sercos phase 4).</p> <p>False (default value): When using the C2C cross communication module, the cycle times for the Motion and for the master communication (if unequal to the cycle time for the C2C cross communication) are not changed. In this case (cycle time of the C2C cross communication not equal to the cycle times for the Motion and the master communication), it cannot be switched to the operating mode (sercos phase 4) when using a C2C cross communication module.</p> <p>Example (JavaScript):</p> <pre>args.Add("AdjustC2CMotionSercosCycleTimes", true);</pre> <p>Key: "CanOverwriteProject" [string] [optional key/value pair]</p> <p>Value: [bool], indicates whether a third-party project in the control may be overwritten.</p> <p>True (default value): A third-party project in the control is overwritten when going online.</p> <p>False: A third-party project in the control is not overwritten when going online. Going online is aborted in this case.</p> <p>Example (JavaScript):</p> <pre>args.Add("CanOverwriteProject", false);</pre> <p>Key: "CanStopPlcWhenAxesAreNotActive" [string] [optional key/value pair]</p> <p>Value: [bool], indicates whether the PLC may be stopped if the axes are not active.</p> <p>True: If the axis configuration was changed in IndraWorks, if the PLC is running ("RUN" state) and if no axis is active, the PLC is stopped when going online. If at least one axis is active (real axis "Power on", virtual axis not in "Standstill"), the PLC cannot be stopped. In this case (PLC cannot be stopped), going online is aborted and an exception is thrown.</p> <p>False (default value): If the axis configuration was changed in IndraWorks and if the PLC is running ("RUN" state), the PLC is not stopped when going online. This applies as well if no axis is active. In this case, going online is aborted and an exception is thrown.</p>

XLC/MLC Objects

Method	Type	Description
		<p>Example (JavaScript):</p> <pre>args.Add("CanStopPlcWhenAxesAreNotActive", true);</pre> <p>Key: "CanIgnoreSercosDeviceInconsistencies" [string] [optional key/value pair]</p> <p>Value: [bool], indicates whether a mismatch between the planned sercos devices and the actually existing devices at the sercos bus is to be ignored</p> <p>True (default value): Even if it is determined that the devices connected at the sercos bus do not match the planned sercos devices, it is still tried to go online. If this is not possible, going online is aborted and an exception is thrown.</p> <p>False: Even if it is determined that the devices connected at the sercos bus do not match the planned sercos devices, going online is aborted and an exception is thrown.</p> <p>Example (JavaScript):</p> <pre>args.Add("CanIgnoreSercosDeviceInconsistencies", false);</pre>
ImportParameter(string fileName)	string	<p>Imports the sercos parameters to the "fileName" file. Therefore, the MLC has to be in "online" state.</p> <p>Possible writing errors such as "date and time write-protected" are collected by the method and returned as string at the end of the parameter import. The line format is "IDN ParameterName Result".</p> <p>Parameters:</p> <p>fileName [string], contains the complete name and path of the file to be imported of type "PAR". Example: "C:\Archive\Parameters.par"</p>

Method	Type	Description
ImportParameter2(string fileName, Scripting.Dictionary args)	string	<p>Imports the sercos parameters to the "fileName" file. Therefore, the MLC has to be in "online" state. In contrast to "ImportParameter", it has to be specified which parameter types are to be imported to which axes (or controls).</p> <p>Possible writing errors such as "date and time write-protected" are collected by the method and returned as string at the end of the parameter import. The line format is "IDN ParameterName Result".</p> <p>Parameters:</p> <p>fileName [string], contains the complete name and path of the file to be imported of type "PAR". Example: "C:\Archive\Parameters.par"</p> <p>args [System.Dictionary]</p> <p>All parameter specifications on the parameter import are transferred to a System.Dictionary object. The following keys and values are available:</p> <p>Key: "C-ParameterImportListByNumber" [string] [optional key/value pair], control parameter.</p> <p>Value: [string] indicates whether C-parameters are to be loaded (default).</p> <p>Contains the parameter file "C-parameters". If the C-parameters are to be imported, the string "*->*" has to be specified. If the key "C-ParameterImportListByNumber" is not specified or an empty string "" is specified (default value), the C-parameters are not loaded. If the parameter file does not contain any C-parameters, the specification is ignored.</p> <p>Example (JavaScript):</p> <pre>args.Add("C-ParameterImportListByNumber", "*->*");</pre> <p>Loads the C-parameters from the parameter file to the control.</p> <p>Example (JavaScript):</p> <pre>args.Add("C-ParameterImportListByNumber", "");</pre> <p>This causes that the C-parameters are not loaded to the parameter file.</p> <p>Key: "K-ParameterImportListByNumber" [string] [optional key/value pair], kinematic parameter.</p> <p>Value: [string] indicates whether K-parameters are to be loaded (default).</p> <p>Same behavior as "C-ParameterImportListByNumber"</p> <p>Key: "M-ParameterImportListByNumber" [string] [optional key/value pair], touch probe parameter.</p> <p>Value: [string] indicates whether M-parameters are to be loaded (default).</p> <p>Same behavior as "C-ParameterImportListByNumber"</p> <p>Key: "O-ParameterImportListByNumber" [string] [optional key/value pair], oscilloscope parameter.</p> <p>Value: [string] indicates whether M-parameters are to be loaded (default).</p> <p>Same behavior as "C-ParameterImportListByNumber"</p>

XLC/MLC Objects

Method	Type	Description
		<p>Key: "A-ParameterImportListByNumber" [string] [optional key/value pair], drive-related parameters in the control.</p> <p>Value: [string] indicates whether A-parameters are to be loaded (default) and to which axes.</p> <p>Contains the parameter file "A-parameters" and if the A-parameters are to be loaded, the axis numbers to which the A-parameters have to be loaded can be specified with a comma-separated list. Ranges can also be defined with "-". A-parameters are not loaded with "" (default value). If the key "A-ParameterImportListByNumber" is not specified, the A-parameters in the parameter file are not loaded. If the parameter file does not contain any A-parameters, the specification is ignored.</p> <p>Example (JavaScript):</p> <pre>args.Add("A-ParameterImportListByNumber", "1->1; 2->2,3-6,8");</pre> <p>Loads the A-parameters from the parameter file for the axis with the number 1 to the configured axis with the number 1. The A-parameters in the parameter file for the axis with the number 2 are loaded to the configured axes with the numbers 2, 3, 4, 5, 6 and 8.</p> <p>Example (JavaScript):</p> <pre>args.Add("A-ParameterImportListByNumber", "*->*");</pre> <p>Loads the A-parameters from the parameter file to the axes specified in the parameter file.</p> <p>Example (JavaScript):</p> <pre>args.Add("A-ParameterImportListByNumber", "");</pre> <p>This causes that the A-parameters are not loaded to the parameter file.</p> <p>Example (JavaScript):</p> <pre>// nothing</pre> <p>If the key "A-ParameterImportListByNumber" is not specified, the A-parameters in the parameter file are not loaded.</p> <p>Key: "SP-ParameterImportListByNumber" [string] [optional key/value pair], drive parameters in the drive.</p> <p>Value: [string], indicates whether S/P-parameters can be loaded to the drive (default). Same behavior as "A-ParameterImportListByNumber"</p> <p>Key: "SP-ParameterImportListByAddress" [string] [optional key/value pair], drive parameters in the drive.</p> <p>Value: [string], indicates whether S/P-parameters can be loaded to the drive (default). Same behavior as "SP-ParameterImportListByNumber". Only the sercos addresses have to be specified at "SP-ParameterImportListByAddress" instead of logic axis numbers.</p>
ReadListParameter(string idn)	Rexroth .IndraWorks.Automation.AiList	<p>Reads a list parameter of the control or a sercos device (list parameter = parameter of variable length). The individual elements of the list parameter are returned in an "AiList" as a list of strings [string].</p> <p>Parameters:</p> <p>idn [string], ident number of the parameter, such as "C-0-0001" or "C-0-0001.0.0". For axis-related parameters (A-, S-, P-parameters) and for kinematic parameters (K-parameters), the 3-digit reference number has to be specified as well such as "A001:A-0-0001" or "A001:A-0-0001.0.0" as well as "A001:S-0-0001" or "K001:K-0-0001".</p>

XLC/MLC Objects

Method	Type	Description
ReadParameter(string idn)	string	Reads a "normal" parameter of the control or a sercos device. Parameters: idn [string], ident number of the parameter. The same as under "ReadListParameter" applies to the spelling of the ident number.
SwitchPhase(int phase)	Void	Switches to the sercos phase. Therefore, the MLC has to be in "online" state. If the phase cannot be switched, since an invalid phase was specified for example or the master axis is enabled, an exception is thrown. Parameters: phase [int], contains the target phase in numbers. The following target phases are allowed: <ul style="list-style-type: none"> • 0 (download mode) • 2 (parameterization mode) • 4 (operating mode)
WriteListParameter(string idn, Rexroth.IndraWorks.Automation.AiList values)	Void	Writes a list parameter of the control or a sercos device (list parameter = parameter of variable length). Therefore, the MLC has to be in "online" state. The individual elements of the list parameter have to be transferred to an "AiList" as a list of strings [string]. Parameters: idn [string], ident number of the parameter. The same as under "ReadListParameter" applies to the spelling of the ident number. values [Rexroth.IndraWorks.Automation.AiList], contains the elements of the list parameter to be written as list of strings [string].
WriteParameter(string idn, string value)	Void	Writes a "normal" parameter of the control or a sercos device. Therefore, the MLC has to be in "online" state. Parameters: idn [string], ident number of the parameter. The same as under "ReadListParameter" applies to the spelling of the ident number. value [string], contains the value to be written.

6.5 Adding an MLC

Using the Basis Automation Interface object **Project**, MLC devices can be added to the current project below the project node. As for the XLC device, the **CreateChild** method of the **Project** object has to be called to add an MLC.

XLC/MLC Objects

Object - Project, "CreateChild" method

Method	Type	Description
CreateChild(string childLibGuid, object args)	Rexroth .IndraWorks.Automation.ProjectElement	<p>This method of the Basis Automation Interface can create a "ProjectElement" of the types "File", "Folder" or "Device" as child of the project node. To create a "ProjectElement" of the type "MLC", the following arguments have to or can be used.</p> <p>Creating an MLC:</p> <p>Parameters:</p> <p>childLibGuid [string], the object ("MLC") to be created.</p> <p>args [System.Dictionary]</p> <p>The property values of the MLC are transferred as key/value pairs to a System.Dictionary object. The following keys and values are available:</p> <p>Key: "Hardware" [string]</p> <p>Value: [string], hardware name. The following formats are supported:</p> <ul style="list-style-type: none"> • "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" <p>Only the front part is currently evaluated, such as "CML25.1-3N". If the back part is also indicated, the format has to correspond to the given format.</p> <p>Key: "Firmware" [string]</p> <p>Value: [string], firmware name. The following firmware names are supported:</p> <ul style="list-style-type: none"> • "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" <p>Key: CMLx5 = CML25, CML45, CML65</p>

Method	Type	Description
		<p>The firmware has to "fit" the hardware. No hardware "CML25.1-3N" may be used with the firmware "FWA-CML45*-ML*-12VRS" for example (conflict L25, L45). When specifying "VRS" instead of "V02" for example, the latest firmware release is automatically used.</p> <p>Key: "Name" [string] [optional key/value pair] Value: [string], Control name. If no name is given, it is automatically specified.</p> <p>Key: "IpAddress" [string] [optional key/value pair] Value: [string], IP address of the control. If no IP address is specified, it is automatically specified.</p> <p>Key: "PlcGateway" [string] [optional key/value pair] Value: [string], IP address of the PLC gateways. If no PLC gateway is specified, "localhost" is used.</p> <p>Key: "PlcSecureOnlineMode" [string] [optional key/value pair] Value: [string], "True" if the secure online mode is to be used, otherwise "False" (default).</p> <p>Key: "Author" [string] [optional key/value pair] Value: [string], Name of the person who added the control to the project.</p> <p>Key: "Comment" [string] [optional key/value pair] Value: [string], Comment.</p>

6.6 Object - MultiDeviceTable

The **MultiDeviceTable** object is an XLC/MLC-specific object which exclusively provides attributes and methods for the multi-device functionality. This object is accessed from the objects **XLC** or **MLC** via the **MultiDeviceTable** attribute.

Attribute of MultiDeviceTable

Attribute	Type	Description
DeviceNames	Rexroth.IndraWorks.Automation.AiCollection	<p>Read-only.</p> <p>Provides a string collection with the names of all devices in the multi-device table.</p>

Methods of MultiDeviceTable

Method	Type	Description
ActivateDevice(string deviceName)	Void	<p>Enables a device in the multi-device table. The XLC/MLC may neither be in "online" state nor in "logged in" state.</p> <p>The activated device can then be used in IndraWorks. It can be used for a PLC program download or a parameter import.</p> <p>If the specified device cannot be activated, since it is not in the multi-device tables, an exception is thrown.</p> <p>Parameters:</p> <p>deviceName [string]: Name of a device in the multi-device tables such as "XLC_21" or "MLC_21".</p>

7 Example Codes - XLC/MLC Objects

7.1 Example Code - Access to the XLC/MLC

This example can also be used for an XLC if the `GetChildrenOfType("XLC ", false)` call is replaced by `GetChildrenOfType("XLC", false)`.

Program:

```
// 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 Example Code - Going Online/Offline

This example can also be used for an XLC if the `GetChildrenOfType("XLC ", false)` call is replaced by `GetChildrenOfType("XLC", false)`.

Example Codes - XLC/MLC Objects

Program:

```
// 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 Example Code - Extended Online/Offline Going

This example can also be used for an XLC if the `GetChildrenOfType("XLC", false)` call is replaced by `GetChildrenOfType("XLC", false)`.

Program:

```
// 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
```

Example Codes - XLC/MLC Objects

```

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();

// 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 Example Code - Reading/Writing Phase

This example can also be used for an XLC if the `GetChildrenOfType("XLC ", false)` call is replaced by `GetChildrenOfType("XLC", false)`.

Program:

```

// 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;

```

Example Codes - XLC/MLC Objects

```

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();

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 Example Code - Importing Parameter

This example can also be used for an XLC if the `GetChildrenOfType("XLC ", false)` call is replaced by `GetChildrenOfType("XLC", false)`.

Program:

```

// 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)

```

Example Codes - XLC/MLC Objects

```

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 Example Code - Extended Parameter Import

This example can also be used for an XLC if the `GetChildrenOfType("XLC", false)` call is replaced by `GetChildrenOfType("XLC", false)`.

Program:

```

// 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")

```

Example Codes - XLC/MLC Objects

```

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 Example Code - Reading/Writing Parameter

This example can also be used for an XLC if the `GetChildrenOfType("XLC ", false)` call is replaced by `GetChildrenOfType("XLC", false)`.

Program:

```

// 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

```

Example Codes - XLC/MLC Objects

```

// 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 Example Code - Firmware Download

This example can also be used for an XLC if the `GetChildrenOfType("XLC ", false)` call is replaced by `GetChildrenOfType("XLC", false)`.

Program:

```

// 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();
}

```

Example Codes - XLC/MLC Objects

```

// 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 Example Code - Adding XLC/MLC

Program:

```

// 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");
}

```

Example Codes - XLC/MLC Objects

```

// 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 Example Code - Using/Not Using Multi-Device

This example can also be used for an MLC if the `GetChildrenOfType("XLC", false)` call is replaced by `GetChildrenOfType("MLC", false)`.

Program:

```

// 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();
}

```

Example Codes - XLC/MLC Objects

```

// 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 Example Code - Activating an XLC/MLC from the Multi-Device Table

This example can also be used for an MLC if the `GetChildrenOfType("XLC", false)` call is replaced by `GetChildrenOfType("MLC", false)`.

Program:

```

// 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

```

Example Codes - XLC/MLC Objects

```

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 Example Code - Accessing the "Logic" Object

This example can also be used for an XLC if the `GetChildrenOfType("XLC ", false)` call is replaced by `GetChildrenOfType("XLC", false)`.

Program:

```

// 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 ===");
}

```

Example Codes - XLC/MLC Objects

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

8 IL Objects

8.1 Object – Logic

The **Logic** object represents the PLC Logic sub-object.

Methods of Logic

Method	Type	Description
GetObject(string object)	object	Provides the reference to an object assigned to the "Logic" object. Parameter object [string], name of the searched object. Currently, the "Application" object is allowed. Example: <pre>var logic = device.GetObject("Logic");</pre>

8.2 Object – Application

The **Application** object represents an application below the **Logic** object. Different functions of an application can be executed via this object.

To access the application via the Automation Interface, the following commands are available:

Attributes of Application

Method	Type	Description
IsRunning	bool	Returns whether the PLC of the application is running. True: PLC running. False: PLC not running.
IsLoggedIn	bool	Returns whether logged in on the application. True: Logged in on application False: Not logged in.

IL Objects

Methods of Application

Method	Type	Description
Login()	string	<p>Logging to the application.</p> <p>If the PLC program has been changed, it is compiled and subsequently downloaded.</p> <p>If an error occurs (e.g. if the PLC program could not be compiled successfully), an exception is thrown. Otherwise, compiler warnings and other outputs are returned as string.</p> <p>Example:</p> <pre>var application = logic.GetObject("Application"); application.Login();</pre>
Login2(int mode)	string	<p>Logging to the application. If the PLC program was changed, it is compiled and subsequently downloaded as follows:</p> <p>Parameters: mode [int],</p> <ul style="list-style-type: none"> • 0: Log in without change • 1: Log in with online change • 2: Log in with download • 3: If an online change is possible, log in with online change; otherwise download <p>If an error occurs (e.g. if the PLC program could not be compiled successfully), an exception is thrown. Otherwise, compiler warnings and other outputs are returned as string.</p> <p>Example:</p> <pre>var application = logic.GetObject("Application"); application.Login2(0);</pre>
Logout()	Void	Executes the command "Log out".
Build()	string	<p>Compiles the PLC program.</p> <p>Provides the occurred errors as string. If no error occurred, an empty string is returned.</p>
Rebuild()	string	<p>Recompiles the PLC program.</p> <p>Provides the occurred errors as string. If no error occurred, an empty string is returned.</p>
Download()	string	<p>Transmits the PLC program to the control.</p> <p>If the PLC program was changed, it is compiled again before.</p> <p>If an error occurs (e.g. if the PLC program could not be compiled successfully), an exception is thrown. Otherwise, compiler warnings and other outputs are returned as string.</p> <p>Example:</p> <pre>var application = logic.GetObject("Application"); var erg = application.Download();</pre>

IL Objects

Method	Type	Description
Download2(int mode)	string	<p>Transfers the PLC program to the control as follows:</p> <p>Parameters: mode [int],</p> <ul style="list-style-type: none">• 1: Log in with online change.• 2: Log in with "Download".• 3: If an online change is possible, log in with online change; otherwise download. <p>If an error occurs (e.g. if the PLC program could not be compiled successfully), an exception is thrown. Otherwise, compiler warnings and other outputs are returned as string.</p> <p>Example:</p> <pre>var application = logic.GetObject("Application"); var erg = application.Download2(0);</pre>
Start()	Void	Starts the application.
Stop()	Void	Stops the application.

9 Service and Support

Our worldwide service network provides an optimized and efficient support. Our experts offer you advice and assistance should you have any queries. You can contact us **24/7**.

Service Germany Our technology-oriented Competence Center in Lohr, Germany, is responsible for all your service-related queries for electric drive and controls.

Contact the **Service Helpdesk & Hotline** under:

Phone:	+49 9352 40 5060
Fax:	+49 9352 18 4941
E-mail:	service.svc@boschrexroth.de
Internet:	http://www.boschrexroth.com

Additional information on service, repair (e.g. delivery addresses) and training can be found on our internet sites.

Service worldwide Outside Germany, please contact your local service office first. For hotline numbers, refer to the sales office addresses on the internet.

Preparing information To be able to help you more quickly and efficiently, please have the following information ready:

- Detailed description of malfunction and circumstances resulting in the malfunction
- Type plate name of the affected products, in particular type codes and serial numbers
- Your contact data (phone and fax number as well as your email address)

Index

A

About this documentation.....	3
Validity of the documentation	3
Access devices.....	33
Access to "Logic" object.....	67
Access to the MLC.....	57
Access to the XLC.....	57
Activate an XLC/MLC from a multi-device table.	66
Add folders and files.....	30
Add MLC.....	64
Add XLC.....	64
Application	
Attributes	69
Methods	70
Object	69
Archive a project.....	34
Automation Interface	
Example codes, XLC/MLC	57
IL objects	69
Scripting	12
Using automation scripts	12
XLC/MLC objects	39

C

Create an IndraWorks project.....	29
-----------------------------------	----

E

Example code, XLC/MLC	
Access to "Logic" object	67
Example codes, XLC/MLC.....	57
Access to the MLC	57
Access to the XLC	57
Activate an XLC/MLC from a multi-device table	66
Add MLC	64
Add XLC	64
Extended online/offline going	58
Extended parameter import	61
Firmware download	63
Go online/offline	57
Import parameter	60
Read/write parameter	62
Read/write phase	59
Using/not using multi-device	65
Extended online/offline going.....	58
Extended parameter import.....	61

F

Firmware download.....	63
------------------------	----

G

Go online/offline.....	57
------------------------	----

I

IL objects.....	69
Import parameter.....	60
Initialize the Automation Interface.....	29
Intended use	
Use cases	9
Intended Use	
Introduction	9

L

Logic	
Methods	69
Object	69

M

MLC	
Add	53
Attributes	48
Methods	48
Object model	39
Objects	47
MultiDeviceTable	
Attributes	55
Methods	55
Object	55

O

Open an IndraWorks project.....	30
---------------------------------	----

R

Read/write parameter.....	62
Read/write phase.....	59
Restore a project.....	35

S

Scripting.....	12
Search in projects.....	31
Support	
See service hotline	73

U

Unintended use	
Consequences, exclusion of liability	9
Unintended Use.....	10
Using/not using multi-device.....	65
Using automation scripts.....	12

V

Validate a project.....	36
-------------------------	----

Index

X

XLC	
Add	45
Attributes	40

X

...XLC	
Methods	40
Object	39
Object model	39

Notes

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



R911336356

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